



InformaticaTM

Customizing

Informatica MDM - Product 360

Version: 10.5 HotFix 3 SP 1

Table of Contents

1	Integrate with Product 360	21
2	Customize Product 360	21
3	API	21
3.1	Java API	21
3.1.1	Internal	21
3.1.2	Normal	21
3.1.3	API	21
3.1.4	Upgrade path:	22
3.1.5	Modification vs. Customizing	22
3.2	Persistence Layer	22
3.3	Domain Model / Repository	22
4	Software Development Kit	23
4.1	SDK Package.....	23
4.2	Prerequisites	23
4.2.1	Database.....	23
4.2.2	Java Development Kit.....	24
4.2.3	Eclipse.....	24
4.3	Installation of the SDK Online Help.....	24
4.4	Installation of the SDK	25
4.4.1	Install target definition	27
4.4.1.1	SDK Feature	31
4.4.2	Install launches and configuration	35
4.4.2.1	Launch Product 360 Server	37
4.4.2.2	Launch Product 360 Client	37
4.4.2.3	Launch Product 360 Repository Manager.....	37
4.4.2.4	Launch PIM Database Setup	38
4.4.3	Install examples	38
4.4.3.1	Examples at a glance	38
5	Application Layer	40
5.1	Article.....	41

5.1.1	Article Structure Map	43
5.1.1.1	Development Examples	43
5.2	Attributes	45
5.2.1	Attribute value datatype rules regarding decimal values	45
5.3	Item/Variant/Product parent references	46
5.3.1	Background information	46
5.3.2	Datamodel - Repository	46
5.3.2.1	Types area	46
5.3.2.2	Custom area	47
5.3.2.3	Future Plans	51
5.3.3	How to resolve all objects with a parent relationship to one of the objects in a given list	51
5.3.3.1	2 PPD: Resolve all assigned items for a list of products	51
5.3.3.2	3 PPD: Resolve all assigned variants for a list of products	51
5.3.3.3	3 PPD: Resolve all assigned items for a list of variants	51
5.3.4	How to resolve all parent objects for a list of objects	52
5.3.4.1	2 PPD: Resolve all parent products for a list of items	52
5.3.4.2	3 PPD: Resolve all parent variants for a list of items	52
5.3.4.3	3 PPD: Resolve all parent products for a list of variants	53
5.3.5	Creating a superordinate object	53
5.3.5.1	Exclude fields	53
5.4	Key Performance Indicator customization	53
5.4.1	Motivation	53
5.4.2	Extension point definition	54
5.4.3	Custom implementations	56
5.4.3.1	Contributing a KPI	56
5.4.3.2	Additional KPI setting	56
5.4.4	Scheduling KPI implementation	57
5.4.4.1	Validating the calculation results	57
5.5	Datamodel	58
5.6	Structure	59
5.6.1	Copy Structures and/or StructureGroups	59
5.6.1.1	PIM 8 Copy Structures Workflow Structures of which structure types can be copied? Structure Object Rights/ACLs What else is copied, what is not? Copy StructureGroups - General Workflow Workflow Strategies CopyAllStructureGroupsOfStructure CopyStructureGroup (within one structure) CopyStructureGroupInOtherStructure	59

5.6.1.2	PIM 8	59
5.6.1.3	Copy Structure Attributes.....	63
5.6.1.4	Copy Structure Groups	64
5.6.1.5	Copy Structure Values.....	66
5.6.2	Structure Group	68
5.6.2.1	Inheritance of StructureGroupFeatures to items/products/variants	68
5.6.2.2	Sorting of structure groups (StructureGroup.DisplayOrder)	70
6	Server and Platform	71
6.1	Architecture principles	71
6.1.1	Domain model centric	72
6.1.2	A base for customer specific solutions.....	72
6.1.3	Migrate customer specific solutions to newer versions at low cost	72
6.1.4	Solution provides rich client UI as a desktop application	72
6.2	Architecture main concepts	72
6.2.1	Advantages of the approach.....	73
6.3	Multi-Server Architecture	74
6.3.1	Synopsis.....	75
6.3.2	Demand for Multi-Server	75
6.3.2.1	Response times worsen with more and more users working concurrently.....	75
6.3.2.2	Background jobs interfere with users	75
6.3.2.3	More channels with bigger assortments need to be updated	75
6.3.2.4	High-End Multi-Core CPU Systems are expensive	76
6.3.3	Architecture.....	76
6.3.3.1	Server Roles.....	78
6.3.3.2	Load Balancing.....	78
6.4	Domain Model (Repository)	79
6.4.1	General architecture	81
6.4.2	Types Part.....	82
6.4.2.1	Entity Type.....	82
6.4.3	Custom Part.....	87
6.4.3.1	Entities.....	87
6.4.3.2	Audit Trail Settings.....	99
6.4.3.3	Enumeration.....	99
6.4.3.4	Permission Category	101

6.4.3.5	Category	101
6.4.4	Multi-language support of the repository.....	102
6.4.4.1	Referencing on logical keys in field names.....	102
6.4.5	Picture-clause syntax.....	102
6.4.6	Repository Manager	103
6.4.7	Assignment of Entity IDs	103
6.4.7.1	Validation	103
6.4.7.2	Entity id overview.....	104
6.4.8	Lookups aka Dynamic Enumerations	109
6.4.8.1	Background information on Enumerations.....	110
6.4.8.2	Configure the Repository.....	111
6.4.8.3	Create the Lookup.....	115
6.4.8.4	Proposal Enumerations	115
6.4.9	How to register and use a characteristic value provider with parameters	116
6.4.9.1	Introduction	116
6.4.9.2	Create Plugin using SDK	116
6.4.9.3	Deploy the plugin and restart Product 360 server.....	120
6.4.9.4	Attaching value provider to a characteristic.....	120
6.4.9.5	Characteristic UI showing the provided values.....	121
6.4.9.6	DependentLookup Characteristic	121
6.4.9.7	Setup of Lookup Value Reference	122
6.4.9.8	Characteristic modeling:	129
6.5	Data Access.....	131
6.5.1	Overview.....	131
6.5.2	Detail access.....	132
6.5.3	List access.....	132
6.5.4	Data Models.....	133
6.5.4.1	Model Overview.....	133
6.5.4.2	Entity Item	133
6.5.4.3	Entity Proxy	134
6.5.4.4	Detail Model.....	134
6.5.4.5	List Model	136
6.5.5	Data Selection	137
6.5.5.1	Entity Reporting	137
6.5.5.2	Entity Search	151

6.5.6	Data Navigation.....	163
6.5.6.1	Entity Relations	163
6.5.7	Data Security	163
6.5.8	Command Framework	163
6.5.8.1	General	164
6.5.8.2	Commands	166
6.5.8.3	Operators.....	166
6.5.8.4	Create Command	167
6.5.8.5	Add Command	170
6.5.8.6	Put Command	173
6.5.8.7	Validate Command	177
6.5.8.8	Remove Command	178
6.5.9	Spellchecking API.....	178
6.5.9.1	General	178
6.5.9.2	Architecture.....	179
6.5.9.3	Usage	179
6.5.9.4	Examples	181
6.6	Data Audit.....	183
6.6.1	Audit Trail	183
6.6.2	Audit Log.....	183
6.6.2.1	Physical audit log.....	184
6.6.2.2	Logical audit log.....	184
6.6.3	Audit Trail Extension Points	187
6.6.3.1	The extension point ChangeSummaryTransformer.....	187
6.6.3.2	The extension point historyTabToDetailViewBinder	189
6.6.4	Entity item change document	190
6.6.4.1	Datatypes	190
6.6.4.2	Meta attributes	191
6.6.4.3	Change Summary.....	193
6.7	Persistence	204
6.7.1	Detail persistence.....	205
6.7.2	List persistence	205
6.7.3	Business and persistence models	206
6.7.4	Mapping.....	206
6.7.4.1	Persistence mapping in repository	207

6.7.4.2	Persistence meta model	209
6.7.5	Detail Model Persistence	209
6.7.5.1	Persister.....	210
6.7.5.2	Mediators.....	211
6.7.5.3	Audit Log.....	213
6.7.6	List Model Persistence	216
6.7.7	Persistence Model	216
6.7.7.1	Data sources.....	216
6.7.7.2	Persistence model.....	217
6.7.7.3	Named queries	226
6.7.7.4	3rd party libraries.....	227
6.7.8	Debug Persistence Layer	228
6.7.8.1	Log traces	228
6.7.8.2	Extension tracing	229
6.7.8.3	SQL traces.....	229
6.7.8.4	JDBC Driver trace	229
6.8	Infrastructure	229
6.8.1	Identifier Generator	229
6.8.1.1	Default Implementation	229
6.8.1.2	numericId	230
6.8.1.3	Custom Implementation	231
6.8.2	Initializer Framework.....	231
6.8.2.1	Extension Point	232
6.8.2.2	Disposers	232
6.8.3	File Transfer.....	233
6.8.3.1	File Storage.....	233
6.8.3.2	Extension Point	234
6.8.4	Server Job Execution	234
6.8.4.1	Server Job States	234
6.8.4.2	Load Balancing.....	235
6.8.4.3	Customizing.....	235
6.8.4.4	Special handling for Non Scheduled jobs.....	236
6.8.4.5	Preferences.....	236
6.8.5	Security.....	236
6.8.5.1	Encryption of secure information	236

6.8.6	XML Serialization	238
6.8.6.1	Overview	238
6.8.6.2	Services and Interfaces	238
6.8.6.3	Migration to DomPersistable2	241
6.8.7	Message Queue Framework	241
6.8.7.1	Message queue definition in the server.properties file:	242
6.8.7.2	Reading from a message queue:	242
6.8.7.3	Writing to a message queue:	244
6.8.7.4	Answering to a specific message queue:	245
7	Desktop Client	248
7.1	"Image preview" - selection handling extensions	249
7.1.1	Motivation	249
7.1.2	Extension point definition	250
7.1.3	Standard implementations	251
7.1.4	Custom implementations	252
7.2	Automatically set structure group context in the Article Attribute View via customizing	252
7.3	Centralized UI settings	252
7.3.1	Reason	252
7.3.2	Client property	253
7.3.3	How it works	253
7.3.4	Problems	253
7.3.5	Examples	253
7.3.6	References	255
7.3.7	Limitations	255
7.4	CKEditor configuration and customization	255
7.4.1	Motivation	255
7.4.2	Big picture	256
7.4.3	Extension point	257
7.4.3.1	ToolbarConfigProvider	257
7.4.3.2	SpecialChar	257
7.4.3.3	Skin	257
7.4.3.4	Additional plugins	259
7.4.3.5	Remove plugins	260
7.4.3.6	Content CSS	260

7.4.3.7	Style Set.....	261
7.4.4	Providing custom resources.....	262
7.4.5	Additional special characters.....	263
7.4.6	Toolbar configuration.....	264
7.5	CKEditor troubleshooting.....	265
7.5.1	How to enable the debugging of CKEditor	265
7.6	Creating custom views.....	266
7.6.1	View types.....	266
7.6.1.1	List model views.....	266
7.6.1.2	Detail model views.....	267
7.6.2	View contribution.....	267
7.6.3	PermissionProvider	269
7.6.4	Overwriting view methods	270
7.6.4.1	initEntityMapping()	270
7.6.4.2	getEntityForPermissions().....	271
7.6.4.3	initPermissions()	271
7.6.4.4	createContentProvider().....	272
7.6.4.5	createDatasetCreationMechanism()	272
7.6.4.6	initTableConfigGroup().....	273
7.6.5	Customization opportunities	273
7.6.5.1	Selection handling	273
7.6.5.2	Data maintenance.....	274
7.6.6	Example ListModelTableView.....	275
7.6.7	Example EntityDetailModelTableView.....	277
7.7	Creating custom perspectives.....	278
7.7.1	Creating perspectives	278
7.7.2	Adding views to a perspective	280
7.8	Customization of the field handling within views	281
7.8.1	Motivation	281
7.8.2	Extension point	281
7.8.2.1	Availability	282
7.8.3	View extension manager implementation.....	286
7.8.3.1	TableCellValueProvider	286
7.8.3.2	CellEditor	287
7.8.3.3	ICreateMechanism	287

7.8.3.4	IDoubleClickListener.....	287
7.9	Defining the default perspective	288
7.9.1	Summary	288
7.9.2	Example	289
7.10	Entity specific icons	290
7.11	Extension point for SearchView customization.....	293
7.11.1	Predefined search parameters.....	293
7.11.2	Different result view	295
7.11.3	Download	295
7.12	HOWTOs.....	295
7.12.1	How to center an image in a table cell.....	296
7.12.2	How to contribute your own status line in the table view	296
7.12.3	How to customize the status text in the table view	297
7.12.4	How to add a custom menu item to the text celleditor context menu	297
7.12.5	How to contribute a context menu item for a specific Entity	299
7.12.6	How to create a dynamic menu contribution item	299
7.12.7	How to create/override a key binding for a command contribution	301
7.12.8	How to add a permission to popupmenu action.....	302
7.13	ListModelTableViews: selection behavior	302
7.13.1	Problem definition	302
7.13.2	Solution	303
7.13.3	Adverse impacts.....	303
7.13.4	Examples	303
7.13.4.1	Command handler	303
7.13.4.2	Custom views	304
7.13.5	Don'ts.....	305
7.14	Long-running processes	305
7.14.1	Rule number one: Do not block the UI thread!.....	305
7.14.2	Best Practice.....	305
7.15	Possibility to disable standard UI elements	307
7.15.1	Activities	307
7.15.2	General information.....	307
7.15.3	Views.....	307
7.15.4	Perspectives	307

7.15.5	Menu items	309
7.15.5.1	menuContribution -> command	309
7.15.5.2	menuContribution -> dynamic	310
7.15.5.3	menuContribution -> menu	310
7.15.5.4	viewContribution -> action	311
7.15.5.5	objectContribution -> action	312
7.15.6	Preference pages.....	312
7.15.7	Drop listeners	313
7.15.8	Status lines	314
7.15.9	Status text resolvers	314
7.15.10	FieldBasedTableViewExtensionManager.....	315
7.15.11	Complete deletion of action sets	315
7.15.12	Download	316
7.16	Possibility to replace the execution class of a view contribution.....	316
7.16.1	Motivation	316
7.16.2	Solution	316
7.16.2.1	Base class	317
7.16.2.2	Priority	317
7.16.2.3	Additional attributes.....	317
7.16.3	Outlook	318
7.16.4	Download	318
7.17	Richtext conversion	318
7.17.1	Media-neutral format.....	319
7.17.2	Internal markup	319
7.17.3	Overview of single conversion steps	319
7.17.4	Customization of individual conversion steps	320
7.17.5	Sample implementation of a custom conversion handler	321
7.18	Richtext validation	323
7.18.1	Text features.....	323
7.18.2	Validation of text formattings	325
7.18.3	RichTextValidators extension point	326
7.19	Table View Layout	327
7.20	UI Style Guide	327
7.20.1	Style Guide Is Not a UI Specification	329

7.20.2	Layout	329
7.20.2.1	Workbench	330
7.20.2.2	Perspectives	330
7.20.2.3	Views	330
7.20.2.4	Editors.....	330
7.20.3	Text	330
7.20.4	Components	331
7.20.4.1	Commands	331
7.20.4.2	Menus.....	331
7.20.4.3	Toolbars.....	331
7.20.4.4	Status Bar	331
7.20.4.5	Dialogs	331
7.20.4.6	Wizards	331
7.20.5	Interaction style	331
7.20.5.1	Drag & Drop	331
7.20.5.2	Context menu	331
7.20.5.3	Keyboard navigation.....	332
7.20.6	Error Handling.....	332
7.20.7	Language	332
7.20.8	Icons.....	332
7.20.9	Preferences.....	332
7.20.10	Help.....	332
8	Web Client	332
8.1	Technical Overview	332
8.2	Product 360 Web UI and Application Server	332
8.3	Technology Stack.....	333
8.3.1	OSGi	333
8.3.2	Guice	334
8.3.3	Vaadin	334
8.4	Asynchronous loading of custom dashboard components.....	334
8.4.1	Introduction	334
8.4.2	Technical limitations	334
8.4.2.1	Asynchronous loading and Vaadin push mode detection	335
8.4.2.2	Getting AsyncServerOperationExecutorFactory	336

8.4.2.3	Calling AsyncServerOperationExecutorFactory	336
8.4.2.4	Implementing Callable< ResultType >	337
8.4.2.5	Implementing ServerOperationCallback< ResultType >	337
8.4.2.6	Purpose of using a generic < ResultType >	338
8.4.2.7	Extending AbstractAsyncLoadingFlexComponent.....	338
8.4.3	User interface	339
8.4.4	Piecing it all together	340
8.4.5	Thread pool for background execution	342
8.5	Customizing the Application Theme.....	342
8.5.1	Introduction	342
8.5.2	Creating a new theme fragment.....	342
8.5.3	Enabling theme	345
8.5.4	Example of new theme	345
8.5.5	Support of multiple themes	345
8.5.6	Customize Login/SAML/Supplier Portal integration favicon and html page title	346
8.6	Programmatic Customization of Web UI	348
8.6.1	SDK Basics	348
8.6.1.1	Create a new OSGi bundle skeleton.....	348
8.6.1.2	Launch customized application	351
8.6.1.3	Extending existing modules	352
8.6.2	Typical Use Cases.....	352
8.6.2.1	Add a custom tab in the detail area	353
8.6.2.2	Add a custom context area	354
8.6.2.3	Add a custom action to the table menu bar	356
8.6.2.4	Implementing and contributing custom flex components for dashboard	358
8.6.2.5	Contribute Custom Placeholder for XML Configuration Files.....	359
8.6.2.6	Hide or show detail tabs dependent of arbitrary state of selected object.....	361
9	Web Search.....	362
9.1	Repository Customizing.....	362
10	Supplier Portal	363
10.1	Overview Customizing	363
10.2	Subpages.....	364
10.3	High Level Architecture.....	364
10.4	Styling and E-Mail Templates.....	365

10.4.1	Customizing Mail Templates	366
10.4.1.1	List of all Templates	366
10.4.2	Customizing Static HTML Files, e.g. Imprint	369
10.4.2.1	List of Static HTML Files	370
10.4.3	Customizing Styles (CSS)	371
10.4.3.1	Example	371
10.4.4	Build and Deploy Customizings	372
10.5	Customize User Interface.....	373
10.5.1	Customize Application Navigation Bar	373
10.5.1.1	Add new element to Navigation Bar	374
10.5.2	User Permissions control available list of Actions	379
10.5.3	Supplier Portal as embedded application	380
10.5.3.1	Display as embedded application.....	381
10.5.3.2	Initialize with a specific locale.....	381
10.5.3.3	Deep links to the application.....	381
10.5.4	Add custom pages to the support box	382
10.6	Manage and Customize Workflows	383
10.6.1	Introduction	384
10.6.2	Customizing Workflows Definitions	384
10.6.2.1	Changing a workflow definition	385
10.6.2.2	Example - Proceed a running workflow (receive task)	385
10.7	Supplier Portal REST Interface	386
10.7.1	General Remarks	386
10.7.1.1	Authentication and Localization	386
10.7.2	REST API Resources	386
10.7.2.1	Status & SSO	386
10.7.2.2	Feeds.....	388
10.7.2.3	Invitation	388
10.7.3	Generate SSO Token	388
10.7.3.1	HTTP Method Type.....	389
10.7.3.2	Resource URL	389
10.7.3.3	Response	389
10.7.3.4	Example	389
10.7.4	Get a list of invitable suppliers	390
10.7.4.1	HTTP Method Type.....	390

10.7.4.2	Resource URL	390
10.7.4.3	Response	390
10.7.5	Get invitable supplier by HpmSupplierId	391
10.7.5.1	HTTP Method Type.....	391
10.7.5.2	Resource URL	391
10.7.5.3	Parameters	391
10.7.5.4	Response	392
10.7.6	Get status.....	392
10.7.6.1	HTTP Method Type.....	393
10.7.6.2	Resource URL	393
10.7.6.3	Response	393
10.7.6.4	Example Usage:.....	393
10.7.7	Invite a supplier user.....	393
10.7.7.1	HTTP Method Type.....	394
10.7.7.2	Resource URL	394
10.7.7.3	Parameters	394
10.7.7.4	Response	395
10.7.7.5	Decision diagram: Is user invitable?	396
10.7.8	Post a timeline message	396
10.7.8.1	HTTP method Type	396
10.7.8.2	Resource URL	397
10.7.8.3	Parameters	397
10.7.8.4	Response	398
10.7.8.5	Example Usage:.....	399
10.8	Supplier Portal Remote API	399
10.8.1	Overview	399
10.8.1.1	Authentication & Security.....	400
10.8.2	Getting Started	400
10.8.2.1	Sample Code - Create new supplier	401
10.8.2.2	Sample Code - Get information about logged in user.....	402
10.8.2.3	List of Available Services.....	402
10.8.3	Further Reading	404
10.9	Supplier Portal I18n	404
10.9.1	Overview	404
10.9.2	Install Additional Language Pack.....	404

10.10	Customized Data Model for Supplier Data.....	405
10.10.1	Introduction	405
10.10.2	Adding User and Supplier fields	406
10.10.3	Field Permissions	416
10.10.3.1	User Fields	416
10.10.3.2	Supplier Fields.....	416
11	Tooling.....	416
11.1	Run-time Tools	416
11.1.1	Command Inspector	416
11.1.1.1	Installation and Execution.....	417
11.1.2	Eclipse Plug-In Spy.....	417
11.1.2.1	Installation and Execution.....	418
11.1.2.2	Screenshots	418
11.2	Development Tools	421
12	Database.....	421
12.1	Contribute custom database scripts.....	421
12.2	Custom Update Scripts.....	421
13	Integration.....	422
13.1	Database Access.....	422
13.2	File Based Integration	422
13.2.1	Import	422
13.2.2	Export	422
13.3	Rest Based Integration.....	422
13.3.1	Service API	422
13.3.2	Custom REST Services	422
13.4	Message Queue Based Integration.....	422
13.4.1	Message Queue API	422
13.5	REST Service Infrastructure.....	423
13.5.1	Overview	424
13.5.2	Security	425
13.5.3	Create your own REST service	425
13.5.3.1	REST Resource Example	426
13.6	REST Service API	428

13.6.1	Available APIs	429
13.6.1.1	List API	429
13.6.1.2	Management API	430
13.6.1.3	Enum API	430
13.6.1.4	Meta API	430
13.6.1.5	Media API	430
13.6.1.6	Security API	430
13.6.1.7	Object API	430
13.6.2	Getting Started	430
13.6.2.1	Authentication	430
13.6.3	General Information.....	430
13.6.3.1	Data Representation	430
13.6.3.2	Entity Item Reference	430
13.6.3.3	Field Qualification	430
13.6.3.4	Search Query Language	431
13.6.3.5	Rest Java Client	431
13.6.3.6	Rest API Versions	431
13.6.4	REST General Information	433
13.6.4.1	Monitoring	433
13.6.4.2	REST Service Authentication	434
13.6.4.3	REST Datatypes	436
13.6.4.4	REST Field Qualification	441
13.6.4.5	REST Transition Fields	446
13.6.4.6	REST Search Query Language	447
13.6.4.7	REST Java Client	450
13.6.4.8	REST Characteristic.....	452
13.6.5	REST List API.....	463
13.6.5.1	Overview	464
13.6.5.2	List-Based Mass Data Access	464
13.6.5.3	Root Entity vs. Sub Entity Access	464
13.6.5.4	Report vs. Search	464
13.6.5.5	Report Services	464
13.6.5.6	Search Services	465
13.6.5.7	Examples	465
13.6.5.8	REST List API Info	465

13.6.5.9	REST List API Read	470
13.6.5.10	REST List API Write	515
13.6.5.11	REST List API Delete	539
13.6.5.12	REST List API Errors.....	545
13.6.5.13	REST List API Tutorial.....	547
13.6.5.14	REST List API Troubleshooting and How To's	557
13.6.6	REST Object API.....	592
13.6.6.1	Overview Presentation	592
13.6.6.2	Examples	592
13.6.6.3	OpenAPI Support	592
13.6.6.4	REST Object API Read V1	595
13.6.6.5	REST Object API Read V2	632
13.6.6.6	REST Object API Write	674
13.6.7	REST Management API.....	703
13.6.7.1	REST Assortment API	703
13.6.7.2	REST Data Quality API.....	705
13.6.7.3	REST Environment API.....	721
13.6.7.4	REST Export API.....	729
13.6.7.5	REST File API.....	738
13.6.7.6	REST Import API	741
13.6.7.7	REST KPI API	747
13.6.7.8	REST Merge API	755
13.6.7.9	REST Revision API	769
13.6.7.10	REST System API	777
13.6.7.11	REST Task API.....	783
13.6.7.12	REST Workflow API.....	796
13.6.8	REST Enumeration API.....	821
13.6.8.1	All Enumerations	821
13.6.8.2	Meta Data and Enumeration Entries	822
13.6.8.3	Examples	825
13.6.9	REST Media API.....	827
13.6.9.1	Single Media Asset File.....	828
13.6.9.2	Preview of Media Asset File	828
13.6.9.3	Media Asset File Location	829
13.6.9.4	Upload single Media Asset File	829

13.6.9.5	Examples	830
13.6.9.6	REST List API for MediaAssetFile entity.....	833
13.6.9.7	REST Media Asset Category API.....	862
13.6.10	REST Meta API	870
13.6.10.1	All Root Entities	870
13.6.10.2	Metadata for a Root Entity.....	871
13.6.10.3	All Fields of a Root Entity	873
13.6.10.4	Metadata for a Sub Entity	874
13.6.10.5	Metadata for a Field	875
13.6.10.6	Metadata for a Qualifier	878
13.6.10.7	Examples	880
13.6.10.8	REST Meta Media API	885
13.6.11	REST Security API.....	901
13.6.11.1	List all Security ACL APIs.....	901
13.6.11.2	Read ACL object	902
13.6.11.3	Create ACL object.....	904
13.6.11.4	Examples	907
13.6.12	PUBLIC Server Health Status.....	912
13.6.12.1	Overview.....	912
13.7	Message Queue API	914
13.7.1	Overview.....	914
13.7.2	Queue Communication Overview	914
13.7.3	General Queue Configuration.....	917
13.7.3.1	Queue Purposes	917
13.7.4	Custom Queues.....	918
13.7.5	General Headers.....	920
13.7.5.1	Request Headers	920
13.7.5.2	Response Headers.....	922
13.7.6	Default Queues.....	924
13.7.6.1	BPM Queue	924
13.7.6.2	Batch API Queue.....	924
13.7.6.3	Service API Queue	924
13.7.6.4	ObjectAPI Queue	925
13.7.7	BPM Queue	925
13.7.7.1	Trigger Message Header	925

13.7.7.2	Message Body.....	926
13.7.8	Batch API Queue.....	926
13.7.8.1	Architecture Design.....	926
13.7.8.2	Data Quality.....	929
13.7.8.3	Merge	931
13.7.9	Service API Queue	934
13.7.9.1	Message Header	934
13.7.9.2	Error Response	935
13.7.9.3	Fully Supported APIs.....	936
13.7.10	ObjectAPI Queue	945
13.7.10.1	Request & Response Headers.....	945
13.7.10.2	Response Headers.....	946
13.7.10.3	Parameters	946
13.7.10.4	Create Operation.....	946
13.7.10.5	Read Operation	950
13.7.10.6	Update Operation	961
13.7.10.7	Delete Request	965
13.8	How to transfer entity including ACL objects to another PIM system	966
13.8.1	Use case	967
13.8.2	Flow	967
13.8.3	Detail REST calls.....	967
13.8.3.1	Find items with ACL reference.....	968
13.8.3.2	Find ACL objects by id.....	969
13.8.3.3	Create ACL object in target system	971
13.8.3.4	Create items in target system with references to created ACL objects.....	974
13.8.4	More example for assignment of entity with new ACL reference ids in target system	976
13.8.4.1	Assotment	976
13.8.4.2	Export template	978

On this and the following pages you will find a deep dive into the architecture of Product 360 as well as tutorials on various topics. You will learn how to access and modify nearly all the data which is stored, including product, article and structure group information. We will explain the generic business model approach and how the persistence layer works.

1 Integrate with Product 360

Informatica MDM - Product 360 has a powerful [REST Service API \(see page 428\)](#) which can be used to integrate PIM data and logic into your own applications. The complete Service API is an official API of Product 360 and part of the maintenance.

2 Customize Product 360

The API documentation is provided as javadoc web pages which are in total available in the SDK build.

3 API

3.1 Java API

The Product 360 API has three levels of "stability":

3.1.1 Internal

All classes/interfaces which are contained in a package which is marked as internal, as well as all db packages are to be treated as high risk internals. Using these classes is not recommended at all, functionality or even existence is not guaranteed even between hotfixes. Please note that internal packages are not part of this javadoc distribution.

3.1.2 Normal

All classes which are **not contained** in an internal package, but are not (yet) API. It's always a risk to use them, although not as much as for internal classes. Unheralded refactorings might be done by us in order to make them API in the future. However, we try to keep them stable as long as possible. Functional changes between hotfixes and service packs are very uncommon for those classes.

3.1.3 API

All classes, interfaces and methods which are marked as "API" will be held stable as long as possible. API classes are highlighted in this documentation.

3.1.4 Upgrade **path**:

API classes can be marked as deprecated a major or minor release, usually not in service packs or hotfixes. Deprecated classes will continue to work until the next **major** release. As usual in these cases we recommend to update your implementation as fast as possible Example: You use existing methods and classes which are marked as API in release 6.0. Then we define this api as deprecated for 6.5. The methods and classes will continue to work at least till the next major release which is 7.0.

3.1.5 Modification vs. Customizing

Product 360 customers should always be able to apply a service pack or a new release to the installation without breaking any product functionality or customizing added by the consulting partner team. **Therefore no change to any part of the Product 360 source code is allowed.** Additionally, every customizing in an installation **should not use or rely on any interface or Java class, that is not clearly marked and documented as API.** We know that sometimes there is no other possibility to do that, but in general, projects should try to avoid this. Anything that is not an API may change without any warning. Anything that is marked as API is documented and part of the product maintenance. If we do have an API change, the old interface will be available until we deprecate it in the next major release with an official statement.

3.2 Persistence Layer

The database layer, including JPA mappings and linked database plugins are not allowed to modify.

3.3 Domain Model / Repository

Adding own top entities which did not change the origin PIM entities is allowed. It's also allowed to add a Repository types section for that. Note that the 'links' to other PIM entities should modelled in your custom top entity types and not in the existing PIM entities. Our PIM Repository ist not marked generally as API. Please follow link to detailed description and especially what's allowed on 'Types' area and what's not.

API Request Process

Missing extension points, interfaces or API marked Java classes can be requested to be added to the API. This requires a detailed description why and how this extension point is needed, including an actual Customer Use-Case. After a positive review from Product Management and Development, this extension will be developed and shipped with an upcoming Service Pack. As an intermediate workaround it might be tolerable to use non-api classes for the special use case, but this has to be allowed by R&D.

Please contact your Global Customer Service representative and file an API request with them - they will help you through the process.

4 Software Development Kit

The SDK provides documentation on available API's, best practice development infrastructure, ready to use examples for customizing and integration as well as ready to use JUnit test projects



The SDK installation steps for PIM version 10.5.0.01.00 and higher.

4.1 SDK Package

The SDK zip package has the following structure

Folder	Description
eclipse	Third-party eclipse plugins and external development tools
example	Example projects (customizing template projects, testing examples, PDE build examples, export templates, web access, etc)
preferences	Eclipse IDE preferences
projects	Example Eclipse project with configuration files required to start HPM server and client in development environment
targets	Product 360 runtime, javadoc, development tools and frameworks

4.2 Prerequisites

4.2.1 Database

An available database server with the corresponding Product 360 schemas needs to be installed and configured to be accessible from the development machine. Refer to the installation manual for details

4.2.2 Java Development Kit

A Java Development Kit Version 17 in 64bit must be available on the machine to run the eclipse distribution. Due to licensing limitations the SDK package is delivered without a JDK. However, usually it's enough to just use the most recent 17 JDK available.

<https://www.azul.com/downloads/zulu-community/?version=java-17-lts&os=windows&architecture=x86-64-bit&package=jdk>

4.2.3 Eclipse

You need to have Eclipse as your IDE, others will not work. The minimum required version is the Eclipse IDE 4.25.0(2022-09).

In all cases, we recommend to use the "*Eclipse for RCP and RAP Developers*" edition.

Eclipse 2022-09

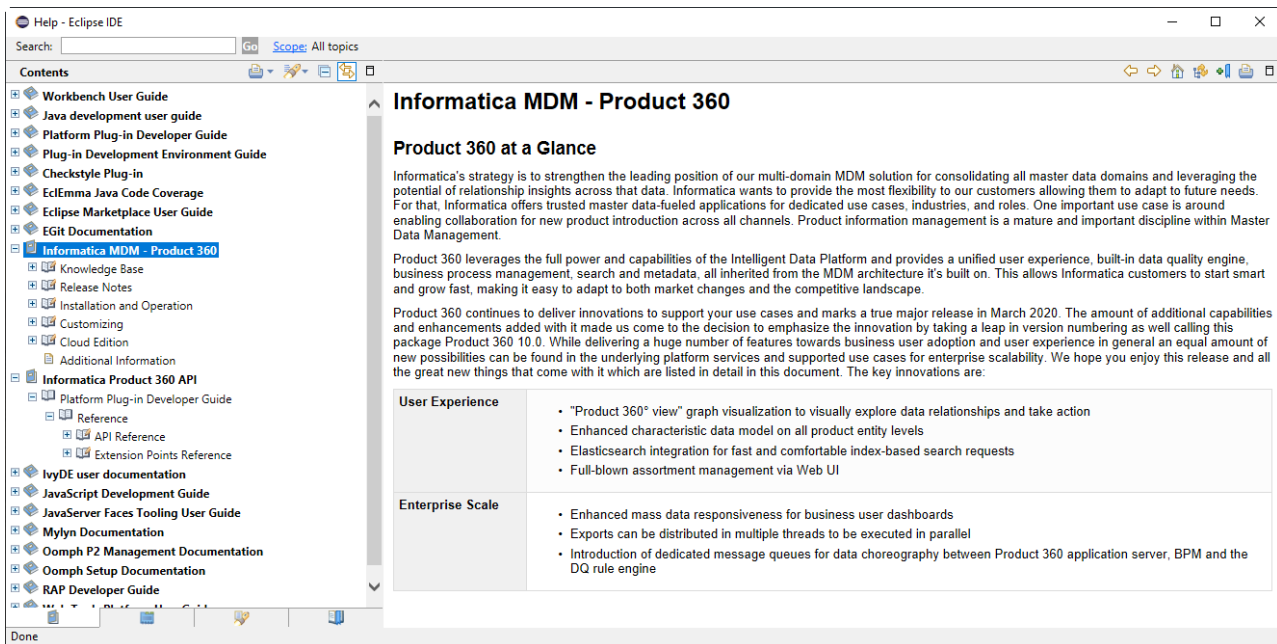
<https://www.eclipse.org/downloads/packages/release/2022-09/r/eclipse-ide-rcp-and-rap-developers>

4.3 Installation of the SDK Online Help

The SDK online help contains the javadoc for the API, the list of available extension points as well as the general knowledge base and technical development articles. After installation it is accessible within Eclipse via Help -> Help Contents.

- Unzip the SDK distribution to an arbitrary place on your hard drive. In this example we use, `C:\Informatica\PIM-SDK`.
- Copy all plugins from the `C:\Informatica\PIM-SDK\eclipse\sdkdoc\plugins` folder to the `dropins\plugins` folder of your eclipse installation (you may have to create the `plugins` subfolder before)
- Generate `com.heiler.sdk.help.wiki_<version>.jar` from knowledge base page and place that in `dropins\plugins` folder of your eclipse installation.
- Restart Eclipse

If you now open Help Contents inside of eclipse, you should see two sections named "Informatica Product 360" and "Informatica Product 360 API" as shown on this picture.



4.4 Installation of the SDK

- Run eclipse and choose an arbitrary workspace for the project you want to build.
For convenience the SDK has a built-in project structure which you might find useful. We use `C:\Informatica\PIM-SDK\projects\HENRI\workspace` as the development workspace location for our "HENRI" project.



A good advice is to use workspace paths as short as possible. Otherwise this may lead to Data Quality related problems during server start or during Data Quality execution that some plugins or other objects cannot be found.

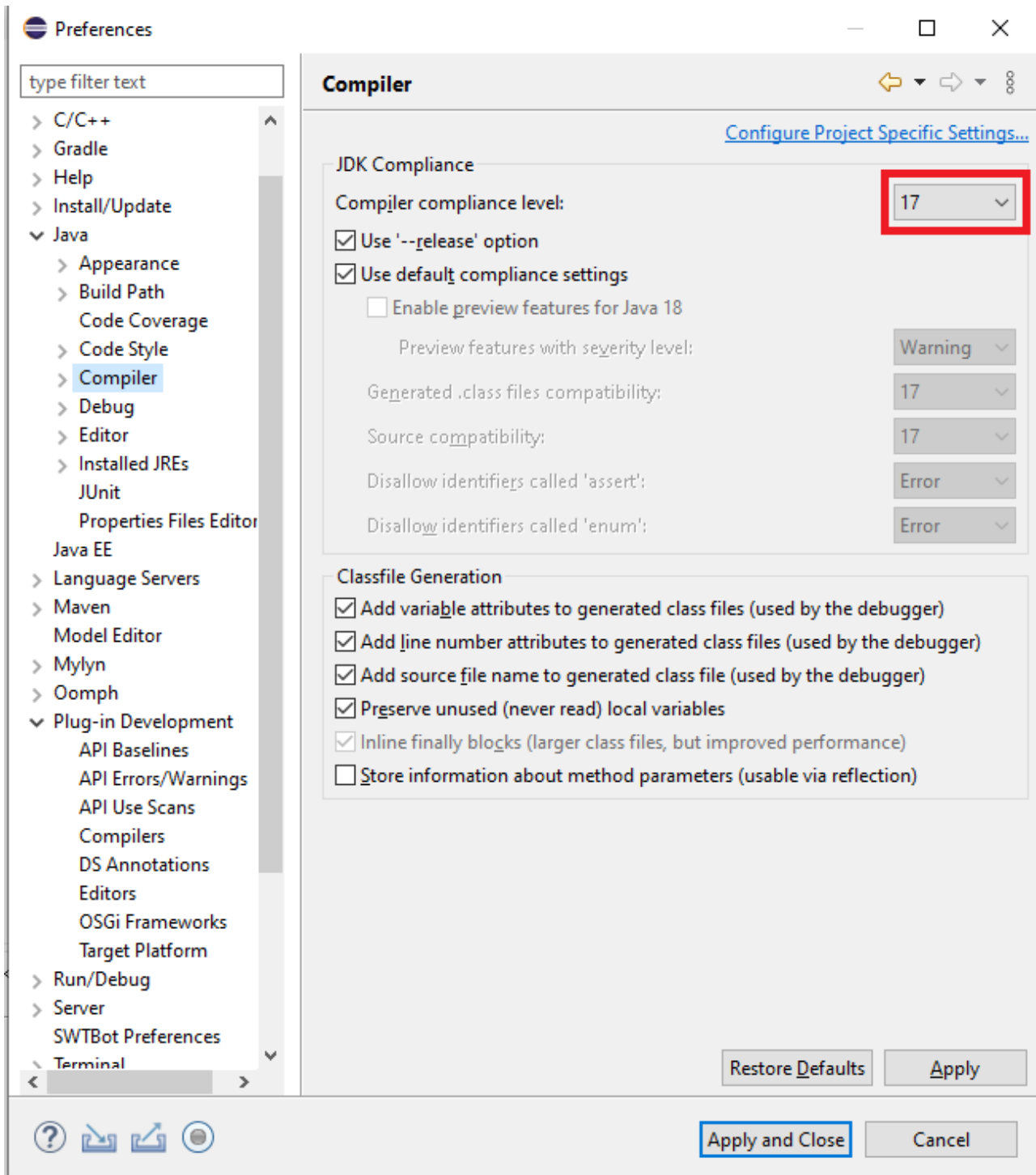
- (optional) Install 3rd party plugins from `C:\Informatica\PIM-SDK\eclipse\3rdparty`.
 - We recommend to install the Resource Bundle Editor plugin which makes it very easy to edit resource property files like the i18n files needed for multi-language support. This editor can be found on the Eclipse marketplace: <http://marketplace.eclipse.org/content/resourcebundle-editor>
- Go to **File > Import > Preferences** and import the `Eclipse_4.4_Preferences.epf` preferences into your workspace. This preference file is located in the `C:\Informatica\PIM-SDK\preferences` folder. Preferences contain Java formatter settings which satisfy the Informatica standard coding conventions.



We suggest to use the provided development preferences, but you can also use your own conventions. In any case, we strongly recommend that the whole team uses the same conventions unless that you will have lots of "merge conflicts" with your source code management system.

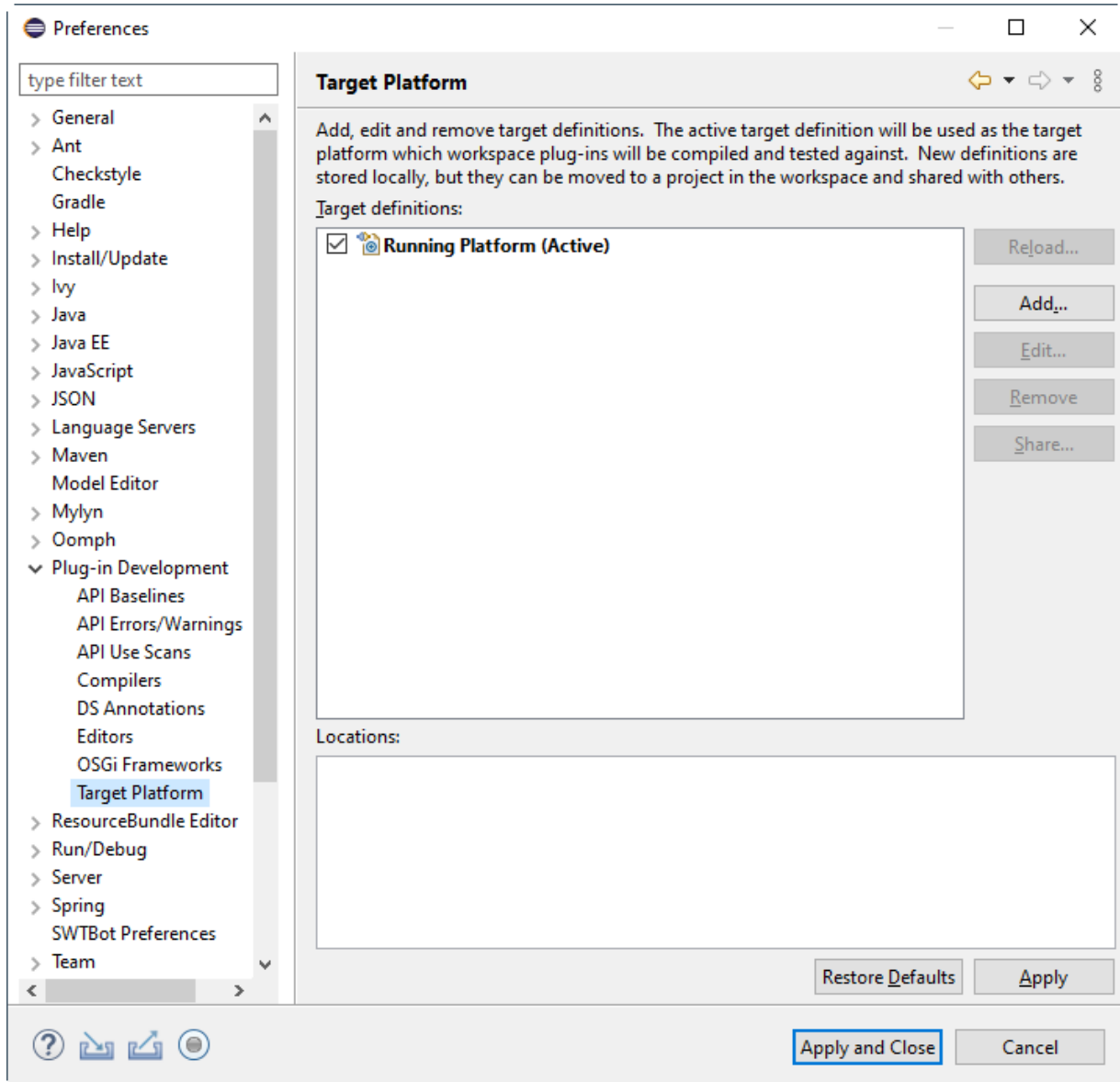
- Add the 17 JDK 64bit to the workspace, if not already done.
 - Open **Window > Preferences > Java > Installed JREs**

- Add the JDK 17 to the list of available JREs and set the JRE instance name, for example `jdk17`.
Make sure to use a JDK. By this means you have the Java source code available.
- Set the installed JRE to "default" (check the checkbox).
- Check the Java compiler settings. Open **Window > Preferences > Java > Compiler** and make sure that the default compliance level is 17.



4.4.1 Install target definition

- Open **Window > Preferences > Plug-in Development > Target Platform**.



- Create new target definition: **Add > Start with empty target definition**. Set name to "Product 360 Target 64Bit"
- Open environment tab and set Target Environment.
 - *Operating System: win32, Windowing System: win32, Architecture: x86_64.*

Edit Target Definition

Target Content
Edit the name, description, and plug-ins contained in a target.

Name:

Locations Content Environment Arguments Implicit Dependencies

Target Environment
Specify the target environment. If left blank, the default environment variables from the host (running) platform will be used.

Operating system:

Windowing System:

Architecture:

Locale:

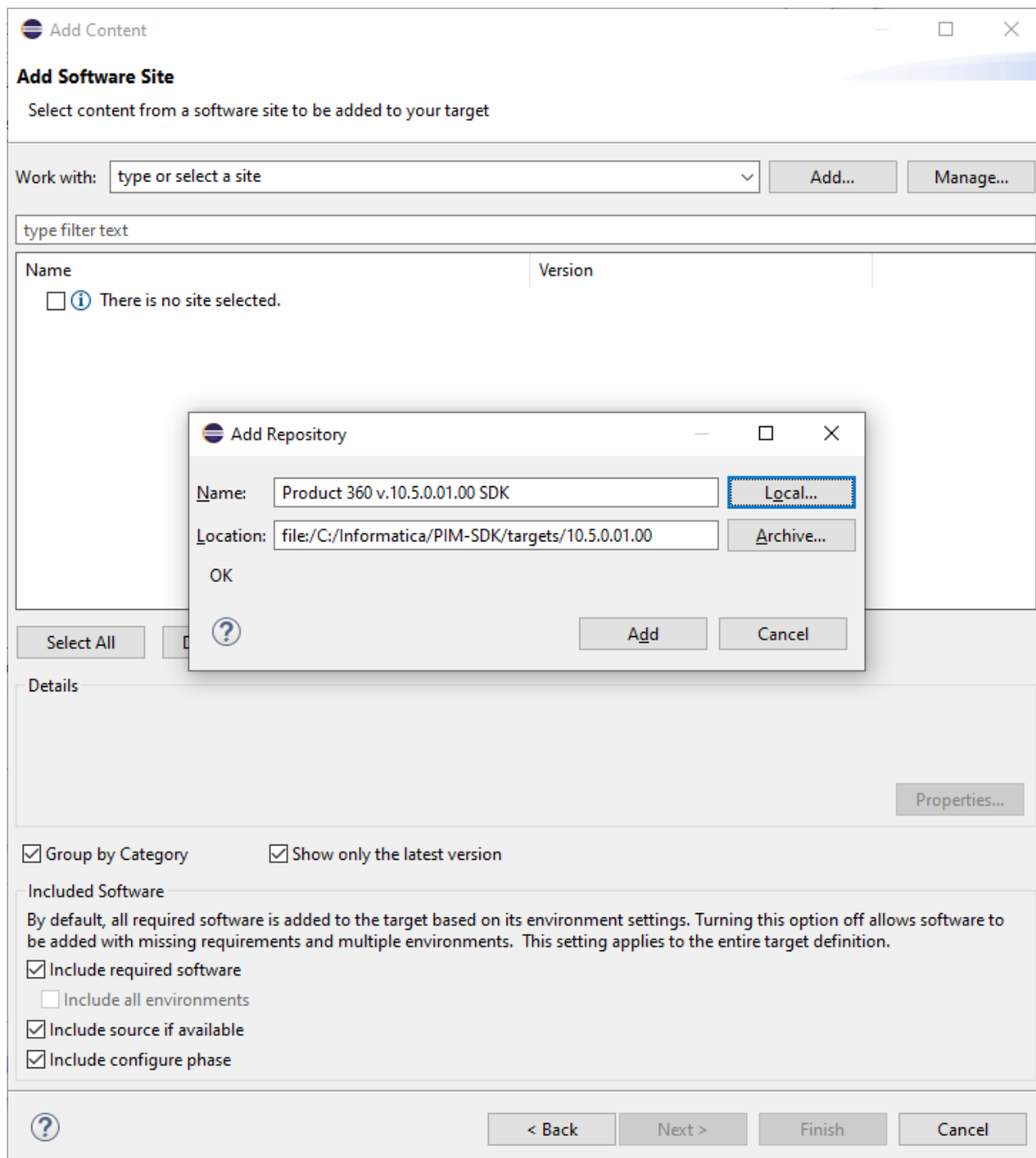
Java Runtime Environment
Specify the JRE or execution environment for this target. Selecting a named JRE will change the workspace default JRE setting.

☒ Default JRE

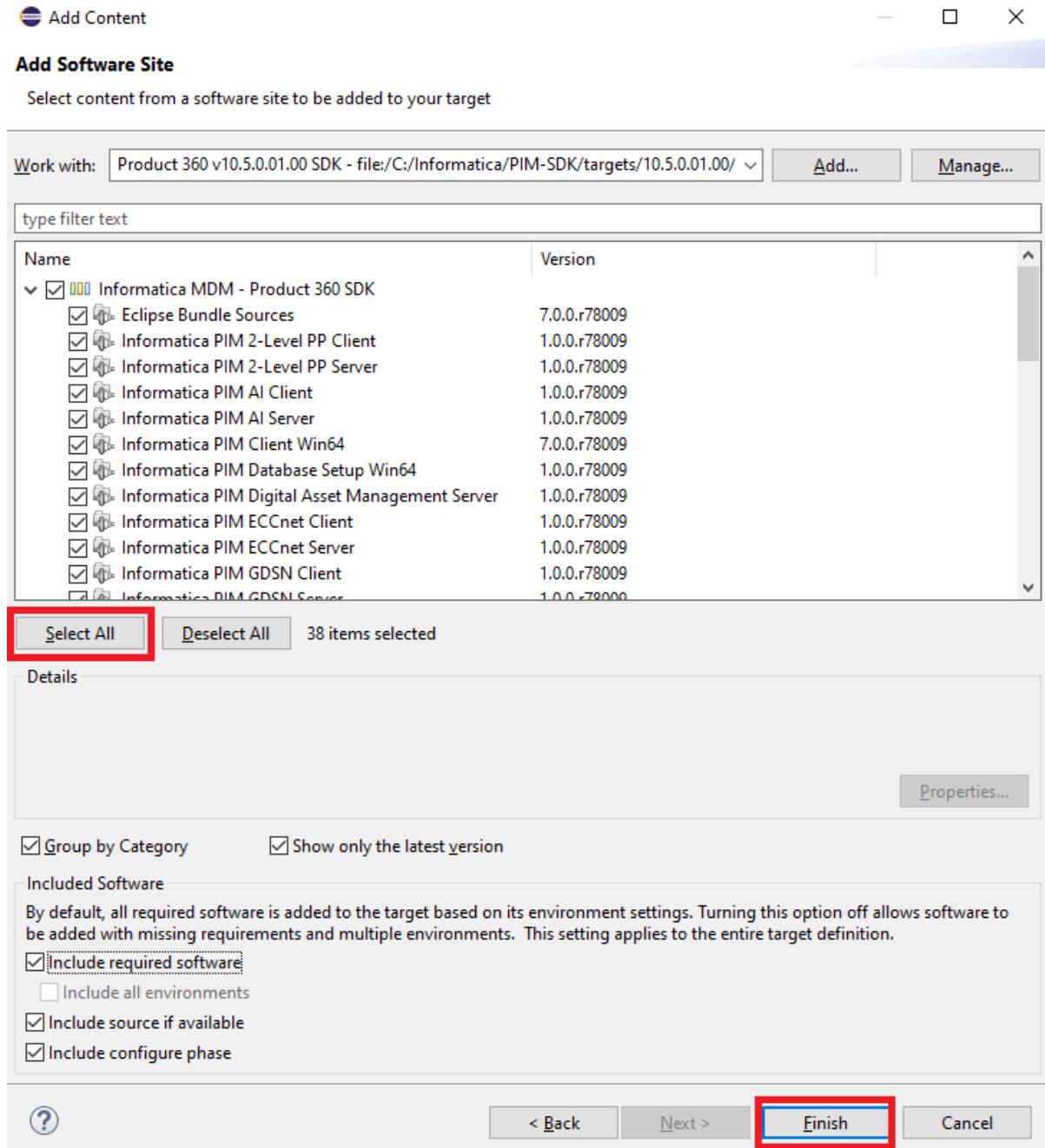
☐ JRE name:

☐ Execution Environment:

- Add new Software Site location: **Locations > Add > Software Site**
- Add new local software site: **Add > Local...** Select `C:\Informatica\PIM-SDK\targets\10.5.0.01.00\` folder and click **ok**.



- Now you will see a list of available features in the SDK including all 3rd Party and Eclipse features:



4.4.1.1 SDK Feature

SDK Feature	Description
Eclipse Bundle Source	Sources of important Eclipse Bundles (e.g. org.eclipse.runtime.core)
Informatica PIM Repository Editor Win64	Repository editor (Windows 64bit platform)
Informatica PIM Database Setup Win64	Database Setup (Windows 64bit platform)
Informatica PIM Server Runtime	Server runtime
Informatica PIM Server Win64 Support	Windows 64 platform server specific bundles (GraphicsMagick, eclipse launcher, etc)
Informatica PIM Rich Client	Runtime of Product 360 Desktop Client
Informatica PIM Client Win64	Windows 64 platform Desktop client specific bundles
Informatica PIM Web Client	Web Access components that are needed when starting the Product 360 Desktop Client (Web Action Rights)
Informatica PIM Web Server	Web Access runtime for the Product 360 Server.
Informatica PIM 2-Level PP Client	Add-On to Desktop Client for 2-tier Product Paradigm.
Informatica PIM 2-Level PP Server	Add-On to Product 360 Server for 2-tier Product Paradigm.
Informatica PIM Web 2-Level PP Server	Web Access runtime for the Product 360 Server for 2-tier Product Paradigm.

SDK Feature	Description
Informatica PIM Variant Client	Add-On to Desktop Client for 3-tier Product Paradigm.
Informatica PIM Variant Server	Add-On to Product 360 Server for 3-tier Product Paradigm.
Informatica PIM Web Variant Server	Web Access runtime for the Product 360 Server for 3-tier Product Paradigm.
Informatica PIM GDSN Client	Add-On to Desktop Client for the GDSN Accelerator
Informatica PIM GDSN Server	Add-On to Product 360 Server for the GDSN Accelerator
Informatica PIM Web GDSN	Web Access runtime for the GDSN Accelerator
Informatica PIM Digital Asset Management Server	This feature contains the plugin(s) that are needed to use Digital Asset Management system
Informatica PIM Supplier Exchange REST Interface	This feature contains the plugin(s) for the use of PIM Supplier Portal system
Informatica PIM Supplier Exchange Client	This feature contains the plugin(s) for the use of PIM Supplier Portal system specific actions in the rich client
Informatica PIM Websphere Client	This feature contains the plugin(s) for the use of IBM Websphere specific actions in the Product 360 Desktop Client
Informatica PIM Websphere Client I18N	This feature contains the language plugin(s) for the use of IBM Websphere specific actions in the Product 360 Desktop Client

SDK Feature	Description
Informatica PIM Websphere Server	This feature contains the plugin(s) that are needed to use IBM Websphere integration in the Product 360 Server
Informatica PIM Websphere Server I18N	This feature contains the language plugin(s) that are needed to use IBM Websphere integration in the Product 360 Server
Informatica PIM SDK Javadoc	Contains the javadoc for the server and client binaries.
Informatica PIM Web SDK Javadoc	Contains the javadoc for the Web Access binaries.
Informatica PIM Testing Framework	Testing framework including junit, mockito and eclipse testing framework
Informatica PIM Tooling Client/Server	Runtime Tools (Command Inspector (see page 416), Eclipse Plugin Spy (see page 417) and others)
Informatica PIM ECCnet Client	Add-On to Desktop Client for the ECCnet Accelerator. (Not a standard feature)
Informatica PIM ECCnet Server	Add-On to Product 360 Server for the ECCnet Accelerator. (Not a standard feature)
Informatica PIM Web ECCnet	Web Access runtime for the ECCnet Accelerator (Not a standard feature)
Informatica PIM AI Client	Add-On to Desktop Client for the CLAIRE Accelerator. (Not a standard feature)
Informatica PIM AI Server	Add-On to Product 360 Server for the CLAIRE Accelerator. (Not a standard feature)

SDK Feature	Description
Informatica PIM Web AI	Web Access runtime for the CLAIRE Accelerator (Not a standard feature)

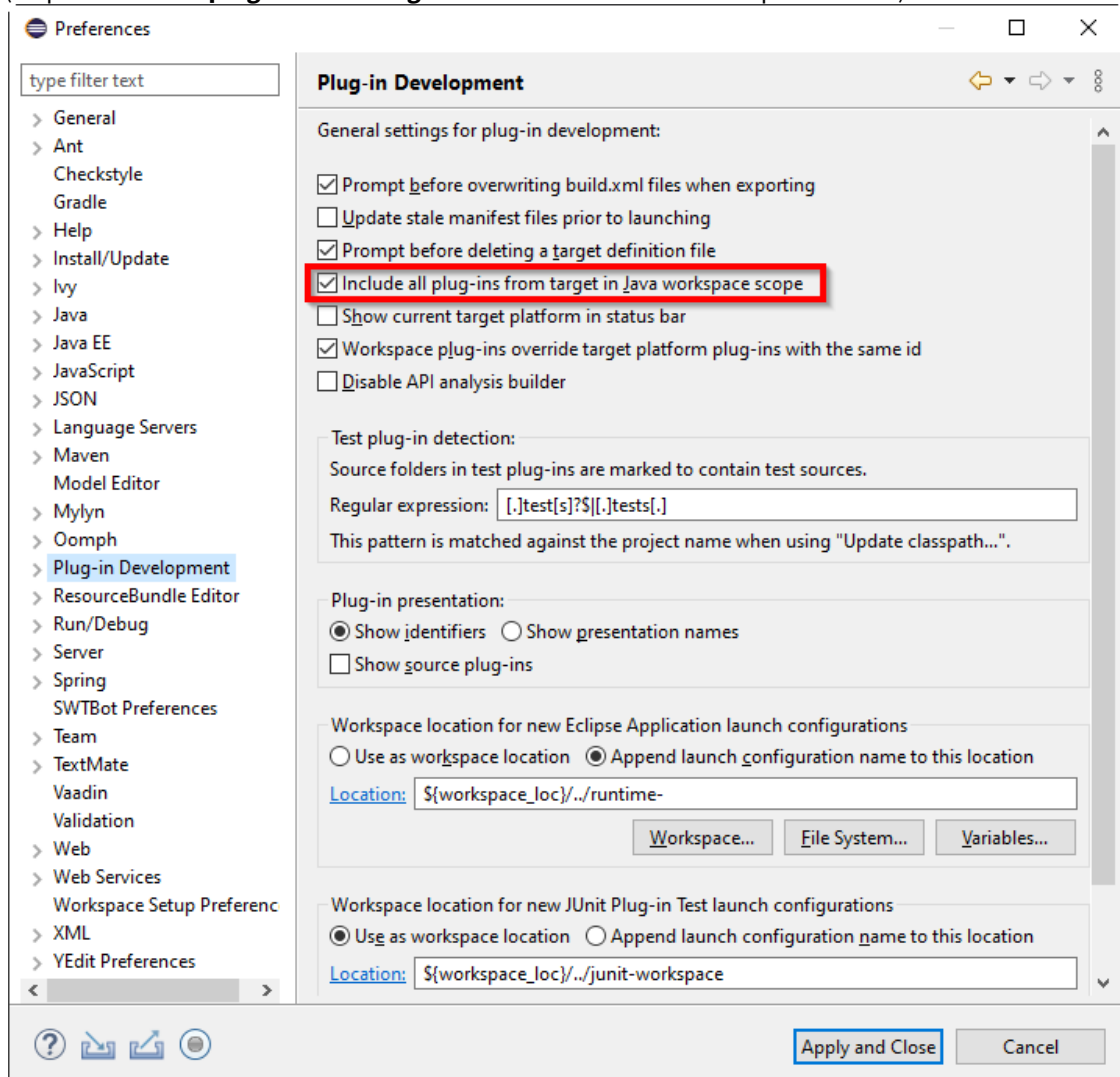
Select at least

- *'Informatica PIM Rich Client'*,
- *'Informatica PIM Client Win64'*,
- *'Informatica PIM Repository Editor Win64'*,
- *'Informatica PIM Server Runtime'*,
- *Informatica PIM Server Win64 Support'*.

For full-fledged development you will need all features. It is recommended to have all the features selected.

- Click **Select All** to select all the features.
- Click **Finish** and select 'Product 360 Target 64Bit' as default target platform.
- Add target platform bundles to java search. **Window > Show View > Plug-in development > Plug-ins**.
 - In the Plug-ins view toolbar click on '**Add All Plug-ins to Java Search**'.

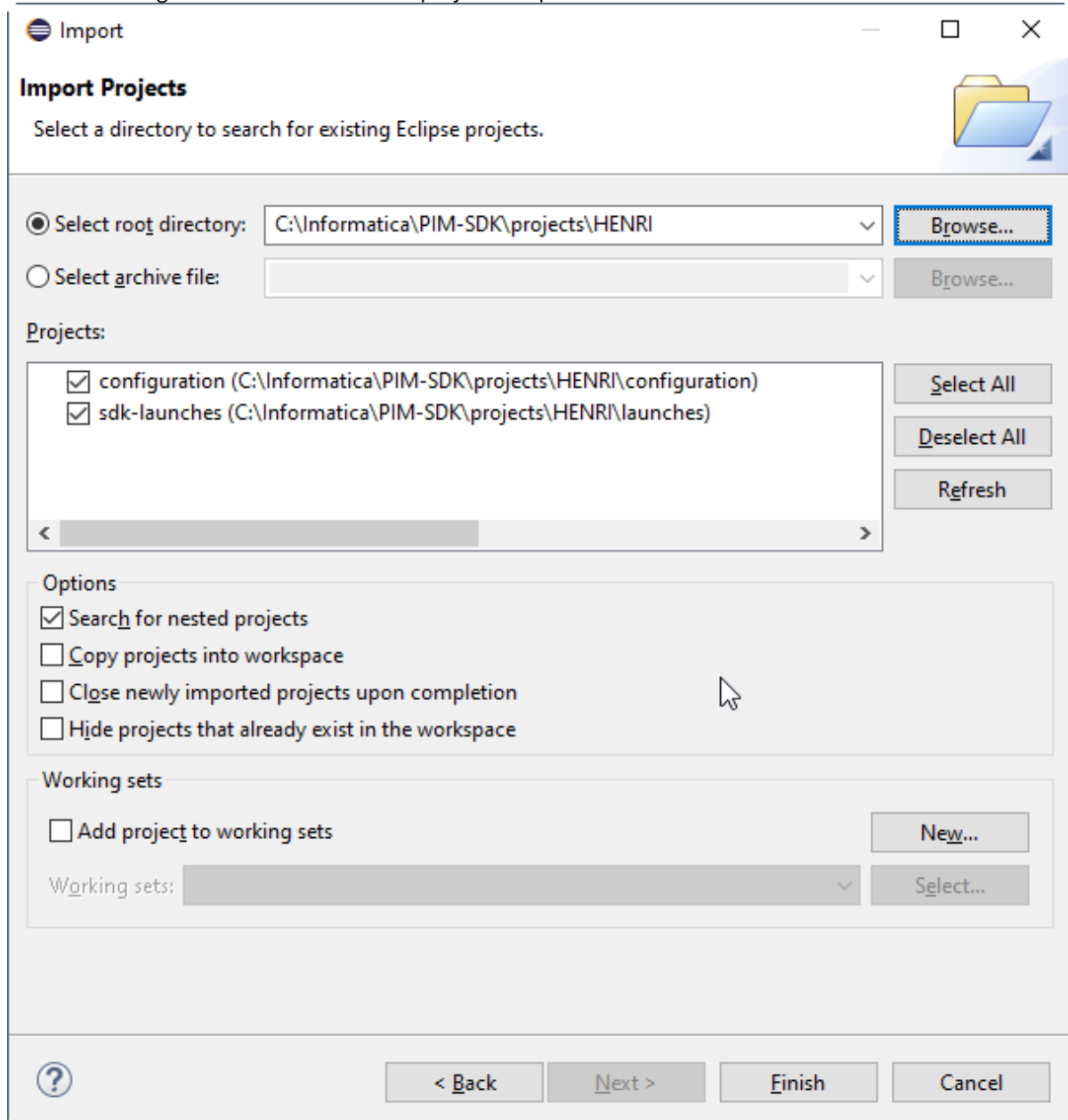
- You can also check the preference **'Include all plug-ins from target in Java workspace scope'** (resp. **'Include all plug-ins from target in Java search'** in older Eclipse versions)



4.4.2 Install launches and configuration

- Import the launch and configuration projects into your workspace.
- Open **File > Import > Existing Projects into Workspace**.
- Select the `C:\Informatica\PIM-SDK\projects\HENRI` directory for the root directory.

- Check the configuration and the launches project and press **finish**



- Configure the configuration settings of your project. In the configuration project in your workspace you will find two sub folders, client and server. The client folder holds all configuration files for the client start, usually you do not need to modify anything here. The server folder contains all required configuration files for the server start. At least the `server.properties` file needs to be configured. In the server folder you will find several `server.properties` templates (one per supported database). There are a few properties that must be set in order to start the server: See the corresponding section in the PIM Core Installation Manual for details on this configuration.

Example: server properties

```

filestorage.dir.shared      = C:/Informatica/PIM-SDK/projects/HENRI/runtime
license.customer.file.local = ${filestorage.dir.shared}/license/
<YourLicenseFile.license>
license.customer.key        = <CustomerKey>
db.default.server           = localhost
db.default.dir              = C:/Informatica/PIM-SDK/projects/HENRI/runtime/database
db.default.schema.suffix    = _SDK105

```



Before setting up your SDK environment you can set up a standalone Product 360 runtime system for your project, you will need the development database anyway. After this you can use the configuration files from this working system and copy them to the configuration/client and configuration/server directory.



For setting up your SDK environment you need a "Informatica Product 360 v10.5" license. Consulting and external implementation partners can request this license from the support department.

- Copy your license file to the directory `C:/Informatica/PIM-SDK/projects/HENRI/runtime/license`

4.4.2.1 Launch Product 360 Server

- Go to **Run > Run Configurations > Eclipse Application > Informatica MDM - Product 360 Server**.
- (Optional) To run P360 with Informatica Media Manager, select '*com.heiler.ppm.feature.opasg.server*' as feature to launch with, too.
- Press run.

4.4.2.2 Launch Product 360 Client

- Go to **Run > Run Configurations > Eclipse Application > Informatica MDM - Product 360 Desktop Client**.
- Press run.
- Enter default credentials in the authentication dialog (Username: `Administrator`, Password: `Administrator`).
- Launch Web Access
 - Open your browser and enter the following url: `http://localhost:1512/pim` (replace *localhost* with the *dest.host* from *server.properties* if configured)
- Congratulations, you're ready to start developing with Informatica MDM - Product 360!

4.4.2.3 Launch Product 360 Repository Manager

- Go to **Run > Run Configurations > Eclipse Application > Informatica MDM - Product 360 Repository Manager**.
- Press run.

4.4.2.4 Launch PIM Database Setup

- Go to **Run > Run Configurations > Eclipse Application > Informatica MDM - Product 360 Database Setup**.
- Press run.
- For further steps to work with Database setup see: Server Database

4.4.3 Install examples

- Open **File > Import > Existing Projects into Workspace**.
- Select the `C:\Informatica\PIM-SDK\examples` directory for the root directory.
- Check all example projects, but the `customizing.build` and press **finish**

4.4.3.1 Examples at a glance

Project	Description
com.heiler.ppm.custom.rest	Examples for own rest based services. Contains a running example for a rest service which is able to return some item data based on EAN numbers
com.heiler.ppm.customizing.activities	Examples how to disable UI contributions (like views, menu items etc.) using "Eclipse activities"
com.heiler.ppm.customizing.article.ui.contributionclass	Example how to replace a standard view (like "Items #1") with a custom view using the extension point <code>contributionClassProviders</code>
com.heiler.ppm.customizing.article.ui.searchView	Example how to customize the standard "Search view" using the extension point <code>searchViewAdditions</code>
com.heiler.ppm.customizing.ckeditor	Examples how to customize the standard RichText-Editor using extension points <code>ckeditorConfiguration</code> , <code>richTextValidators</code> and <code>richTextMarkupConversionExtensions</code>
om.heiler.ppm.customizing.core	Core functionality of your own customizing, is empty by default. Use it as a template for your own customizing code

Project	Description
com.heiler.ppm.customizing.core_test	Test bundle for unit and integration tests. Contains an example for a unit test (<code>CustomizingUnitTest</code>) as well as for an integration test (<code>CustomizingIntegrationTest</code>). Additionally to that it contains working launch configurations for the given test suites. One for integration and one for unit tests. Use this bundle to implement your own tests for your customizing business code.
com.heiler.ppm.customizing.export.core	Examples for implementation of own export data providers, data types and export functions
com.heiler.ppm.customizing.export.core.test	Examples for writing tests for own export functionality
com.heiler.ppm.customizing.server	Server functionality of your own customizing, is empty by default. Use it as a template for your own customizing code
com.heiler.ppm.customizing.spelling.ui	Example how to use the <code>SpellCheckingAPI</code> to spellcheck the whole catalog in a mass-operation
com.heiler.ppm.customizing.ui	UI functionality of your own customizing, is empty by default. Use it as a template for your own customizing code
com.heiler.ppm.feature.customizing.*	Features for the client, server and tests (<code>junit</code>), used for the launch configurations and automatic builds. Add your own customizing plugins to these features
com.heiler.ppm.web.custom.theme	Sample project that demonstrates how to contribute a custom theme for WebAccess. Put your additional style definitions into the styles.css file.
com.heiler.ppm.web.custom.app	Sample project with some basic programmatic extensions of the PIM Web UI.

Project	Description
com.heiler.ppm.web.vaadin.sample	Sample web project that shows how to integrate a all new web interface by leveraging the PIM stack and the Vaadin framework.
customizing.build	Ready to use ant targets to build your customizing bundles and tests in an integrated and automated build system. Based on Eclipse p2 build infrastructure.

5 Application Layer

Based on the PIM platform we introduced our Product Information Management aimed at B2B dealers and manufacturers of technical products. This complete solution is designed for medium and large suppliers and brings together the entire process chain from data acquisition to information management to automated publication and distribution of printed and electronic catalog data in one standard software solution.

User productivity when processing large amounts of data is critical for the success of a PIM solution. Global providers are increasingly recognizing the positive effects that an automated PIM solution can have on sales and costs. In a climate of fierce international competition, companies can only achieve their goals with structured, standardized processes for producing and distributing their product information.

This section is intended primarily for readers who have not used a product data management system before, or have only dealt with some aspects of product data management. Product Information Management refers to media-neutral management of product and business information for various publication channels. Three major scenarios are distinguished:

A company is a manufacturer and sells its products to other companies ("manufacturer scenario").

A typical feature of the manufacturer scenario is that a relatively small number of products normally has to be handled. The quality of the product data can vary from "rather scant" to "perfect". In the manufacturer scenario, the preparation of product data is often left to specialist firms, since it is not generally a part of the company's core processes.

A company buys products from other manufacturers to use or consume themselves ("purchasing scenario" or "e-procurement scenario").

The e-procurement scenario primarily involves optimization of the purchasing processes (key issues here are "streamlining the supplier structure" or "optimizing the approval workflow") and is less concerned with perfect representation of product data. Product data management systems must provide supporting functions. Software products in the e-procurement scenario are therefore often called "content management systems" because the objective of product data management is to prepare available items for use in the e-procurement system.

A company buys products from other manufacturers to sell them on ("dealer scenario").

The dealer scenario is the most complex and – from the perspective of a product data management system – most demanding scenario, as the handling of product data is one of the company's core processes. The objectives in the dealer scenario are to provide simpler and faster read and write access to data volumes that in some cases may be enormous, as well as more flexible interaction with interfaces for importing and exporting data.

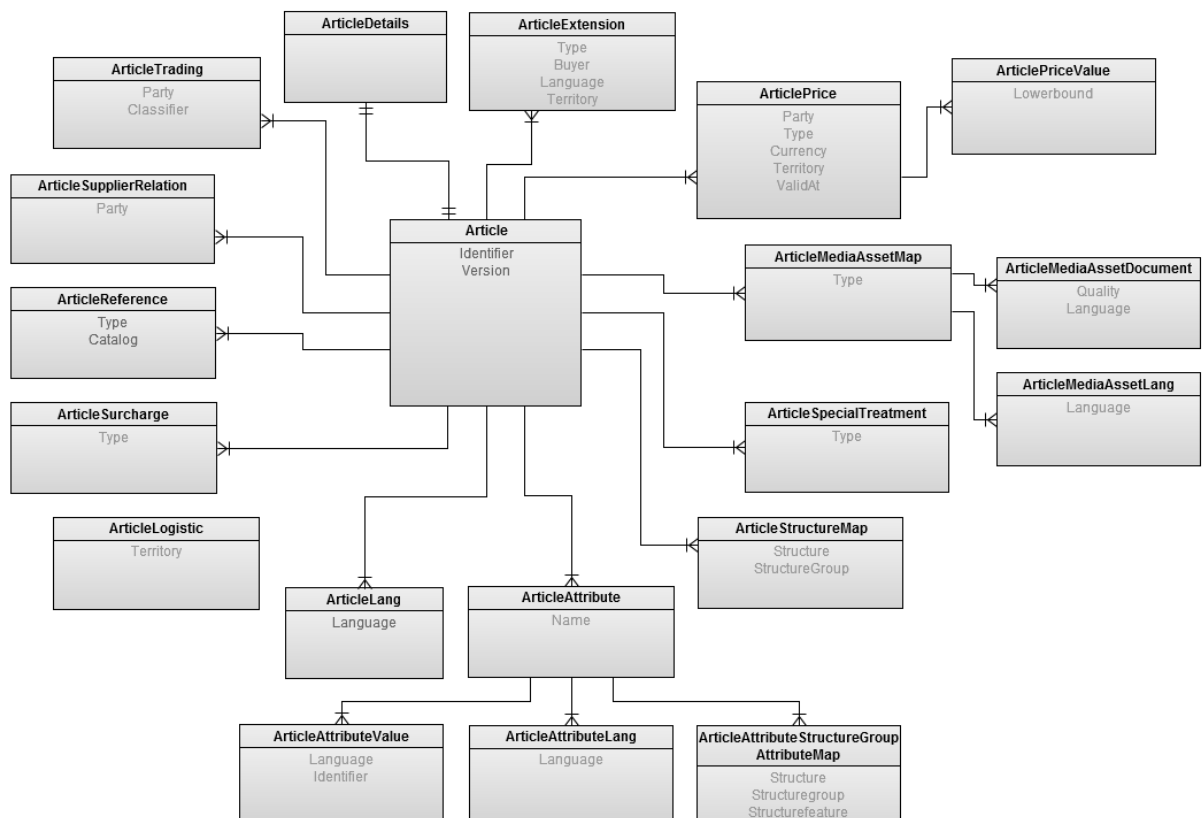
5.1 Article

The entity “Article” is representative for the entities “Variant” and “Product” as well. These entities are all based on the same type and physical data model.

Which information is finally provided and maintained on which level is decided on a more logical business driven level.

The figure shows a more detailed but still high level view on the entity “Article” and its sub-entities.

PIM “Article” Model: Product | Variant | Item



Copyright © Heiler Software AG, 2012

Entity	Sub-entity	Sub-entity description
Article	ArticleDetails	Detail information of the Article. Non-language dependent. E.g. Manufacturer name, manufacturer item id, GTIN, etc.
	ArticleTrading	Trading information like order units, content unit, minimum order quantity, or lead time, etc.
	ArticleSupplierRelation	Relationship to one or more suppliers providing the Article including specific information valid in the context of this relationship only (e.g. the supplier's item id)
	ArticleReference	Relationship between Articles. Typically merchandising relations of the "cross-selling", "up-selling", "accessories", etc.
	ArticleSurcharges	Surcharges of different types used in specific areas of products of some industries.
	ArticleLogistic	Logistic information of Articles like weights, measures, etc. Might be territory dependent.
	ArticleLang	All language dependent text information like names, descriptions and long descriptions.
	ArticleAttribute	Free list of attributes used to provide product type specific information like technical attributes. A list of key-value pairs where the attribute's name as well as the value are language dependent. An attribute may have additional information like a unit.

Entity	Sub-entity	Sub-entity description
	ArticleStructureMap	The assignment of the Article to a certain structure system. Structure systems in PIM are hierarchies representing taxonomies or other classifications of the products information. The assignment points to a certain structure group (a node of leaf in the hierarchy, a class or category). An Article might be categorized in multiple structure systems.
	ArticleSpecialTreatment	Special treatment flags for e.g. hazardous materials used in specific areas of products of some industries.
	ArticleMediaAssetMap	The assignment of digital assets (mainly images and text documents, but also videos, audios). Medias are differentiated by a type the documents assigned represent the different formats or qualities the media asset is available in. For images these are the different derivatives of the images used for different output channels.
	ArticlePrice	Price information of the Article. Prices differ in price type, currency, territory of validity, a period of time of validity and might be buyer-specific. Scaled prices dependent on the buying volume are supported.
	ArticleExtension	Flexible data entity where additional information can be provided, differentiated by a type. Optionally the information might be language dependent, buyer-specific and/or territory-specific.

5.1.1 Article Structure Map

5.1.1.1 Development Examples

How to create an Article Structure Map entry on an Article Type based entity.

```

private void mapArticleToStructureGroup( ArticleProxy article, String
structureGroupIdentifier,
                                RevisionToken revisionToken,
AccumulatedFeedbackProcessor feedbackProcessor )
    throws CoreException
{
    RepositoryService repositoryService = RepositoryComponent.getRepositoryService();
    EntityType articleEntityType = article.getEntityType();
    EntityType articleStructureMapEntityType =
repositoryService.getEntityTypeByIdentifier( "ArticleStructureMapType" ); //$NON-
NLS-1$
    Entity articleEntity = article.getEntity();
    Entity articleStructureMapEntity = repositoryService.getEntityByIdentifier(
"ArticleStructureMap" ); //$NON-NLS-1$

    LoadHint articleWithASMLoadHint = new
LoadHintBuilder( articleEntityType ).add( articleStructureMapEntityType,

false )
                                .build(
);
    EntityDetailModel articleDetailModel =
article.getDetailModel( articleWithASMLoadHint, true, revisionToken );

    articleDetailModel.acquireWrite( null );
    try
    {
        EDataObject articleDataObject = articleDetailModel.getDataObject();

        EntityPath articleStructureMapEntityPath = new
EntityPath( articleStructureMapEntityType );
        articleStructureMapEntityPath.setLogicalKeyValue(
"ArticleStructureMapType.LK.StructureId",
structureGroup.getContainer().getId() ); //$NON-NLS-1$
        FieldPath manualMapFieldPath = new FieldPath( articleStructureMapEntityPath,
"ArticleStructureMapType.ManualMap" ); //$NON-NLS-1$

        List< String > manualMapValue = new ArrayList<>();
        manualMapValue.add( structureGroupIdentifier );

        CommandContextParam commandContextParam = new
CommandContextParam( articleEntity, articleDetailModel );
        commandContextParam.setFeedbackProcessor( feedbackProcessor );
        CommandContext commandContext =
CommandContextFactory.createContext( commandContextParam );
        PutCommand putCommand = ( PutCommand )
commandContext.getCommand( PutCommand.TYPE );
        putCommand.put( commandContext, articleDataObject, articleDataObject,
articleStructureMapEntity,
                manualMapFieldPath, manualMapValue, true );
    }
}

```



```

    if ( commandContext.isCanceled() )
    {
        // do error handling, e.g. get the errors through
        feedbackProcessor.getFeedbacks()
    }
    else
    {
        articleDetailModel.save( null );
    }
}
catch ( CoreException ce )
{
    // do error handling, e.g. log or rethrow exception
}
finally
{
    articleDetailModel.releaseWrite();
}
}

```

5.2 Attributes

5.2.1 Attribute value datatype rules regarding decimal values

Datatype decimal:

- valid decimal places are orientating on
 - Scale-setting of Repository
 - physical database field size
- Entered decimal '0' digits are not persisted but added in the user interface by current scale settings of repository

Floatingpoint:

- valid decimal places are orientating on
- physical database field size
- Entered decimal '0' digits are not persisted in the database and are not displayed in the user interface (1.0000 will be displayed as 1)

Entering decimal or floatingpoint values

Currently if you are entering 15.0 this will be changed to 150 when a german locale is given. This is an issue

Solution: A validation error is thrown if the value can not be converted unambiguously in a number.

1.500 -> 1.500 = is correct

1.50 -> Error, no valid number

Check if grouping character is on correct position. The conversion algorithm is valid for both, decimal and floating point values.

5.3 Item/Variant/Product parent references

5.3.1 Background information

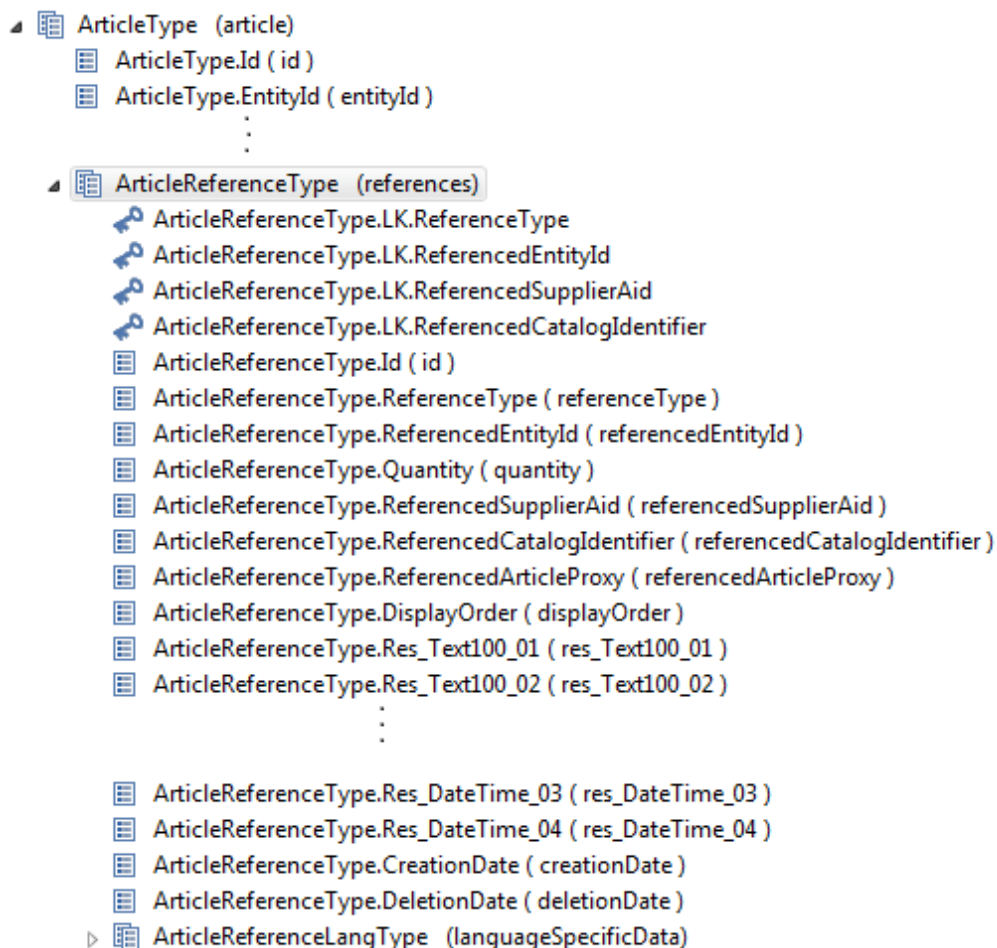
For an introduction to items variants and products see:

Knowledge Base > General Information > Product Paradigm - a understanding of products and their items
























Knowledge Base > General Information > References between products and items















5.3.2 Datamodel - Repository

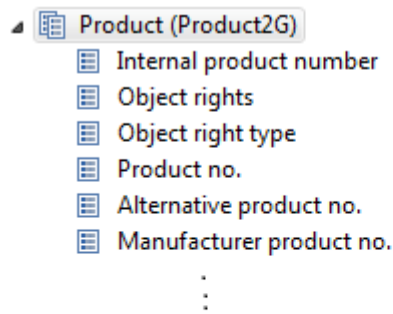
5.3.2.1 Types area



5.3.2.2 Custom area

- ▲  Items (Article)
 -  Internal item number
 -  Object rights
 - ⋮
- ▲  Higher-level product (ProductReference)
 -  Reference type
 -  Referenced object type
 -  Referenced product
 -  Referenced catalog
 -  Product reference type
 -  Referenced object type
 -  Referenced product
 -  Referenced catalog
 -  Referenced product number
 - ◆ Permissions com.heiler.ppm.article.core.permission.ArticleEdit
- ▲  Higher-level variant (VariantReference)
 -  Reference type
 -  Referenced object type
 -  Referenced variant
 -  Referenced catalog
 -  Reference type
 -  Referenced object type
 -  Referenced variant
 -  Referenced catalog
 -  Referenced object number
 - ◆ Permissions com.heiler.ppm.article.core.permission.ArticleEdit

- ▲  Variant (Variant)
 -  Internal variant number
 -  Object rights
 - ⋮
- ▲  Higher-level product (SuperordinateProductReference)
 -  Reference type
 -  Referenced object type
 -  Referenced product
 -  Referenced catalog
 -  Product reference type
 -  Referenced object type
 -  Referenced product
 -  Referenced catalog
 -  Referenced product number
 -  Permissions com.heiler.ppm.article.core.permission.ArticleEdit



General

In the hierarchy the product is at the top level, so it can't have a parent and has no subentity to store the reference. In 3 PPD the variant is on the level beneath products and according to that it has a subentity to store the reference to its parent product. At the lowest level there is the item. In 3PPD its parent is a variant and this parent reference is stored in the subentity Higher-level variant. In 2PPD the parent of an item is the product. This reference is stored in the subentity Higher-level product. In 2PPD mode the entity variant and the subentity Higher-level variant will be removed from the runtime repository whereas in 3PPD the Higher-level product item subentity remains in the repository.

Logical Keys

As you can see, in all subentities are the following four logical keys:

- Reference type: Defines the relationship of this reference. In the parent context only the following three numbers may be logical key value
 - 100 - Parent relation to a product of an item
 - 110 - Parent relation to a product of a variant
 - 120 - Parent relation to a variant of an item
- Referenced object type: entityId of the parent object (product 1100; variant 1200)

- Referenced variant/product SupplierAid
- Referenced catalog: MASTER

All logical keys except the ReferencedSupplierAid are not editable and preset with a fix value (depending of the entity). The parent object can change but there is business logic making sure that each object has one parent at most. This means there is none or exactly one data set for all of the subentities discussed here.

5.3.2.3 Future Plans

We would like to simplify the business model for parent relationships. Instead of different subentities we have a single field containing the parent proxy in mind. If the changes took place, the way to implement what is described on this page is to use the RelationHandler. However, today we don't know where in the roadmap this feature might fit.

5.3.3 How to resolve all objects with a parent relationship to one of the objects in a given list

5.3.3.1 2 PPD: Resolve all assigned items for a list of products

```
EntityItemList entityItemList = new ArticlesOfProductsList( productProxies );
EntityItem[] entityItems = entityItemList.toEntityItems( new NullProgressMonitor() );
```

Revision: current revision context

Datasource: Master catalog

5.3.3.2 3 PPD: Resolve all assigned variants for a list of products

```
EntityItemList entityItemList = new VariantsOfProductsList( productProxies );
EntityItem[] entityItems = entityItemList.toEntityItems( new
NullProgressMonitor() );
```

Revision: current revision context

Datasource: Master catalog

5.3.3.3 3 PPD: Resolve all assigned items for a list of variants

```
EntityItemList entityItemList = new ArticlesOfVariantsList( variantProxies );
EntityItem[] entityItems = entityItemList.toEntityItems( new
NullProgressMonitor() );
```

Revision: current revision context

Datasource: Master catalog

5.3.4 How to resolve all parent objects for a list of objects

5.3.4.1 2 PPD: Resolve all parent products for a list of items

```
RepositoryService repositoryService = RepositoryComponent.getRepositoryService();
Entity productEntity =
repositoryService.getEntityByIdentifier( ProductCoreConst.ENTITY_PRODUCT );

CatalogProxy currentCatalog = CatalogContext.getInstance().getCatalogProxy();

ReportListModelQuery query = new ProductsByArticlesQuery( articleProxies,
currentCatalog );
EntityItemList productEntityItemList = query.toEntityItemList( productEntity );
EntityItem[] products = productEntityItemList.toEntityItems( aProgressMonitor );
```

Revision: HEAD



Note that ListModelQuery is deprecated and not part of the API. It's better to use an EntityReport. Unfortunately there is no EntityReport with this functionality ready to be used. However, this might change in the future!

5.3.4.2 3 PPD: Resolve all parent variants for a list of items

```
RepositoryService repositoryService = RepositoryComponent.getRepositoryService();
Entity variantEntity =
repositoryService.getEntityByIdentifier( VariantCoreConst.ENTITY_VARIANT );

CatalogProxy currentCatalog = CatalogContext.getInstance().getCatalogProxy();

ReportListModelQuery query = new VariantByArticleQuery( articleProxies,
currentCatalog );
EntityItemList variantEntityItemList = query.toEntityItemList( variantEntity );
EntityItem[] variants = variantEntityItemList.toEntityItems( aProgressMonitor );
```

Revision: HEAD



Note that ListModelQuery is deprecated and not part of the API. It's better to use an EntityReport. Unfortunately there is no EntityReport with this functionality ready to be used. However, this might change in the future!

5.3.4.3 3 PPD: Resolve all parent products for a list of variants

```
RepositoryService repositoryService = RepositoryComponent.getRepositoryService();
Entity productEntity =
repositoryService.getEntityByIdentifier( ProductCoreConst.ENTITY_PRODUCT );

CatalogProxy currentCatalog = CatalogContext.getInstance().getCatalogProxy();

ReportListModelQuery query = new ProductByVariantQuery( variantProxies,
currentCatalog );
EntityItemList productEntityItemList = query.toEntityItemList( productEntity);
EntityItem[] productss = productEntityItemList.toEntityItems( aProgressMonitor );
```

Revision: HEAD



Note that ListModelQuery is deprecated and not part of the API. It's better to use an EntityReport. Unfortunately there is no EntityReport with this functionality ready to be used. However, this might change in the future!

5.3.5 Creating a superordinate object

5.3.5.1 Exclude fields

The creation of a superordinate object is basically a clone operation using the merge.

For setting values, the command operators (like initializers, validators and setters) of the product are used. Therefore the item is internally handled like a product (in 2PPD).

This means if you want to exclude certain fields from this process you have to set the field of the product (as example of the **superordinate object**) to **cloneable = false**. It's clonable and not mergeable because the merge uses a ClonePredicate in this case, which is checking the cloneable attribute of the fields.

5.4 Key Performance Indicator customization

5.4.1 Motivation

Product 360 provides a framework for the calculation of key performance indicators (KPIs). Therefore an extension point is provided which can be used to contribute custom implementations of KPI value calculators.

5.4.2 Extension point definition

The extension point **com.heiler.ppm.kpi.server.kpi** provides a contribution element: `kpi`. This element is used to define a KPI implementation with the following attributes:

Key	value
type	Unique identifier for KPI type. Indicates what kind of objects are being described by this KPI.
name	The human readable name of the KPI.
description	The description of the KPI.
daysToBeBacktracked	<p>Days to be backtracked from now, which is used to determine the start day to calculate the KPI Values. It must be an integer. Default value is 1.</p> <p>Example:</p> <ul style="list-style-type: none"> - when <code>daysToBeBacktracked = 1</code> and KPI values calculation is started at 2017-06-02 11:00:00 AM - then KPI values for the following days will be calculated: - 2017-06-01 (0:00:00 AM - 23:59:59 PM) - 2017-06-02 (0:00:00 AM - 11:00:00 AM)
calculator	The implementation class which calculates the actual KPI values.

The `calculator` class must implement the interface `com.heiler.ppm.kpi.server.KPIValuesCalculator`, which contains the following method:

com.heiler.ppm.kpi.server.KPIValuesCalculator

```
public interface KPIValuesCalculator
{
    /**
```

```

    * The calculation function which calculates KPI Values of certain objects. Created
    {@link CalculationResult}s must be
    * valid.
    * This means in concrete:
    * <ul>
    * <li>The first group value is set with a not-blank value</li>
    * <li>All group values below a group value have to be filled with not-blank
    values. For example if the third group
    * value is filled then also the second and the first one have to be filled with
    not-blank values.</li>
    * </ul>
    * @param calculationContext The calculation context.
    * @param progressMonitor The progressmonitor. Must not be <code>null</code>.
    * @param problemLog the problemLog. Must not be <code>null</code>.
    * @return A list of {@link CalculationResult}s.
    */
    List< CalculationResult > calculateKPIValues( CalculationContext
    calculationContext, IProgressMonitor progressMonitor,
                                                ProblemLog problemLog )
        throws InterruptedException, CoreException;
}

```

A KPI element can have up to five sub elements `groupLabel` . These can be used to define groups, in which the calculated KPI values can then be categorized. E.g. a KPI implementation describes workflows which can have a group label called "workflows" as it groups KPI values of different workflows.

The following table indicates the attributes of the sub element `groupLabel` .

key	value
<code>group</code>	The group itself which groups KPI data. Its value has to be selected from a predefined list.
<code>label</code>	The label of the current group.

You can define up to five different group labels for one KPI which are build as a hierarchical structure. This means that each filled group label needs to have a valid group label above it. For example if you fill your second group label, the first group label also has to be filled. If you fill your third group label, the first and the second group label have to be filled.

5.4.3 Custom implementations

5.4.3.1 Contributing a KPI

To add a custom implementation of `kpi` just contribute to the extension point `com.heiler.ppm.kpi.server.kpi` and create a class implementing the interface `KPIValuesCalculator`.

KPI Contribution sample

```
<extension
  point="com.heiler.ppm.kpi.server.kpi">
  <kpi
    calculator="com.heiler.ppm.kpi.sample.calculator.SampleKPICalculator"
    description="%sampleKPI.description"
    daysToBeBacktracked="1"
    name="%sampleKPI.name"
    type="sampleKPI">
    <groupLabel
      group="GROUP_LABEL_1"
      label="groupLabel_1">
    </groupLabel>
    <groupLabel
      group="GROUP_LABEL_2"
      label="groupLabel_2">
    </groupLabel>
  </kpi>
</extension>
```

5.4.3.2 Additional KPI setting


The administrator is able to add a corresponding preference in `plugin_customization.ini` to replace the defined `daysToBeBacktracked` for a single contributed KPI after the server is restarted.

The preference should use the following format, where `<numberOfDays>` must be a not negative integer value.

```
kpi.<kpiIdentifier>.daysToBeBacktracked = <numberOfDays>
```

For example:

```
com.heiler.ppm.kpi.server/kpi.mySampleKPIIdentifier.daysToBeBacktracked = 7
```

 During server start all contributed KPI implementations are validated and registered in Product 360.

5.4.4 Scheduling KPI implementation

A server job is running periodically to execute all contributed KPI value calculators. The job instances can be seen in the Desktop Client's Process overview in the category "Calculate KPI values" or be requested via Rest API.

The screenshot displays the Informatica Desktop Client interface. On the left, the 'Overview' pane shows a tree view of system processes, with 'Calculate KPI values (manual)' selected under 'Key performance indicators'. The main pane, titled 'Current and past processes (143)', shows a table of process instances for 'Calculate KPI values (manual)'. The table has columns: Number, User name, Date, Last change, Scheduled for, Status, and Progress. All instances shown are 'Completed' with 100% progress.

Number	User name	Date	Last change	Scheduled for	Status	Progress
1	10,115	rest	6/22/2017 12:37 PM	6/22/2017 12:37 PM	Completed	100%
2	10,113	rest	6/22/2017 12:18 PM	6/22/2017 12:18 PM	Completed	100%
3	10,112	rest	6/22/2017 12:18 PM	6/22/2017 12:18 PM	Completed	100%
4	10,111	rest	6/22/2017 12:18 PM	6/22/2017 12:18 PM	Completed	100%
5	10,110	rest	6/22/2017 12:18 PM	6/22/2017 12:18 PM	Completed	100%
6	10,109	rest	6/22/2017 12:18 PM	6/22/2017 12:18 PM	Completed	100%
7	10,108	rest	6/22/2017 12:18 PM	6/22/2017 12:18 PM	Completed	100%
8	10,107	rest	6/22/2017 12:18 PM	6/22/2017 12:18 PM	Completed	100%
9	10,106	rest	6/22/2017 12:18 PM	6/22/2017 12:18 PM	Completed	100%
10	10,105	rest	6/22/2017 12:18 PM	6/22/2017 12:18 PM	Completed	100%
11	10,104	rest	6/22/2017 12:18 PM	6/22/2017 12:18 PM	Completed	100%

Below the table, a log entry is displayed for the selected process instance. The log entry is titled 'Log - Log entries: CalculateKPIValues_SingleJob_10115_10115 (6)' and shows a sequence of messages related to the KPI calculation process.

Status	Category	Data type	ID	Message
1	Summary			Starting KPI Value calculation for specific KPIs.
2	i	Summary		Reference time is: 6/22/2017 12:37 PM
3	i	Summary		Starting calculation for simpleSampleKPI (simpleSampleKPI, 1 day(s) back)
4	i	Summary		The calculator of KPI 'simpleSampleKPI' created 1 result entry/entries for the interval 6/22/2017 12:00 AM
5	i	Summary		Finished calculation of KPI values.
6	i	Summary		KPI value calculation successfully finished.

In the server's `plugin_customization.ini` you can define at which time and how often the calculation should be executed. Default is every day at 0:00:00.

```
# Specifies the cron expression for the KPI Values calculation server job.
# Default is every day at 0:00:00
calculateKPIValuesJob.pattern = 0 0 0 ? * *
```

5.4.4.1 Validating the calculation results

After calculation the corresponding results will be validated for the following criteria:

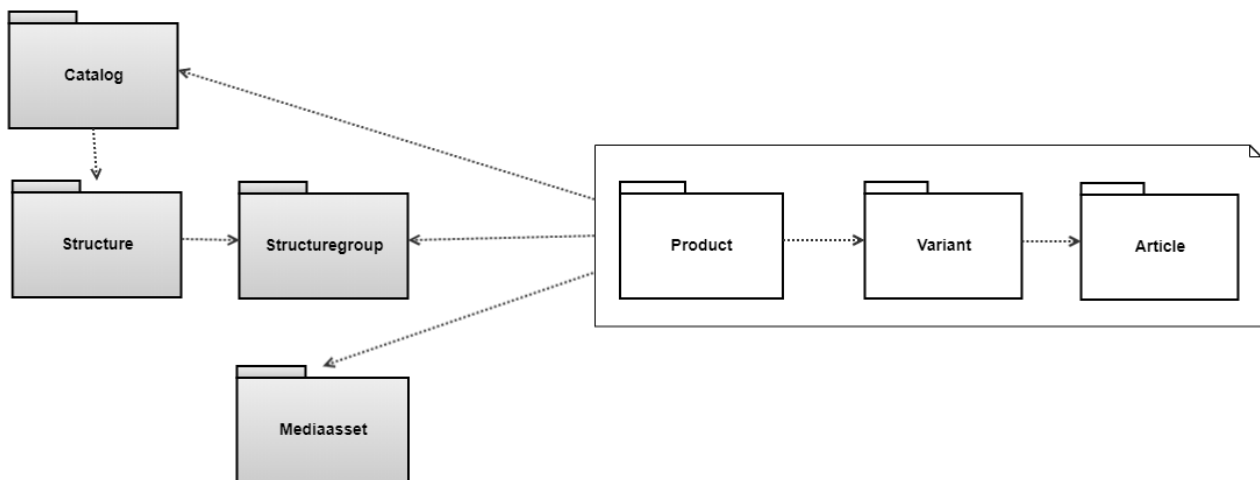
- all calculation results are unique, regarding the group values 1 - 5 and the `dataSetLabel`
- the group values are filled from bottom (1) to top (5)
- only group values that are defined for the given KPI are filled
- the `dataSetLabel` is not `null`
- the `value` is not `null`
- the `objectcount` is not negative

i Calculation results of one day are only stored in the database if all of them are valid. Validation errors are stored in the server job's problem log.

5.5 Datamodel

A bird view on the PIM's data model shows the main data entities of the PIM system. The basic data entity "Article" typically represents the SKU level in the hierarchy of product information. This is the entity of buyable items.

The higher level entities of "Variants" and "Products" are abstract describing entities grouping similar items by one or more defining attributes. The three level product paradigm is often used in the fashion industry where the entity "Variant" represents the color level of a certain article of clothing. The entity "Article" represents the different sizes of a certain color variant in that case. In other industries (with more technical products) often a two level product paradigm is used, where the entity "Variant" is not used and "Article" directly is related to "Product".



5.6 Structure

5.6.1 Copy Structures and/or StructureGroups

- 5.6.1.1
- [PIM 8](#)
 - [Copy Structures](#)
 - [Workflow](#)
 - [Structures of which structure types can be copied?](#)
 - [Structure Object Rights/ACLs](#)
 - [What else is copied, what is not?](#)
 - [Copy StructureGroups - General Workflow](#)
 - [Workflow](#)
 - [Strategies](#)
 - [CopyAllStructureGroupsOfStructure](#)
 - [CopyStructureGroup \(within one structure\)](#)
 - [CopyStructureGroupInOtherStructure](#)

5.6.1.2 PIM 8

This page and its subpages explain the copy process of structures and structure groups. There are three usecases to distinguish:

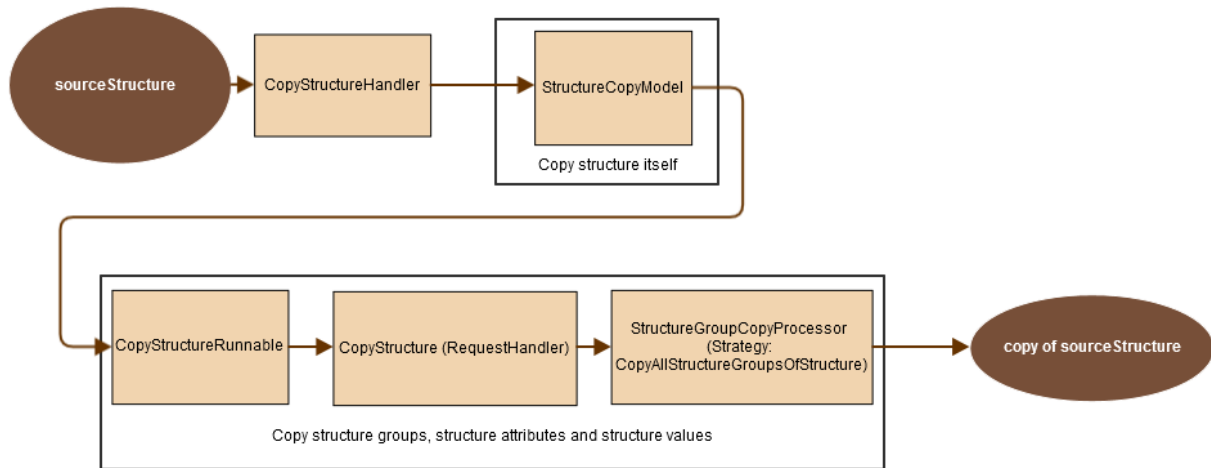
- copy a complete structure with all its structure groups, structure attributes and structure values
- copy a structure group with all its subgroups into another structure
(All structure attributes and structure values referenced from the copied structure groups and children need to be copied into the other structure if not already present.)
- copy a structure group with all its subgroups within the same structure

There used to be different implementations for these cases before. We refactored the code to always use the same basic workflow to copy structure groups, attributes and values.

Copy Structures

Copy structures is a bit of a special case because not everything is handled by the unified workflow. Additionally we have the structure object itself which need to be copied. As you can see in the picture below this step is handled by the StructureCopyModel.

Workflow



Structures of which structure types can be copied?

StructureType of Source	Can be copied	StructurType of Copy
Primary Maintenance	✓	Secondary Maintenance
Secondary Maintenance	✓	Secondary Maintenance
Output	✓	Output
Material group system	✓	Material group system
classification system	✓	classification system

Structure Object Rights/ACLs

Acls are copied. This means the values of the fields StructureType.AclFlag and StructureType.AclProxy are copied from the sourceStructure.

What else is copied, what is not?

Copied:

- StructureType.Type (Except for primary maintenance structures which are copied. The copy will become a secondary maintenance structure.)
- StructureType.MappingType
- StructureType.MappingLevel
- StructureType.GroupIdentifierPattern
- StructureType.Levels
- StructureType.UnitSystem
- StructureDetail.DisplayOrder
- StructureType.Revision
- language specific data (for all languages)
 - StructureLangType.FullName
 - StructureLangType.Description

Not copied:

- subentity StructureLogType
- StructureType.IsIdentifierHierarchical (probably not used anymore) - Will have database default value: true
- StructureType.IsBalanced (probably not used anymore) - Will have database default value: false
- StructureType.LastModified
- StructureType.UserId
- StructureType.DeletionDate
- StructureType.Status

Adjusted:

- StructureType.Type (Except for primary maintenance structures which are copied. The copy will become a secondary maintenance structure.)
- StructureType.Identifier - copyOf_<30 chars of the identifier of the sourceStructure>_<currentTimeMillis> shortened to 75 chars
- StructureLangType.Name - <"Copy of" in the according language of the EDataObject> <name of the source structure in the according language>

Copy StructureGroups - General Workflow

The general workflow in all three use cases is the same:

- find the things which need to be copied (e.g. structure attributes, structure values, structure groups)
- copy the things which need to be copied
- maybe handle assigned items/variants/products and their attributes

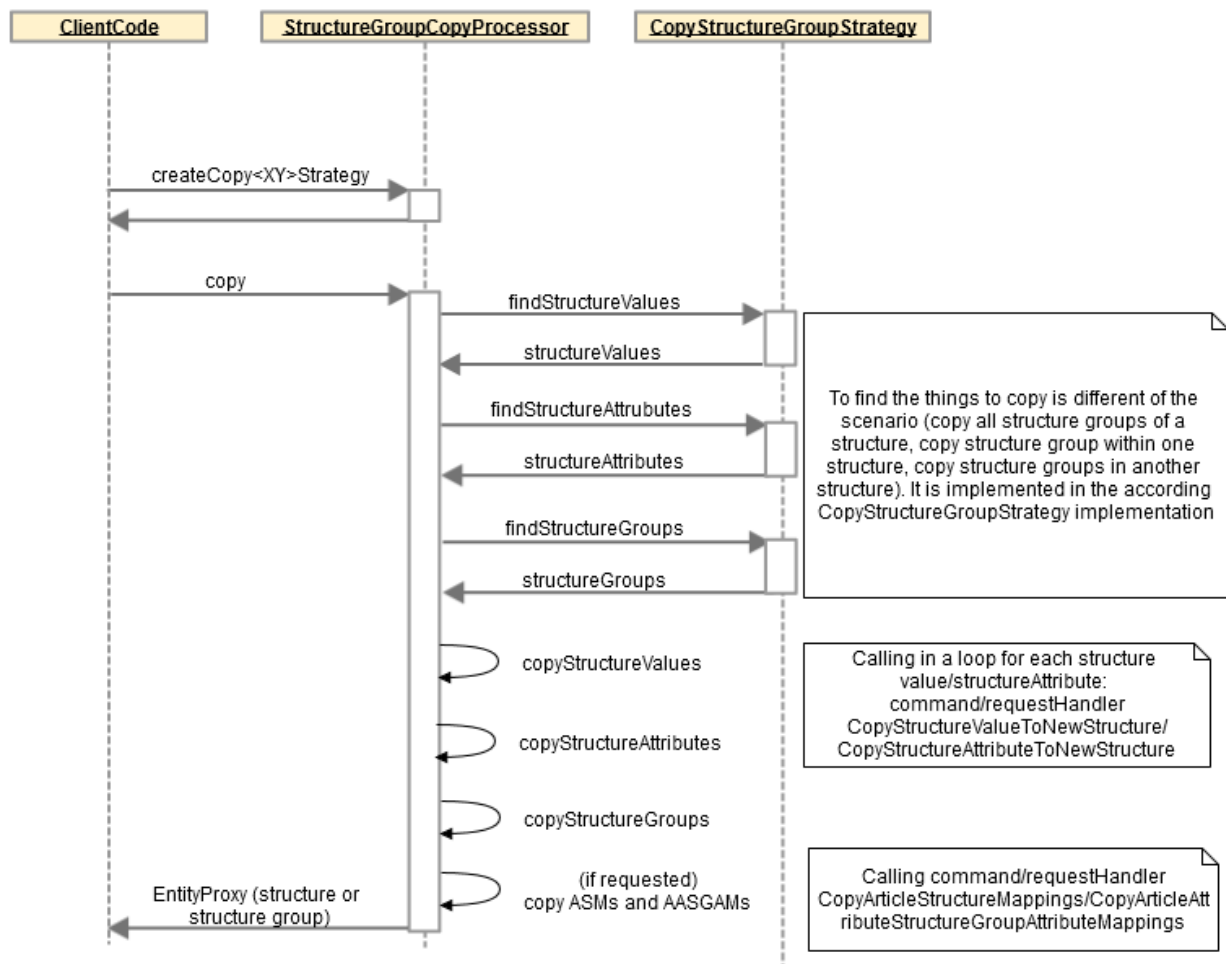
What varies is only the way the things which need to be copied are found. And this varying behaviour was therefore encapsulated in a special implementation of the interface CopyStructureGroupStrategy. There are static methods in the StructureGroupCopyProcessor to obtain the strategy object for your copy process. Note that this might change in the future!

To copy structure attributes, structure values, ASMs and AASGAMs requestHandler on the server side are used. There is no RequestHandler to copy a structureGroup at the moment. Note that this might change in the future. At the moment the structureGroups are copied in the StructureGroupCopyProcessor.

If the target structure is a maintenance structure no item/variant/product assignments and attribute assignments will be copied, no matter what the preferences say.

Workflow

Visit subpages to learn more about how a specific object is copied.



Strategies

CopyAllStructureGroupsOfStructure

- findStructureValues returns all structure values of the source structure (in the given revision)
- findStructureAttributes returns all structure values of the source structure (in the given revision)
- findStructureGroups returns all top level structure groups of the source structure (in the given revision). Child groups will be copied automatically by the copyStructureGroups method.
- returned value is the EntityProxy of the copied structure

CopyStructureGroup (within one structure)

- findStructureValues returns nothing. We copy a structure group within one structure so all values and attributes which are referenced by the structure group (and its children) are already present in the structure. the copied structuregroups reference the same objects.
- findStructureAttributes returns nothing. We copy a structure group within one structure so all values and attributes which are referenced by the structure group (and its children) are already present in the structure. the copied structuregroups reference the same objects.
- findStructureGroups returns the top level structure group the user selected to be copied. Child groups will be copied automatically by the copyStructureGroups method.
- the returned value is the EntityProxy of the copied structure group

CopyStructureGroupInOtherStructure

- findStructureValues traverses the preset values of all structure group attributes of the given structure group and recursively all its child groups and collects their referenced structureValues.
- findStructureAttributes traverses all structure group attributes of the given structure group and recursively all its child groups and collects their referenced structureAttributes.
- findStructureGroups returns the top level structureGroup the user selected to be copied. Child groups will be copied automatically by the copyStructureGroups method.
- the returned value is the EntityProxy of the copied structure group

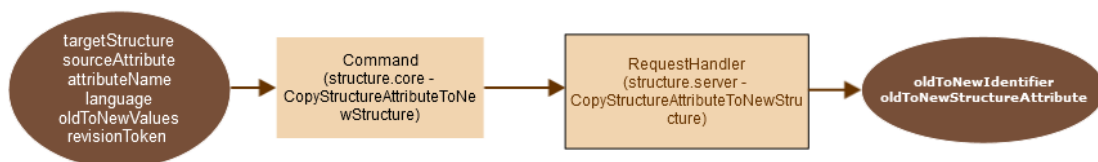
5.6.1.3 Copy Structure Attributes

- [Copy structure attributes \(see page 63\)](#)
 - [When is the given structure attribute considered to exist in the target structure? \(see page 63\)](#)
 - [Creating the copy \(see page 64\)](#)
 - [What happens if a structureAttribute references a structure value as preset value? \(see page 64\)](#)
 - [The new identifier \(see page 64\)](#)

Copy structure attributes

The basic idea of the Request Handler handling the copying of the structure attributes is to

- search if the structure attribute already exists in the target structure
- if not copy the structure attribute
- return the mapping of structure attribute proxy in the source structure to the structure attribute proxy in the target structure and the mapping of the source structure attribute identifier to the target structure attribute identifier (to be able to substitute these values in further processes like the copying of structure groups)



When is the given structure attribute considered to exist in the target structure?

Part of the input is a name and a language (usually the feature key language). It will be searched for an structure attribute with the given name in the given language in the target structure. If found this structure attributes proxy will be returned for the given sourceAttribute.

Note that neither names or other field values of the sourceAttribute are checked against the found target structure attribute nor is checked if the given name really belongs to the given source structure attribute

Creating the copy

For copying the structure attribute `createItemDataGraph()` and `createItem()` of the `structureAttributeManager` is used together with the `StructureEDataObjectCopyService`.

This means all field values except id, identifier, structureProxy, deletionDate and logs are copied.

Afterwards the structureValues in the subentity StructureFeaturePresetValueType are substituted according to the given oldToNewValues map.

What happens if a structureAttribute references a structure value as preset value?

When copying a structure attribute (in another structure) the structure value proxies of the preset values need to be substituted. Therefore the given oldToNewStructureValues map will be searched for the source structure value proxy.

If not found, the CopyStructureValueToNewStructure command will be called and a new structure value will be created in the target structure.

The new identifier

Compared to PIM 7 the creation of the identifier for a new structure Attribute changed.

If the identifier of the source structure attribute isn't already taken in the target structure, the copy will get the identifier of the source structure attribute.

Otherwise it will get a new identifier. A new identifier will have the format: <generatedId>_<identifier of the source structure attribute>. This identifier will be cut on the right hand side to fit the max length of the identifier field.

5.6.1.4 Copy Structure Groups

- [General Information](#)
- [Copy structure groups](#)
 - [The new identifier](#)
 - [Handling of Acls/Object Rights](#)
 - [Handling of DisplayOrders](#)
 - [Adjustment of hierarchical numbers](#)

General Information

Extension Points: none

API classes: none

Copy structure groups

StructureGroups are copied in the StructureGroupCopyProcessor. There is no RequestHandler as there is for StructureAttributes and StructureValues.

The copy logic recursively steps through the structureGroup which should be copied and all its children. To copy a structureGroup the structureGroupManager.createItem() is used. InitProperties for this are values for the fields

- StructureGroup.StructureProxy
- StructureGroupType.StructureId
- StructureGroup.Parent (There is a new StructureGroupParentPostSetter setting StructureGroupType.ParentId and StructureGroupType.ParentIdentifier automatically when the parent proxy is set)

Next all fields are copied using StructureGroupCopy.copyStructureGroup(). The following fields and subentities are omitted

- domainValues - This is the calculated field in the languageSpecificData containing the labels of the presetValues
- id
- revisionId
- left
- right
- parentId - Set (indirectly) in the init properties and different to the source
- parent - Set in the init properties and different to the source
- structureId - Set in the init properties and maybe different to the source
- structureProxy - Set in the init properties and maybe different to the source
- parentIdentifier - Set (indirectly) in the init properties and maybe different to the source
- level
- status - Each object has a 1:1 relationship to a status object, so they are not copied.
- deletionDate
- lastModified
- subentity: logs

StructureGroupCopy.copyStructureGroup() also handles the following adjustments:

- generates a new identifier if necessary
- filters invalid structureGroup mappings
- handles the copy of MediaAssets
- adjusts structureValue proxies and structureAttribute proxies and identifier

After a structure group is copied the acls are adjusted using the UpdateAclInheritance RequestHandler.

The new identifier

Compared to PIM 7 the creation of the identifier for a new structure Attribute changed.

If the identifier of the source structure attribute isn't already taken in the target structure, the copy will get the identifier of the source structure groups.

Otherwise it will get a new identifier. A new identifier will have the format: <generatedId>_<identifier of the source structure attribute>. This identifier will be cut on the right hand side to fit the max length of the identifier field.

Handling of Acls/Object Rights

First AclProxy and AclFlag are copied, next the UpdateAclInheritance Request Handler is called. If for example a structureGroup is copied to another structure group with Object rights, so that this structure group becomes the parent, the object rights might have to be inherited to the copied structure group. For more information see Inheritance of Acls

Handling of DisplayOrders

When copying a structureGroup, only the displayOrder of the top structureGroup (i. e. the one you selected to copy) is adjusted. Child groups keep their displayOrder.

For more information on the generation algorithm of a new displayOrder see Handling of DisplayOrders - A comparison

Adjustment of hierarchical numbers

Hierarchical numbers aren't adjusted at all. This is a known problem. If you/your customer runs into problems with hierarchical numbers please talk to PM about the expected behaviour and file a JIRA issue.

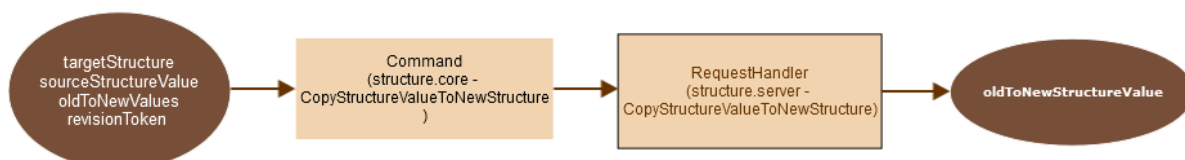
5.6.1.5 Copy Structure Values

- [Copy structure values](#) (see page 66)
 - [When is the given structure value considered to already exist in the target structure?](#) (see page 67)
 - [Examples](#) (see page 67)
 - [Creating the copy](#) (see page 68)
 - [The new identifier](#) (see page 68)

Copy structure values

The basic idea of the Request Handler handling the copying of the structure values is to

- search if a single structure value already exists in the target structure
- if not copy the structure value
- return the mapping of structure value proxy in the source structure to the structure value proxy in the target structure (to be able to substitute these proxies in further processes like the copy of structure groups)



When is the given structure value considered to already exist in the target structure?

A structureValue is considered already existing if

1. a structureValue with the same **identifier** exists in the target structure **and**
2. the structureValue in the target structure has no **names** different to the names of the source structure value (if the source or the target structureValue have names which the other one doesn't have, that's ok).
Note: Missing entries will be added in the target structureValue

Examples

A structure value in the source structure has

- Identifier: XYZ
- Name (de) Grün
- Name (en) green

and will be copied in on of the following target structures

Target Structure	StructureValue Identifier	StructureValue Name(de)	StructureValue Name(en)	What to do?
Structure A	ABC	Grün	green	create new structure value add proxies to result map
Structure B	ABC	Grün	<something else> light green	create new structure value add proxies to result map
Structure C	XYZ	Grün	green	do nothing
Structure D	XYZ	Grün	<something else>	create new structure value add proxies to result map

Target Structure	StructureValue Identifier	StructureValue Name(de)	StructureValue Name(en)	What to do?
Structure E	XYZ	XL	XL	create new structure value add proxies to result map
Structure F	XYZ	Grün	<empty>	add english name to existing structure value
Structure G	XYZ	<empty>	<empty>	add names to existing structure value

Creating the copy

For copying the structure value `createItemDataGraph()` and `createItem()` of the `structureValueManager` is used together with the `StructureEDataObjectCopyService`.

This means all field values except id, identifier, structureProxy, deletionDate and logs are copied.

The new identifier

Compared to PIM 7 the creation of the identifier for a new structure value changed.

If the identifier of the source structure value isn't already taken in the target structure, the copy will get the identifier of the source structure value.

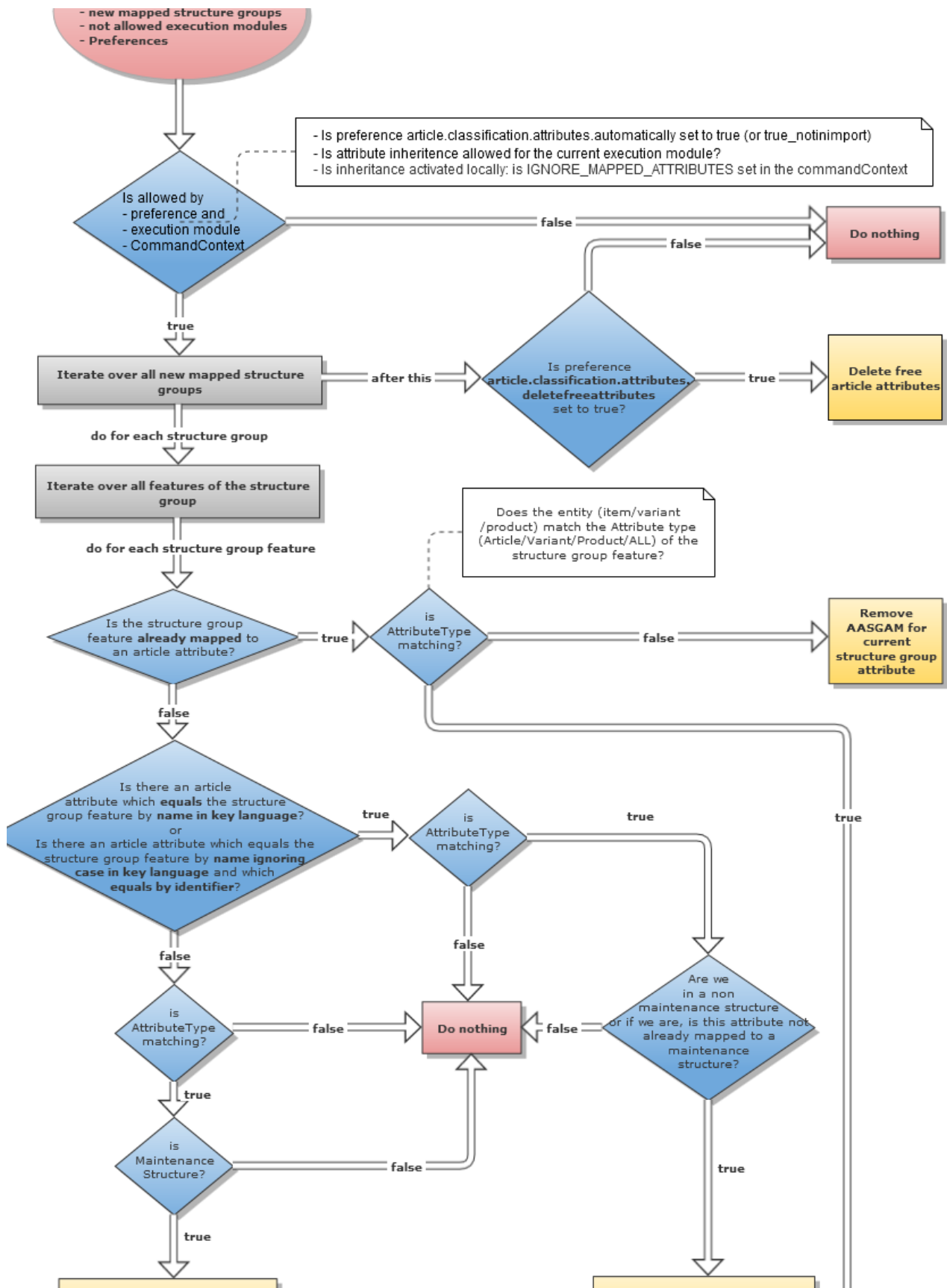
Otherwise it will get a new identifier. A new identifier will have the format: <generatedId>_<identifier of the source structure value>. This identifier will be cut on the right hand side to fit the max length of the identifier field.

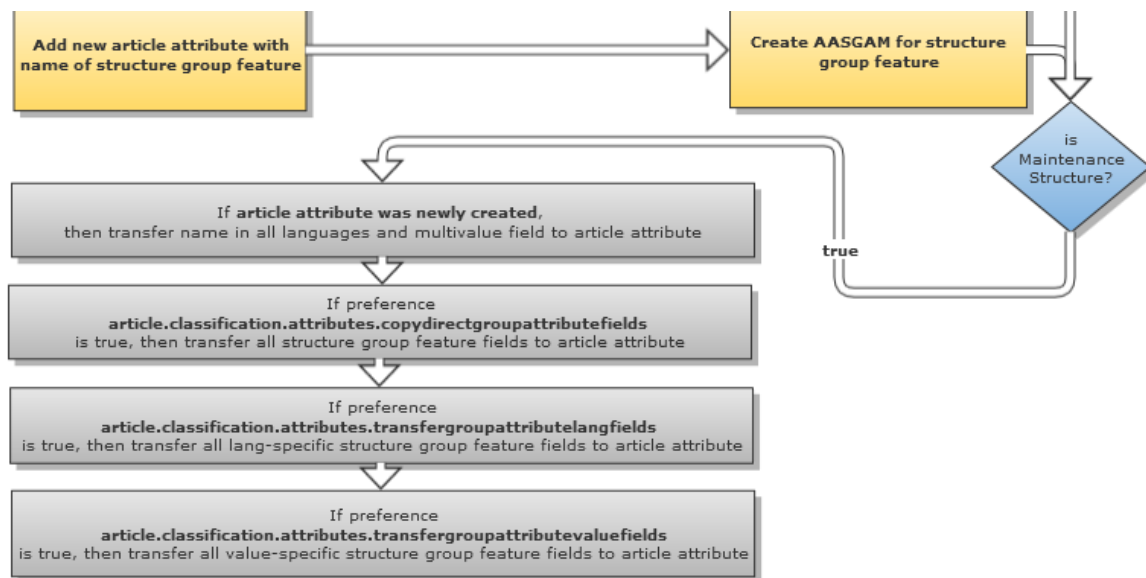
5.6.2 Structure Group

5.6.2.1 Inheritance of StructureGroupFeatures to items/products/variants

The following flow diagram describes the behaviour when classifying an item/product/variant to a structure group (f.e. when Drag'n'Dropping Article(s) onto a structure group).







i In this diagram, the word "Article" may also be used as a synonym for an article-based-entity (Article, Variant, Product).

5.6.2.2 Sorting of structure groups (StructureGroup.DisplayOrder)

i since PIM 7.1.01

How are structure groups sorted?

In the tree of the Primary (/Secondary) Structure View, structure groups are shown in a sorted way, using the following criterias (in the specified order):

- StructureGroup.DisplayOrder (=Sequence)
- StructureGroup.StructureGroupIdentifier.HierarchicalNumber (=Hierarchical Number)
- StructureGroup.StructureGroupLang(structureLanguage).name (=Name)
- StructureGroup.Id

What is a display order and what is it used for?

A display order is an integer value stored in the field StructureGroupType.DisplayOrder. It is used to determine in which order structure groups and child structure groups should be displayed in the structure tree views for example (see above).

When are display orders changed?

There are three situations in which a new display order has to be determined.

- a new structure group is created

- a structure group is moved
- a structure group is copied
 - within one structure
 - into another structure

Default display order

The default display order is Integer.MAX_VALUE = 2147483647.

How are display orders generated?

- New structure groups are always created with the default display order, independent of through which action they are created.
- When moving or copying a structure group into a structure group (as child) in the same structure without concrete position (this is done by drag'n'dropping it directly on to a structure group in the "Structure View" tree), it always gets the default display order.
- When moving or copying a structure group to another structure, it always gets the default display order. Copied child structure groups keep their display order.
- if a whole structure is copied, all structure groups keep their display order
- When moving or copying a structure group to a concrete position (between two other structure groups, before or after a structure group), it gets a concrete display order, which is calculated with help of the surrounding structure groups.
 - If in this case, no space is left between the surrounding structure groups (concerning the display order), the display orders of all child structure groups of the parent structure group are redefined. Therefor the sort order criterias mentioned above are used.

6 Server and Platform

This section covers the PIM core architecture. It is a comprehensive documentation of the general architecture and the principles which were used in the core.

- [Architecture principles \(see page 71\)](#)
 - [Domain model centric \(see page 72\)](#)
 - [A base for customer specific solutions \(see page 72\)](#)
 - [Migrate customer specific solutions to newer versions at low cost \(see page 72\)](#)
 - [Solution provides rich client UI as a desktop application \(see page 72\)](#)
- [Architecture main concepts \(see page 72\)](#)
 - [Advantages of the approach \(see page 73\)](#)

For training purposes we created a presentation which covers the main parts of the platform architecture which can be downloaded here. PIM_Core_Platform.pptx

6.1 Architecture principles

The architecture is driven by four main aspects:

1. Domain model centric
2. Base for customer specific solutions which are developed externaly (product line)

3. It is possible to migrate customer specific solutions to newer versions at low cost
4. Provides rich client UI as a desktop application as well as a Web UI and Service interfaces

6.1.1 Domain model centric

Most of the functionality is focused on data management: import, creation, classification, validation, presentation, export to external formats etc. PIM core has a declarative domain model. The domain model contains business entities, descriptions and relations between entities as well as meta information for the data management functionality.

6.1.2 A base for customer specific solutions

Customer specific solutions which are developed externally (product line)

Hpm provides "vanila" domain model (PIM model shared by different PIM branches) and corresponding logic. In most cases this domain model must be adjusted and extended to suite needs of a concrete customer. Customization is also performed on the other levels such as presentation - to support user workflows, integration - to connect HPM with other systems and data sources, data management - to validate input data, etc. Application integrator/developer has a possibility to extend any part of the standard applicaiton if it is required.

6.1.3 Migrate customer specific solutions to newer versions at low cost

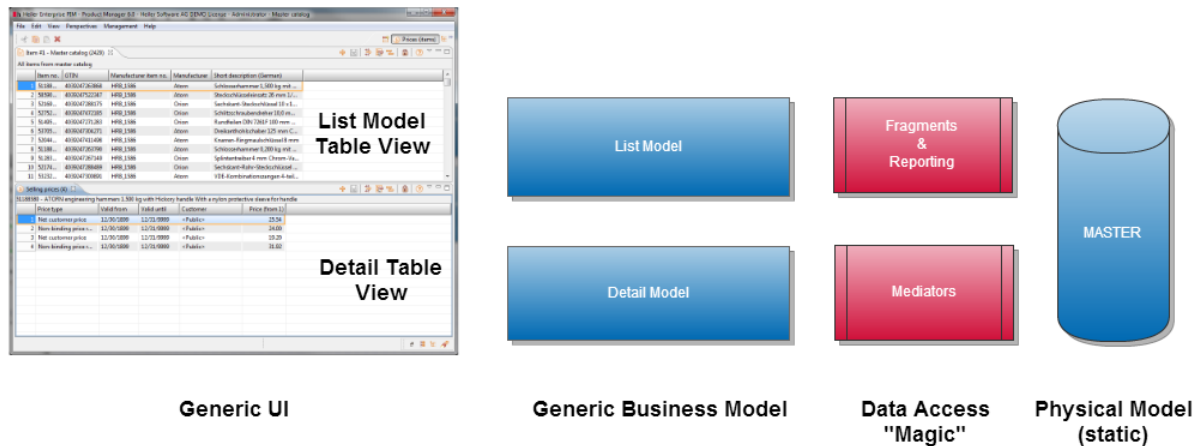
It is a common situation that customer specific solutions are migrated to newer versions of PIM. For this reason the public API is backward compatible and a migration path is provided. If new programming concepts in the platform replace old ones then a compatibility layer is provided whenever this is feasible (or possible)

6.1.4 Solution provides rich client UI as a desktop application

Clients contain not only presentation but also business logic. In a such distributed environment data modifications are synchronized accross many distributed nodes. Client have different presentation of the data structures and is capable of presenting large datasets.

6.2 Architecture main concepts

- Architecture is domain model centric. Model structure is described declaratively and additionaly contains meta info for different functional areas.
- The system consists of functional layers which interprets the structure using meta info.
- No code generation with round trip.
- Layers of abstraction to support customization
- Different public APIs (data access, integration, ui, etc.)

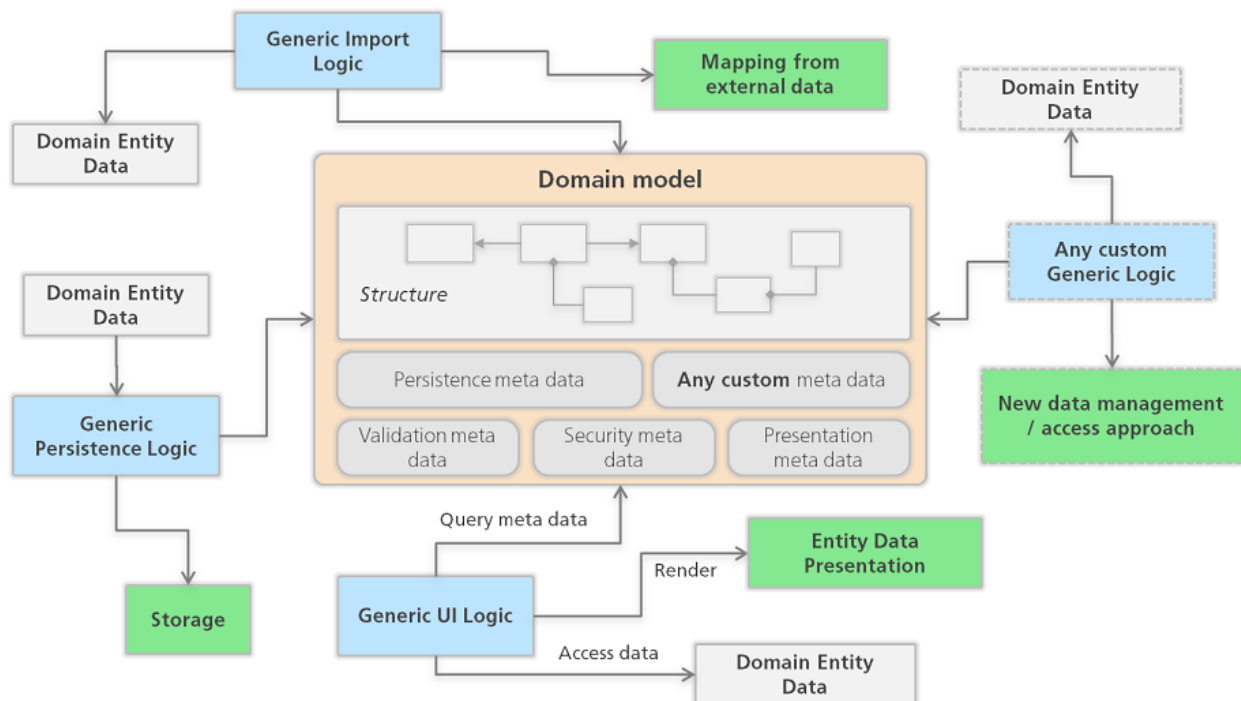


This picture shows the general overall architecture in respect to the most important aspects. The key to the outstanding performance in mass data access is a separated model with it's own data access layer for mass data retrieval. Additionally to that we combined a static physical model with a generic business model. This provides us the high performance of a classic, normalized database structure with all the abilities to create customer specific indizes and optimizations as well as the flexibility needed to have additional fields and entities for each customer.

6.2.1 Advatages of the approach

- Domain model structure can be changed without changing the logic
- New functional layers can be added without changing the existing ones

Figure 10.10 Architectural Principles



1 HPM_Arch_Principal.png

6.3 Multi-Server Architecture

This page describes the general architecture of the Informatica PIM 8 Multi-Server support.

- [Synopsis](#) (see page 75)
- [Demand for Multi-Server](#) (see page 75)
 - [Response times worsen with more and more users working concurrently](#) (see page 75)
 - [Background jobs interfere with users](#) (see page 75)
 - [More channels with bigger assortments need to be updated](#) (see page 75)
 - [High-End Multi-Core CPU Systems are expensive](#) (see page 76)
- [Architecture](#) (see page 76)
 - [Server Roles](#) (see page 78)
 - [Client Server](#) (see page 78)
 - [Job Server](#) (see page 78)
 - [Web Server / Service API](#) (see page 78)
 - [Message Queue Consumer Server](#) (see page 78)
 - [Load Balancing](#) (see page 78)
 - [Client Requests](#) (see page 78)
 - [Jobs](#) (see page 79)
 - [Web Clients and Service API Clients](#) (see page 79)
 - [Message Queue Consumer Server](#) (see page 79)

6.3.1 Synopsis

More and more customers face the need to increase the size of their item and product assortment which is managed in Informatica PIM. Not only the pure object count for these objects increase, but also the number of fields per item. Business needs for more distribution channels and a collaborative approach for maintenance increase the number of jobs as well as the concurrent users which work using the PIM Desktop as well as PIM Web or other clients attached by the Service API.

The only option to increase performance on the Server in the past was to increase the number of CPU's or CPU Cores. However, this so called **vertical scalability** has a natural limit in the availability and cost of high-end multi core CPU's. The solution to this problem is **horizontal scalability**.

By adding additional application servers to the PIM installation the customer can further increase the performance and the maximum load the system can handle. Additionally the customer can use mid-size hardware for these application servers, keeping the hardware costs for the PIM solution at a reasonable figure.

6.3.2 Demand for Multi-Server

The new multi server architecture of PIM provides answers to multiple issues customers face in high load scenarios.

6.3.2.1 Response times worsen with more and more users working concurrently

By adding additional application servers the users will be distributed across these servers - which increases the response time for every single user.

6.3.2.2 Background jobs interfere with users

Background jobs like Import, Merge or Export put high load on the application server which leads again to bad response times for users. Informatica PIM 8 multi-server edition can separate the execution of jobs and client requests. You can specify dedicated servers for background job processing as well as client connections. This helps to prevent that an executing background job affects users.

6.3.2.3 More channels with bigger assortments need to be updated

Background jobs are affected by the increased total number of objects in the PIM system. The more products and items are exported or imported the longer it will take. At the same time the number of distribution channels which should be updated with this data increase as well. Sooner or later the day won't have enough hours to execute all those jobs.

By adding multiple application servers for background job processing the PIM 8 multi-server edition can update more channels in the same amount of time.

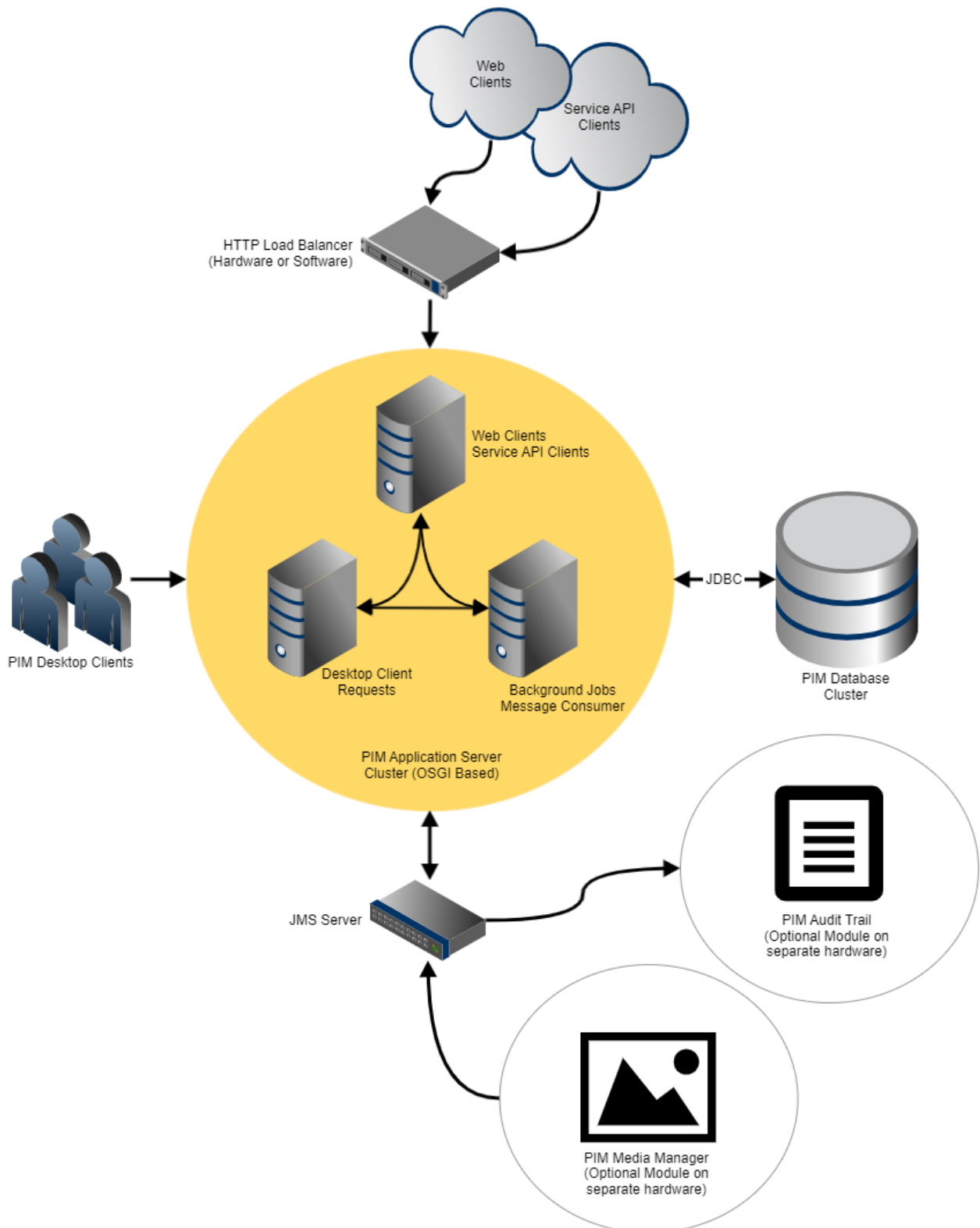
6.3.2.4 High-End Multi-Core CPU Systems are expensive

High-End impose tremendous costs on the application landscape. Two mid-size application servers are usually way cheaper than a single high-end one. Informatica PIM 8 multi-server edition distributes mass-data operations on all available application servers. It can't handle "just" more users or more background jobs, it also provides faster execution time for a single job by distributing the load of that job to all application servers. Using this technology the customer is able to leverage cheaper hardware by providing multiple application servers.

6.3.3 Architecture

Multiple application servers can be combined to form a cluster. The application servers are connected with each other, so every server has a connection to all other servers. This eliminates the need for a centralized server which is by definition a single-point of failure and usually a scalability bottleneck in terms of communication overload (since all messages must go through him).

All servers have the full application functionality. In case of a server shutdown or a hardware failure, other servers can (and will) take the load of the missing server. This reduces the need for special fail over hardware.



6.3.3.1 Server Roles

A server role is a logical categorization of server instances. Currently these roles are defined at deployment time of the cluster. A server instance can also incorporate multiple roles (that's how the single server deployment is realized). The fact that these roles are pure logical roles and do not imply a different deployment approach in terms of bundles and features will provide the possibility to have dynamic roles in the future.

Client Server

The "client server" role defines that this server handles rich client requests. So rich clients are allowed to connect to this server and send requests etc. Multiple servers can have this role.

Job Server

The "job server" role defines that this server executes server side jobs like import, export, merge, search/replace etc. Multiple servers can have this role.

Web Server / Service API

All application server instances are identical in terms of installed functionality and have the needed server components for web client requests or service API requests. An explicit "web" or "service api" role is not available since this is defined by the load balancer anyway.

Message Queue Consumer Server

A server which has this role is allowed to consume messages from the message queues defined in the section "Message Queue Settings" of the server.properties file. The default queue configurations in the server.properties file are used for the communication from Product 360 to BPM and vice versa. Also all DQ jobs which get triggered by entity change and entity created triggers are written to a message queue and will be consumed by all servers which have this server role.

6.3.3.2 Load Balancing

Client Requests

Clients connect during start-up with an arbitrary server of the network, it actually doesn't matter which one since all servers know the whole network (which servers are there, which clients are connected to which server). From this "lookup" server he receives the list of available servers which have the "client server" role. In case there are only "job servers" available, the client will also connect to to a job server. After the determination of available servers, the client disconnects from the lookup server and reconnects to the one with the least number of connected clients. Clients stay connected to the server the whole time they run.

Jobs

The Quartz job framework we utilize is capable of having multi-server deployments. All jobs to be executed are stored in the database. Every "job server" in the network will now poll the quartz tables and see if there are new jobs to do, up till all job worker threads are full. This way, the jobs will be balanced on all servers. A differentiation between job types is not supported.

Web Clients and Service API Clients

Web and Service API clients are similar in the way they connect to the server, both use the standard HTTP/HTTPS protocols. Therefore we suggest to use a hardware or software load balancing functionality which is available from various vendors. Please make sure that the load balancer is configured to use "sticky sessions" in case of the web clients. The service API is stateless, so it's not required there, but recommended due to the caching logic inside the service API.

Message Queue Consumer Server

The messages from message queues are evenly distributed to all servers with the appropriate server role to consume messages from the queue. Additionally the thread count for consuming from a specific queue and writing to specific queues can be customized in the section "Message Queue Settings" of the `server.properties` file.

6.4 Domain Model (Repository)

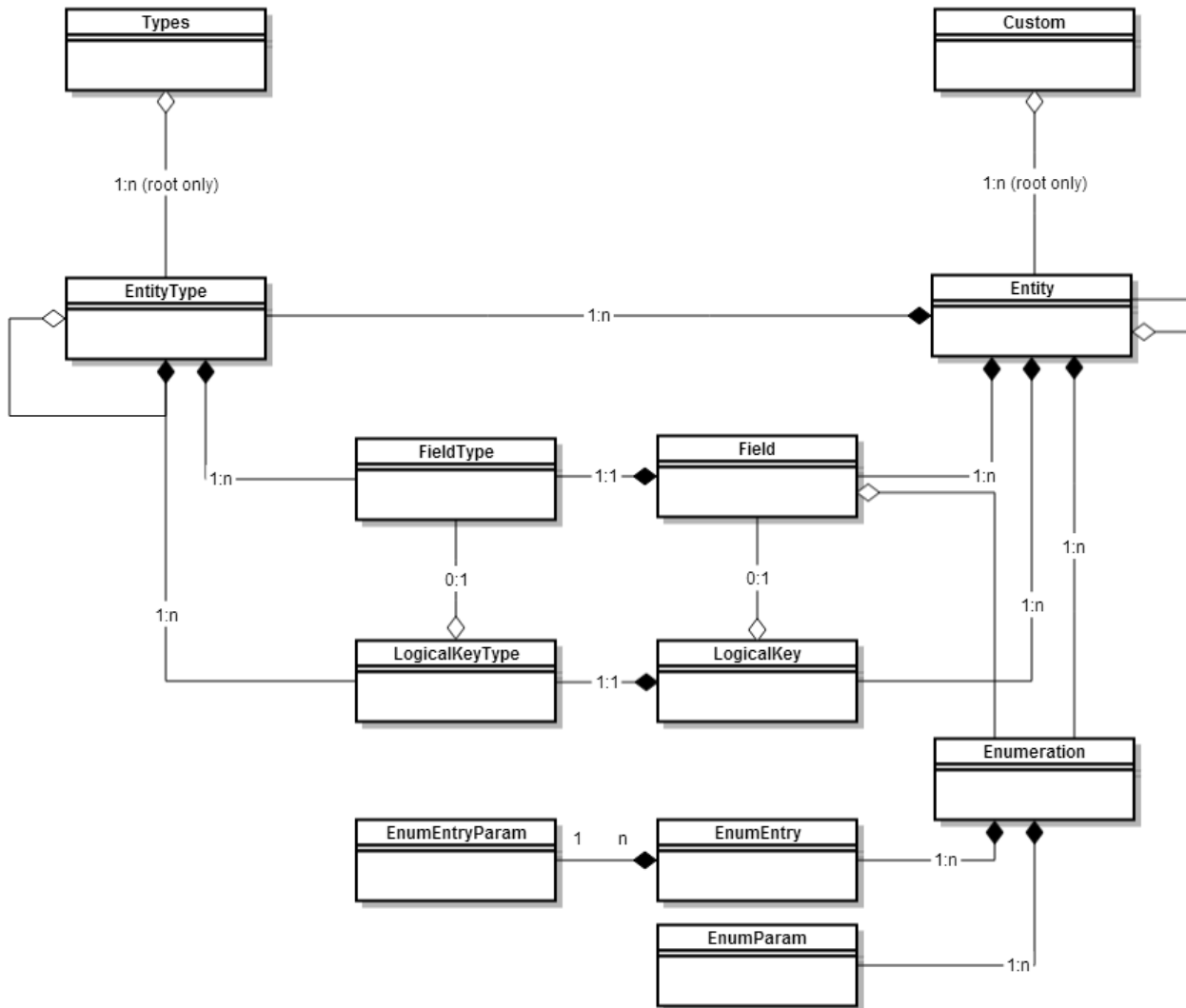
Users of the SDK can obtain the repository information through the object model and a variety of utility methods located in the `RepositoryUtils` class. For customizing purposes of the Repository you can use the Repository Editor which is either integrated in the SDK distribution or available as stand alone tool. In the following sections you will find a description of the repository configuration file, respectively the object model. You will find an explanation on each property and how a modification of those affect the Product 360.

In general, the repository has two parts. The types and the custom part. The types part reflects the business model of Product 360 whereas the custom part defines logical "views" of the business model.

- [General architecture](#) (see page 81)
- [Types Part](#) (see page 82)
 - [Entity Type](#) (see page 82)
 - [Application](#) (see page 82)
 - [Constraints](#) (see page 83)
 - [General](#) (see page 83)
 - [Misc](#) (see page 83)
 - [Persistence](#) (see page 83)
 - [Deprecated](#) (see page 84)
 - [Proxy Type](#) (see page 84)
 - [Entity Type Param](#) (see page 84)
 - [Logical Key Type](#) (see page 85)
 - [Field Type](#) (see page 85)
- [Custom Part](#) (see page 87)
 - [Entities](#) (see page 87)

- [Multiple Entities for the same Entity Type](#) (see page 87)
- [The attributes of entities:](#) (see page 88)
- [Entity Param](#) (see page 91)
- [Logical Key](#) (see page 92)
- [Field](#) (see page 93)
- [Permissions](#) (see page 98)
- [Audit Trail Settings](#) (see page 99)
- [Enumeration](#) (see page 99)
 - [Enum Entry](#) (see page 100)
 - [Enum Entry Param](#) (see page 100)
 - [Enum Param](#) (see page 101)
- [Permission Category](#) (see page 101)
- [Category](#) (see page 101)
- [Multi-language support of the repository](#) (see page 102)
 - [Referencing on logical keys in field names](#) (see page 102)
- [Picture-clause syntax](#) (see page 102)
- [Repository Manager](#) (see page 103)

6.4.1 General architecture



This diagram shows the general repository structure which will be explained in detail in the following chapters.

The general idea of this architecture is to provide a "safe for customization" part, and a "this needs code modification" part. In the types part of the repository you will find all "hard facts". It constructs the generic business model of Product 360 and therefore any change in the types area might affect or even break working code. In contrary to this defines the custom part all things which are not really necessary to know at coding time. For example labels and descriptions of fields. However, during the evolution of the repository this strict distinction has become a bit messy. In this documentation you will therefore see some hints also in the custom part that you shouldn't change certain attributes, and in the types area you will need to set reserved fields to "inactive=false" in order to use them.

6.4.2 Types Part

The types part defines the available entities and fields of Product 360. It is used to build the generic business model on which the concrete business logic is implemented. It is also responsible for the dynamic data access logic which is available in Product 360. Developers are intended to treat the types part also as some kind of "generic", so that not every change leads to problems, but in general they can rely on what is defined here. Attention: Any changes in the types part, might lead to fatal damages to the Product 360 system. Changes in the types area are treated as modifications to the system and are not allowed.

Following Exceptions are allowed:

- Changing 'Reserved Fields'
 - setting to active
 - adjust class name to e.g. enable usage of entity proxies
- Adding complete new top entities
- HMM-Provider: Adding new HMM technical attribute fields as **new** field types in MediaAssetDocumentAttributeType area.

6.4.2.1 Entity Type

Entity types are no subject for customizing in any way, they should be treated as totally internal. Entity types are defined in the types section of the repository (repository/types/entity-type). Entity types in general group fields and sub entity types. This grouping is mainly controlled by the relationships and the dimensions. For example, the ArticleType top-level entity type, has subentity types called ArticleLangType and an ArticleLogisticType. Both sub-entity types contain fields which belong to an article in general, but they are in different entity types because they have a different set of dimensions. Field types belonging to ArticleLangType can be language dependent, field types belonging to ArticleLogisticType can be territory dependent, but not language dependent. In terms of a relational database you would say that ArticleLangType has a unique index (Article + Language), and ArticleLogisticType has a different unique index (Article + Territory). The concrete modeling of the entity types is somewhat historic and of course also bound to our database structure more or less, in no way it can or should be changed for existing entity types – this would break the code which relies on this structure. We distinguish two kinds of entity types, the "top-level" entity types and the "sub-entity-types". Top-Level entity types are those which reside directly under the Types node in the repository, thus have no parent entity type.

Available attributes of entity types

Application

- Bulkload Order
(Root entity types only): Defines the order of entities for the import. This settings is mostly used in the import scenario where the user is able to import entity items of different types in one import. E.g. StructureGroups before Articles.
Although the name suggest something else, this setting is not deprecated.
- Max System Id
(Root entity types only): Defines the maximum id which is reserved for internal purposes; e.g. standard structure systems have a range from 1-8999, user systems start with id 9000.

Constraints

- Lower Bound and Upper Bound
The lower bound property specifies the cardinality of this entity type, regarding its parent. Possible values are 0 or 1 for lower-bound, where 0 means the entity is completely optional, and 1 or -1 for the upper-bound, where -1 means an unlimited amount of objects, and 1 exactly one child object. A fixed definition of 1..5 for example is not supported yet.

General

- Entity Id Field Type
- Identifier
Unique alphanumeric identifier of the entity type. The convention for entity type identifiers is: <Name>Type (e.g. ArticleType, ArticleLangType, StructureType, etc.).
- Identifying Field Type
Defines the field type which identifies the entity type uniquely. Currently this property is used in the export to be able to use the entity type in a sub-module. You could say this field type identifies the primary key of the corresponding database table (combined primary keys are not supported).
- Object Name
The object name of an entity type makes it accessible by property accessors. For example, to access the list of surcharges of an article, you would use: article.get("surcharges"). The returning value would be a list of ArticleSurchargeType objects.
- Report Type (mandatory for top-level entity types)
The report type is a unique integer which identifies a root entity within the reporting framework uniquely. The report type property applies only to root entities.
- Status Proxy Field Type

Misc

- Container Entity Type
(Root entity types only): Some entity types are unique only within a container. Like ArticleType which is unique only within a CatalogType, or StructureGroupType within a StructureType. This attribute defines the container entity type.
- Documentation
This attribute can be used to document the repository content, it is not being used by any logic in Product 360 currently. Providing meaningful explanations about the entity type is recommended since the attribute might be used in the repository editor someday.
- Referenced-entities
- Searchable

Persistence

- AuditLog Entity Type
(Root entity types only): Defines the sub-entity type which stores the audit log data for the entity type. See the documentation for audit log support on details. If no entity type is defined, no audit log is supported for this entity type.
- Deletion Date Field Type
- Deletion Mode
- Persistence Class Name
Defines the class name of the corresponding class in the persistence layer. This attribute is used to create generic HQL statements, especially in the generic search functionality. Therefore it's a mandatory attribute for all searchable entity items as well as all entity types which support object rights

- Persistence XPath (mandatory for sub-level entity types)
The XPath to the corresponding persistence objects.
- Revision Persistence Class
- Revision Property
- Supports Versioning
(Root entity types only): Defines if this root entity supports the versioning functionality. This setting must never be changed, since in case it's false, the persistence layer of this entity is not capable of versioning.

Deprecated

- Class Name
Deprecated setting which must, for compatibility reasons, be set to the same value as identifier.
- Help Context
Deprecated: Has never been used
- Supports Bulkload
Deprecated: Bulkload is not longer used.

Proxy Type

Each top-level entity type has a proxy type element which defines the construction details for the corresponding entity proxy classes. See the "Model" section of the programmer's guide for detailed information about entity proxies.

- internal-constructor-field-type-refs
List of identifiers of field types which are needed for the "internal" constructor of the entity proxy. The order of field types must correspond to the internal constructor of the entity proxy. The internal constructor is absolutely mandatory for an entity proxy.
- external-constructor-field-type-refs
List of identifiers of field types which are needed for the "external" constructor of the entity proxy. The order of field types must correspond to the external constructor of the entity proxy. If the entity proxy has no external constructor this field can be empty.
- mixed-constructor-field-type-refs
List of identifiers of field types which are needed for the "mixed" constructor of the entity proxy. The order of field types must correspond to the mixed constructor of the entity proxy. If the entity proxy has no external constructor this field can be empty.
- proxy-class-name
Fully qualified class name of the entity proxy implementation of this entity type.

Entity Type Param

Generic list of key/value parameters which can be defined for every entity type. These parameters can also be used by customizings as the values can be interpreted at runtime. The following list of parameters are supported currently:

Key	Value	Description
supportsVirtual	true/false	Flag indicating if this entity supports the virtual table mode

Logical Key Type

As described before, entity types are distinguished by the dimensions they have. So, with the logical keys, one can define these dimensions for an entity type. Every logical key type adds one dimension to the corresponding entity type. For example, the ArticleLangType has only one logical key for "language" which means that this entity type has one dimension, ArticlePriceType has several more. Any changes of the logical keys would make the complete business logic of this entity type useless and would lead to damages to the system. The logical key types correspond for example also to the unique indexes in our database. A change of the logical key type needs also a change in the unique index.

The properties of logical key types

- class-name
If the logical key has no direct 1:1 relationship to a field type, a class name must be set which defines the business data type of the logical key.
- documentation
This attribute can be used to document the repository content, it is not being used by any logic in Product 360 currently. Providing meaningful explanations about the logical key type is recommended since the attribute might be used in the repository editor someday
- field-type (either the field type or the class name are mandatory!)
Most logical keys directly refer to a field in the same entity type. So all needed information regarding data types etc. will come from this field type then. This property is optional, if not set, a class name is needed.
- fragment-expression
Can hold additional information which is needed in certain specific fragment implementations, usually its empty. The usage is up to the fragment implementation.
- identifier
The unique identifier of the logical key type. All identifiers in the repository must be unique over the whole repository. The naming convention for logical key type identifiers is: <EntityType>_LK_<Name of Logical Key> (ArticleLangType_LK_Language, ArticlePriceType_LK_Buyer, etc.)
- persistence-class-name
Defines the persistence class of the logical key, if no field type is set. Defaults to the class name if omitted.
- persistence-filter-name
Defines the name of the Hibernate filter definition which should be used for the logical key. If omitted no hibernate filter will be used. This attribute has two parts which are separated by a colon (":"). The first value is the name of the filter definition and the second value is the name of the filter parameter. The filter definition must correspond to the persistence model filter annotations.
- xpath-expression
Deprecated and no longer used since Heiler Product Manager 4.6.

Field Type

Field types represent concrete data fields in the persistence storage. The attributes of field types:

- class-name
Defines the business data type of the field with the fully qualified java class name. Fields can have different persistence and business data types. For example, the field ArticleType.Catalog is a "Catalog" from the business point of view, from the persistence storage it's a simple CatalogID which is just an integer. The generic data access facilities use the ConvertUtils of Product 360 (and it's extension point) to manage the conversion from business to persistence data types and vice versa. More complex conversions (multi-column to one object etc.) must be done in the corresponding mediator implementation.
- documentation
This attribute can be used to document the repository content. Providing meaningful explanations about the field type is recommended since the attribute might be used in the repository editor

someday. Besides this, if this attribute contains "ESCAPE_VALUES", then semicolons are escaped for this field type (therefor the class-name must be "java.lang.String" and upperBound != 1).

- **fragment-column-access**
The physical access path defines the table and the column in the database, and can additionally define a parameter name for the mass-data access. The syntax is: <Table>.<Column>#<ParameterName><Table>= the table in the database, case sensitive, no escape characters like '[' or ' "' ' <Column>= The column of the field in the table, case sensitive, no escape characters like '[' or ' "' ' <ParameterName>=The optional parameter name for generic data access, if omitted, the column name is used for parameter name.
- **help-context**
The help context holds the path to the help page which is responsible for the field. Currently the help context is not used.
- **identifier**
The unique identifier of the field type. All identifiers in the repository must be unique over the whole document. The naming convention for field type identifiers is:
<EntityType>.<Fieldname> (ArticleType.Ean, ArticleLangType.DescriptionShort)
- **inactive**
The property defines if the field type is inactive for this repository. This flag has been added to provide a single point of enabling or disabling reserve fields of the repository. If set to true, anyone who tries to access the field will get an exception, since the field has not been added to the business model at all. This attribute is an exception of the general "do not edit the types area" rule, since in customizing you need to change this to false in case you want to use a reserved field.
- **internal**
Is just a hint to those who want to edit the types area, that if they change this field, nothing is gonna work any more. However, you should never change the types area unless prior consultation with Informatica.
- **lower-bound and upper-bound**
Defines the cardinality of a field. Normally fields are 0 to 1 or 1 to 1, seldom 1 to many (ArticleLangType.Keywords for example). A lower-bound of 0 means the field is optional for the entity, 1 means it is mandatory. An upper-bound of 1 means the field is a single value, where as -1 means, the field is a list of values. (The business model returns a List interface in this case!) It is also possible to define lower-bound = 0 and upper-bound = 5 which means that this is an optional multi-value field which is not allowed to have more then 5 values.
- **max-length and min-length (mandatory for Strings)**
Only used for strings. Defines the minimum and maximum length of the String. In the types area the database limitations must be used. Settings can be overwritten in the custom area. (For unlimited use -1 (and only -1!)). Default is 0, so be sure to enter here the correct values from the database.
- **object-name**
To access a field in the generic business model, you need an accessor, or property name. The object-name is exactly that. For example, to get the EAN number of an article you would say: article.get("ean"). The string "ean" comes from the object-name of the ArticleType.Ean field type.
- **persistence-class-name**
Optional property, if not set, the value from class name is used. Can be used to define the persistence data type of a field. See class-name (above) for details.
- **persistence-model-class**
The class name of the persistence model. If the persistence model is not set for the field, the persistence model class name of the parent entity type is used. Currently this information is needed to find the corresponding hibernate mapping for the field.
- **persistence-xpath**
Needed to find the field in the persistence hibernate object model in a generic way, used mainly for the generic search.
- **physical-column-is-big**
Set this attribute to true, if the column needs to be treated as BigStrings. This is needed for the unified data access, where for example normal strings are treated differently then ntext (or nclob) strings. Thus, if you have such an ntext (or nclob) field in the database, define here true. Default is false.

- proxy-transition-entity-type
If the field type defines an entity proxy, set here which entity type this proxy represents. Currently used only by the export.
- qualified-serialization
True if the value of this field should be serialized in EMF with it's class name. Used for very generic fields which have only an interface defined in the repository and therefore can't be really serialized with the standard mechanisms. Do not set fields to true without a proper reason since this slows the EMF serialization down.
- range-min and range-max
Only relevant for numeric data types, not for strings. Optional except for BigDecimals because those have no min and max ranges by themselves. If omitted, the minimum and maximum range of the corresponding data type should be taken by the client of the repository. (The repository isn't doing any "default-logic" here, because it has no idea of the data type which is used (which can also be a self-made class!)).
- scale
The precision of decimal data types. Defines the number of digits after the comma.
- searchable
True if the generic search functionalities can be used to search for this fields content, false if not. Default is false. Attention: Do not change this setting, unless you know what to do. Otherwise unexpected exceptions in the search could occur (usually we enable all fields which are searchable, if they are disabled for the search this has a reason!).

6.4.3 Custom Part

The custom part of the repository is especially designed for "customizing". Consultants or system administrators (with special training) can configure Product 360 according to its entities and fields. The custom part relies on the types part as being there and correctly defined. It is something like a view on the types part.

Information: If something is changed in the custom part, it can be treated as "customization" without the need to code something.

6.4.3.1 Entities

Entities directly correspond to entity types in a many to one relationship (multiple entities can correspond to the same entity type)

Multiple Entities for the same Entity Type

Top-level entities, and sub-entities can be defined multiple times for the same entity type. This helps to remove some of the complexity of the system from the user. (See logical-keys for details). Creating additional top level or sub level entities is a task which should not be done by untrained people as you can seriously damage the system if done wrong. Please contact Informatica Professional Services or Support for assistance.

Top-Level Entities

Some top level entities can be used multiple times, most of them can't. Generic Data and Enhanced Generic Data can be used multiple times. However, this must be done carefully of course. Depending on the entity

there are fields which contain some kind of "classifier" or "entityId" which need to be set to a fixed value in order to distinguish instances of the entity type from each other.

Sub-Level Entities

Although entities might be defined multiple times, the general hierarchy of entities must be the same as defined in the types area. For example, it is not possible, to define an entity which is based on the ArticlePriceValueType outside of an entity which is based on ArticlePriceType.

Many sub-level entities do have an entityId logical-key/field which makes it quite easy to use them multiple times. The entityId needs to be unique in the whole repository! The entity id logical key can be treated like any other logical key, besides that this one is never editable, and thus has always a fixed value in the repository. Same goes for it's accompanying field. That must also be not editable and must have the same entityId as the logical key and the entityId attribute on the Entity object itself.

Entities are actually "only" some kind of filter on the list of same entity types. For example in the types area there is an entity type named "ArticlePrice". This entity type holds all price records, no matter if it's a sales price or purchase price. In the custom part, we do have two sub-entities, one for sales and one for purchase prices. The major rule for having multiple sub-entities based on the same entity type is, that the possible qualifications must not overlap. This means, a single record of ArticlePriceType must always be uniquely assignable to a single entity. In our example we do have at least one major difference. The price type enumeration. Sales prices have different price types as purchase prices and the keys of those enumerations do not overlap. So there is no sales price type with the same ID as a purchase price type. This alone would be enough to separate all purchase from all sales prices.

By introducing the "entityId" we made this whole process a lot easier since the entityId logical key works as this "distinguisher". Using the entityId we always instantly know to which entity a record belongs. But for sub-entities which don't have an entityId logical key, you must make sure that *"the possible combination of qualification values (logical keys) of one entity, must never overlap the possible combinations of another entity"*.

Having multiple entities for the same entity type enables you to reuse the same fields over and over again. But be careful - with every additional entity you will essentially add another record in a database table. This might result in very huge tables. We usually recommend to distribute your field requirements across the datamodel and not overuse a single entity type.

The attributes of entities:

- active
Defines if this entity is active at all. In case this is set to false, the entity will not be available at runtime at all. This setting is useful for entities which are not available by standard, but need to be accessible during automated testing etc. Default is true. Attention: Do not set entities to inactive if they are active in the standard repository, this might lead to exceptions since available functionality might rely on the entity being available.
- calc-source-field
Deprecated and currently not used.
- cloneable
Defines if this entity supports the "clone" functionality of the dataset creation feature. This setting is not intended to be changed by anyone except the core development since the clone functionality might need additional business logic which must be provided before the feature can work. Default it's false.
- default-view-id
Only applicable for top-level entities. Defines the view id of the view which should be opened in case entity

items of this entity should be visualized in a generic way. E.g. when the user opens a task for a specific item list, this view id is being used to visualize the items. (Usually a table view)

- **description**
Language specific description of the entity. See section for multi-language support of the repository.
- **documentation**
This attribute can be used to document the repository content, it is not being used by any logic in Product 360 currently. Providing meaningful explanations about the entity is recommended since the attribute might be used in the repository editor someday.
- **entity id**
Each repository entity needs a unique entity id.
- **entity-type**
Defines the corresponding entity type to which this entity is mapped.
- **export-purpose**
Defines if the entity is available for the export. Specialized bit flags apply here, see export documentation for this.
- **help-context**
Overwrites the help-context setting of the corresponding entity type. See entity type section for details. Currently not used.
- **identifier**
The unique identifier of the entity. All identifiers in the repository must be unique over the whole repository. Naming convention: EntityTypename (without the Type suffix). Like: Article for an entity which is mapped to ArticleType, etc.
- **import-purpose**
Defines if the entity is available for the import.
0 = not available
1 = available
- **label-pattern-short, label-pattern-long, label-pattern-description**
Optional and only applicable for top-level entities. These patterns define how entity items should be identified in different contexts (short, long, description). It is used by the GenericEntityItemLabelProvider and can be modified by consultants in order to provide a different visualization for certain entity items. Implementers are encouraged to use the EntityItemLabelProvider mechanisms to obtain labels for entity items in a generic way. The pattern consists of field identifiers in curly brackets combined with any other sign. Note that only fields with no, or a single language logical key are supported currently. If you need a more intelligent logic of label creation feel free to contribute a specialized EntityItemLabelProvider implementation to the corresponding extension point.
- **lowerBound**
It's overwriting the corresponding setting of the entity type if it's more specific. It is possible to make a entity "mandatory" which is "optional" in the types area. You can also define how many elements of an sub-entity is necessary. It's not possible to define a field as "optional" whereas the entity type defines it as mandatory (optional is 0, mandatory is 1).
- **mergeable**
Defines if the (sub-)entity is available for the merge process. This setting must not be changed outside the core development and is only relevant for the supported top-level entities. Currently supported is only the Article entity (the only one for which a merge is available).
- **name**
Language specific name of the entity. See section for multi-language support of the repository.
- **purpose (deprecated)**
Defines if the entity is available for the direct import.
0 = not available
1 = available
- **supports-acl**
Only applicable for top-level entities. Flag which indicates if items of this entity are securable through the object right feature. This attribute must not be changed outside the core development.

- **supports-tasks**
Only applicable for top-level entities. Flag which indicates if this entity should be used for the task management in Product 360. Usually all entities which make sense for tasks are activated by default.
- **upperBound**
It's overwriting the corresponding setting of the entity type if it's more specific. It is possible to limit the maximum number of objects by setting a number here (-1 = unlimited amount of objects). It's not possible to define an entity as "not multiple" whereas the entity type defines it as multiple (multiple is anything else than 0 or 1).
- **whiteboard-enabled**
Currently deprecated and should be ignored.
- **shortIdentifier**
The short identifier is used in canonical forms of an entity item like for EntityItemChangeDocuments or EntityItemDocuments which are both json structures used in AuditTrail or the ObjectAPI. The short identifier must be unique within it's parent entity in a case insensitive way. ("Name" and "name" are equal regarding the short identifier!).

An entity without a short identifier will not be saved in audit trail or returned in the Object API. For custom entities we recommend to adjust the shortIdentifier accordingly, a meaningful but literally *short* one is the best. Keep in mind that it's always relative to the parent! Don't: `articleLogisticExtension`, Do:

`logisticExtension` and the parent of that one is already `article` !

The short identifier is mandatory for entities.

Allowed Characters:

- a-z
- A-Z
- 0-9
- - (minus)
- _ (underscore, but not as first letter!)



The "shortIdentifier" must not start with _(underscore). Although we support upper/lower case characters they are basically for readability.

Entity id

Each entity must have an entity id. The range that can be used for non-standard repository entities: 20002 - 32768

See Assignment of Entity IDs

Export Purpose

With the export purpose attribute you can configure the behaviour of the export context menu. It is used like a bit mask on which you can modify different behaviors with a single value. Therefore you can combine any of the following values (except 0 of course):

Behavior	Purpose Value
Not available for export	0

Behavior	Purpose Value
Available for export	1
Available for preview	2
Available for supplier exchange	4
Available for immediate export	8
Available as sort field	16
Available as field for deleted data	32

Example:

0 - entity is not available for export (default value)

1 - entity is available for export

3 - (= 1 + 2) entity is available for export and the context menu preview is shown

5 - (= 1 + 4) entity is available for export and the context menus for supplier exchange are shown

7 - (= 1 + 2 + 4) entity is available for export and context menus preview and supplier exchange are shown

Entity Param

Generic list of key/value parameters which can be defined for every entity. These parameters can also be used by customizing as the values can be interpreted at runtime. The following list of parameters are supported currently:

Key	Value	Description
entity.report.hasOwnGroup	true/false	Deprecated, has currently no effect
transferTypeName	Drag and Drop Transfer name	Defines which transfer type to use for this entity (overrides default logic)

Logical Key

As described in the types area, logical keys define the dimensions of an entity. Some of our entities are very complicated regarding dimensions (for example the prices). So, we found a way to remove this complexity from the user. We can do this by predefining some or even all of the dimensions right in the repository custom part. With predefined logical keys, there is no need for the user to define a dimension every time he wants to use a field from this entity. This predefinition can either be done in a "default value" way, or in a "fixed value" way, depending on the editable flag of the logical key. Attention: An entity must have all the logical keys which are defined in its corresponding entity type. It is not allowed to just "skip" some of them to remove a dimension. Dimensions can only be "removed" by predefining them!



Some logical keys in the repository are based on a logical key type with has a reference to another entity as value, e.g. **ArticleTradingType.LK.PartyMS**, which references a **Party** object. If such a logical key is defined with a "fixed value", it is necessary, that the the referenced object (e.g. Party) with the ID defined as "fixed value" is also existing in the database. Therefor an update script has to be implemented, that inserts the corresponding object into the database (e.g. the Party table).

The attributes of logical keys:

- description
language specific description of the logical key. See section for multi-language support of the repository.
- documentation
This attribute can be used to document the repository content, it is not being used by any logic in Product 360 currently. Providing meaningful explanations about the entity is recommended since the attribute might be used in the repository editor someday.
- editable
Defines if the value of the logical key can be edited by the user or not. If a logical-key has a predefined value, and is not editable, this logical key will be removed from any visible list in the application – thus this dimension is effectively "removed" (or hidden) from the user. It is not allowed to set a logical key to editable=false without having a valid value defined!
- enumeration
The enumeration identifier from which this logical key should take the possible values
- logical-key-type
Defines to which logical key type this logical key is mapped. You should never change this attribute.
- name
Language specific name of the logical key. See section for multi-language support of the repository.
- proposal-enum
Defines if the value of the logical key can come from an enumeration. This means that the enumeration is available to make it easy for the user, but its not necessary to choose a value from this enumeration, it's just a proposal.
- purpose
This flag indicates if the logical key is available for the import. Some logical keys can't be used in the import – a specialized import logic is contributed for those cases. 0 disables the logical key for the import, any other value enables it. Modification of this attribute is not recommended.
- qualification-permission-identifiers
Some sub-entities have logical keys which reflect the same thing; e.g. the language dimension is used in the ArticleLangType and the ArticleAttributeLangType sub-entities – in both cases it is a list of possible languages. The qualification permission identifier groups those different logical keys for the qualified field

permissions. See the Product 360 User manual for details on the qualified field permissions functionality. Modification of this attribute is not recommended.

- value

If the logical key should be predefined (either as default or as fixed value), the value needs to be entered here in ISO String coding. (dates as YYYY-MM-DD, numbers with dot for comma separation, true/false for booleans). For dates also the code "now" is allowed, which leads to always be "the current date".

- valuePath

There is no need to define any value for this attribute in the repository editor. It is an attribute which is used in runtime to define paths to logical keys which apply to the corresponding context. For example an XPath in case of the XML Import, or some global variable for the export etc. Evaluation of this valuePath is done by the context in which it is used.

- shortIdentifier

The short identifier is used in canonical forms of an entity item like for EntityItemChangeDocuments or EntityItemDocuments which are both json structures used in AuditTrail or the ObjectAPI. The short identifier must be unique within it's parent entity in a case insensitive way. ("Name" and "name" are equal regarding the short identifier!).

An editable logical key without a short identifier is an invalid configuration. For custom entities we recommend to adjust the shortIdentifier accordingly, a meaningful but literally *short* one is the best. Keep in mind that it's always relative to the parent! Don't: `articleLangLanguage` , Do: `language`

The short identifier is mandatory for logical keys. It is supposed to have the same short identifier as the corresponding field.

Allowed Characters:

- a-z
- A-Z
- 0-9
- - (minus)
- _ (underscore, but not as first letter!)



The "shortIdentifier" must not start with _(underscore). Although we support upper/lower case characters they are basically for readability.

Field

Fields hold the most properties in the whole repository. The fields in the custom part define the visualization aspects of fields. And also provide properties to overwrite settings from the types part. Attention: It is not allowed to have two fields which are mapped to the same field type inside the same entity!

The attributes of fields:

- active
Defines if this field is active at all. In case this is set to false, the field will not be available at runtime. This setting is useful for fields which are not available by standard, but need to be accessible during automated testing etc. Default is true. Attention: Do not set fields to inactive if they are active in the standard repository, this might lead to exceptions since available functionality might rely on the field being available.
- average-length
The average length of this field in "number of characters". This is needed for the standard table layout when adding a field to it.
- category
The category this field belongs to.

- **cloneable**
Defines if this field supports the "clone" functionality of the dataset creation feature. This setting is not intended to be changed by anyone except the core development since the clone functionality might need additional business logic which must be provided before the feature can work. Default it's false.
- **default-column-order**
Defines the column order of fields when the application is started the first time in detail views. After this the order defined by the user is persisted. In case multiple fields have the same default column order, the order is not defined.
- **default-column-order-from-top**
Defines the column order of fields when the application is started the first time in top-level views. After this the order defined by the user is persisted. In case multiple fields have the same default column order, the order is not defined.
- **description**
Language specific description of the field. See section for multi-language support of the repository.
- **display-by-default**
Defines if this field should be shown when the application is started the first time in detail views.
- **display-by-default-from-top**
Defines if this field should be shown when the application is started the first time in top-level views.
- **documentation**
This attribute can be used to document the repository content, it is not being used by any logic in Product 360 currently. Providing meaningful explanations about the entity is recommended since the attribute might be used in the repository editor someday.
- **editable**
Defines if this field can be edited or not. This setting will only be evaluated by the GUI, it is still possible to change the field programatically.
- **enumeration**
The enumeration identifier from which this field should take the possible values.
- **export-purpose**
Defines if the field is available for the export. Specialized bit flags apply here, see Export purpose flag. Modification of this attribute is not recommended.
- **field-type**
The corresponding field type of the field. It is not allowed to change this attribute.
- **help-context**
Overwrites the help context setting of the corresponding field type. See field type section for details.
- **identifier**
The unique identifier of the field. All identifiers in the repository must be unique over the whole repository. Naming convention: <EntityName>.<Identifier> (for example. Article.Ean, or ArticleLang.DescriptionShort) It is not allowed to modify the identifier at all! Field identifiers are persisted in workspaces and also in the database, changing of a field identifier breaks the backward compatibility of the repository. Valid characters for a field identifier are letters, numbers, ".", "_", "-" and "-".
- **import-purpose**
Defines if the field is available for the import.
0 = field is not available in import
1 = field is available in import
- **lower-bound**
Overwrites the corresponding setting of the field type. Optional of course. It's not possible to define a field as "optional" whereas the field type defines it as mandatory (optional is 0, mandatory is 1). But you can make a field mandatory which is optional in the types area. For 1:n fields like the article keywords, you can also say for example "at least 3 keywords needed".
- **min-length and max-length**
Overwrites the min-length and max-length settings from the field type. It is not possible to exceed the limitations of the field type. See field type section for details.

- **mergeable**
Defines if this field can be merged from the supplier to the master catalog, if false, the field does not appear in the merge profile. (Only applies for entites which will be merged in general)
- **multiline**
Defines if carriage return line feeds are allowed in the string, also adjusts the visualization of the field. (Making it a multi line edit control for example)
- **name**
Language specific name of the field. The name should be somewhat unique corresponding to its entity, its used for detail views for example. (See section for multi-language support of the repository. Names can reference logical key values, see appropriate section below for details.)
- **name-from-top**
Language specific name of the field for top-level based views. Like an article view for example. Since these views show fields "flattened", a more or less "unique name" is required. (Doesn't make much sense to have 5 times "Amount" in an Article Table). (See section for multi-language support of the repository. Names can reference logical key values, see appropriate section below for details.)
- **password**
Defines if this field is a password field, which should not be visualized in clear-form, but with stars (*) for example.
- **picture-clause**
An optional picture clause which overwrites the default picture clauses which come from the corresponding datatype. See the picture clause section for details.
- **priority**
Deprecated and is currently not used.
- **proposal-enum**
Defines if the value of the field can come from an enumeration. This means that the enumeration is available to make it easy for the user, but its not necessary to choose a value from this enumeration, it's just a proposal.
- **proxy-transition-entity**
Defines the entity for which the field represents a proxy for. Some similar setting is also defined in the types area, for the custom area we need this especially for "field selection dialogs etc".
- **purpose**
Deprecated, make sure that the purpose has the same value as the import-purpose for compatibility reasons.
- **qualification-permission-identifier**
Identifier of the qualified field permission which defines user specific security guidelines for the fields content.
- **range-min and range-max**
Overwrites the range-min and range-max settings from the corresponding field type. See it's description for details. Attention: It is not allowed to exceed the ranges which are defined in the types area!
- **richtext**
Defines if this field can hold formatting markers. It's needed to adjust the visualization of the field (by adding formatting functionality).
- **scale**
Overwrites the setting of the field type. See field type section for details.
- **searchable**
Defines if this field can be used for the search functionalities. Overwrites the field type setting. You cannot set a field to be searchable, when the types part defines it as not searchable. Just the other way round.
- **upper-bound**
Overwrites the corresponding setting of the field type. Optional of course. It's not possible to define a field as "not multiple" whereas the field type defines it as multiple (multiple is anything else then 0 or 1). But you can limit the maximum number of entries by setting this number here. You can also say for example "at least 3 keywords needed (lower-bound), but not more then 10 (upper-bound)".

- **validation-severity-enum**
Defines how a validation problem should be treated. In general, validation always returns an error if the field type's settings are violated. If the field's settings are violated (custom part), The validation returns the severity which is defined here. Choose from INFO, WARNING or ERROR.
- **validation-severity-lower-bound**
This setting applies if the field is a multiple value field (like keywords), and the minimum number of entries is not reached.
- **validation-severity-range**
This setting applies if the value of strings violate the min/max-length, and the value of numbers violates the range-min/range-max.
- **validation-severity-upper-bound**
This setting applies if the field is a multiple value field (like keywords), and the maximum number of entries is exceeded.
- **value**
With the value property, one can define a default value for the field. Depending on the editable property, the value is to be treated as a "fixed" value, or as a "default" value. The value needs to be entered here in ISO String coding. (dates as YYYY-MM-DD, numbers with dot for comma separation, true/false for booleans).
- **visible**
Defines if the field is visible in general. If this is set to false, the field won't be seen in any view.
- **visible-from-top**
Defines if this field is visible in the top-level view (like the Article Table). The visible property takes precedence, so when visible is false, visible-from-top is also false.
- **shortIdentifier**
The short identifier is used in canonical forms of an entity item like for EntityItemChangeDocuments or EntityItemDocuments which are both json structures used in AuditTrail or the ObjectAPI. The short identifier must be unique within it's parent entity in a case insensitive way. ("Name" and "name" are equal regarding the short identifier!).

A field without a short identifier will not be saved in audit trail or returned in the Object API.

For custom entities we recommend to adjust the shortIdentifier accordingly, a meaningful but literally *short* one is the best. Keep in mind that it's always relative to the parent! Don't:

`articleLogisticExtensionMyCustomField` , Do: `myCustomField` as the parent entity is already called `logisticExtension` and the parent of that one is already `article` !

The short identifier is mandatory for fields.

Allowed Characters:

- a-z
- A-Z
- 0-9
- - (minus)
- _ (underscore, but not as first letter!)



The "shortIdentifier" must not start with _(underscore). Although we support upper/lower case characters they are basically for readability.

Export Purpose

With the export purpose attribute you can adjust the behaviour of the field for the export. It is used like a bit mask on which you can modify different behaviors with a single value. Therefore you can combine any of the following values (except 0 of course):

Behavior	Purpose Value
Not available for export	0
Available for export	1
Mandatory field in export	2
Only available in sub module	4
Not available in sub module	8
Available as sort field	16
Available as field for deleted data	32

Example: A field which is available for export, must contain a value (mandatory), can be sorted by and can deliver deleted values would have 51 as export purpose. (1 + 2 + 16 + 32)

Attention: Use the flags "4" and "8" only if the corresponding data type supports the new isOnlySubDataType property. This flag is currently supported by the following data types:

- DataTypeArticleLangList
- DataTypeArticleMediaAssetList
- DataTypeArticleReferenceList
- DataTypeArticleReferenceUnfilteredList
- DataTypeStructureGroupMediaAssetList

Configure the visibility of a repository field for the export Combine the values of the flags you want to set, e.g.

- 0 - field is not available for export (default value)
- 1 - field is available for export
- 3 - (= 1 + 2) field is available for export and for mandatory fields list
- 5 - (= 1 + 4) field is available for export, but only in sub-modules
- 9 - (= 1 + 8) field is available for export, but only in main modules

Field Params

Key	Value	Description
DataTypeArticleFeatureList.mapping .StructureGroupAttribute	Field Identifier	Mapping to field of StructureGroupAttribute entity --> used in export of article features
showTooltipInWebTable	true or false	Display tooltip for the field in the web tables

Permissions

Element which defines a set of commonly available action rights for the entity. Note: All attributes of the element permissions are deprecated and will be replaced by the sub-element "permission".

The attributes of the permissions element are:

- category
The permission category for the defined permissions
- create
The identifier of the "user can create objects of this entity" action right
- delete
The identifier of the "user can delete objects of this entity" action right
- edit
The identifier of the "user can edit objects of this entity" action right
- read
The identifier of the "user can read objects of this entity" action right
- security-edit
The identifier of the "user can edit the object rights of objects of this entity" action right
- security-read
The identifier of the "user can read the object rights of objects of this entity" action right
- version-release
The identifier of the "user can add objects of this entity to a version" action right
- version-remove
The identifier of the "user can remove objects of this entity from a version" action right

Permission

Defines a sub-element for each permission. This also defines the language dependent naming of the corresponding action right as well as the category or license dependencies. All permissions which are defined in the repository using this sub-entity will automatically be contributed to the application – an additional PermissionProvider implementation is not necessary.

The attributes of the permission element are:

- **category**
The permission category for the defined permission
- **description**
Language specific description of the permission, see section for multi-language support of the repository
- **identifier**
Unique identifier of the permission
- **license-identifier**
Unique identifier of the license this permission depends on. In case the user does not have this license, he automatically also does not have this permission.
- **name**
Language specific name of the permission; see section for multi-language support of the
- **repository type**
The type of permission. Currently supported types are:
 CREATE User can create objects of this entity
 DELETE User can delete objects of this entity
 EDIT User can edit objects of this entity
 READ User can read objects of this entity
 SECURITY_EDIT User can edit the object rights of objects of this entity
 SECURITY_READ User can read the object rights of objects of this entity
 VERSION_RELEASE User can add objects of this entity to a version
 VERSION_REMOVE User can remove objects of this entity from a version
 CLONE User can clone objects of this entity

6.4.3.2 Audit Trail Settings

`AuditTrailSettings` is a child element of Entity which is only relevant for root entities. Please refer to the Audit Trail Configuration page for details in how to configure it.

6.4.3.3 Enumeration

Enumerations define a list of values which can be used by field and logical keys. Fields which are mapped to an enumeration will be visualized differently (most times with a smart input control or combo box) in any part of the application. Additionally not only fields can be mapped to enumerations, but also logical keys. For details see the logical key description.

The attributes of enumerations

- **class-name**
Full qualified class name of the corresponding enumeration class which provides programmatic access to the enumeration. This class needs to implement the `com.heiler.ppm.repository.enumerations.EnumProvider` interface. Use the `StdEnumProvider` for pure repository based enumerations. (This provider uses the repository as data provider.) This attribute will no longer be used. The contribution of the enum provider using the `com.heiler.ppm.repository.core/enumProviders` extension point is recommended.
- **description**
Language specific description of the enumeration. See section for multi-language support of the repository.
- **documentation**
This attribute can be used to document the repository content, it is not being used by any logic in Product 360 currently. Providing meaningful explanations about the enumeration is recommended since the attribute might be used in the repository editor someday.

- **identifier**
The unique identifier of the enumeration. All identifiers in the repository must be unique over the whole repository. Naming convention: Enum.<Identifier>
- **key-class-name**
The business data type of the enumeration keys. This needs to be the same class as the business data type of the fields/logical keys which use this enumeration.
- **name**
Language specific name of the enumeration. See section for multi-language support of the repository.
- **user-specific**
Defines if the enumeration has user-specific content. In this case the enumeration will not be shared for all users, but each user has its own instance. It is absolutely necessary to set this flag to true in case the enumeration content can be secured through object rights or similar user specific functionality. However, it should not be set if it's not really necessary since the memory consumption of the server increases in this case.
- **case-sensitive**
Defines if the enumeration works in the "case sensitive" mode. The default value is <false>. This setting affects the search for a key using a synonym. Should be set to <true> if the enumeration contains similar labels which differ only in the case-sensitive notation.

Enum Entry

Depending on the class which implements the EnumProvider interface, enum entries can be used either to be the "content" of the enumeration (this is the case for the StdEnumProvider), or, for example, the implementing class can use these enum entries for filtering.

Attributes of enum entries:

- **documentation**
This attribute can be used to document the repository content, it is not being used by any logic in Product 360 currently. Providing meaningful explanations about the enum entry is recommended since the attribute might be used in the repository editor someday.
- **external-code**
Optional but unique alphanumeric code which identifies the enumeration entry for 3rd party systems. Usually the official ISO code will be returned here. With a corresponding export function this external code can be used in export templates.
- **key**
Defines the unique key of the enum entry. The string representation of the key will be converted to the key class using the ConvertUtils.
- **key-synonym**
Defines one or more unique synonyms for the enum entry. It is not allowed to defined the same synonym for two different enum entries. The enum synonym will be treated as string.
- **label**
The unique language dependent label of the enum entry. The label must be unique for the enumeration since it is automatically used as a synonym for the entry.

Enum Entry Param

Enumeration entries can be parameterized using enum entry params. How these parameters will be interpreted is up to the enum provider implementation and should be documented by the implementation itself.

- name
The parameter name
- value
The value of the parameter

Enum Param

Enumerations can be parameterized using enum params. How these parameters will be interpreted is up to the enum provider implementation and should be documented by the implementation itself.

- documentation
This attribute can be used to document the repository content, it is not being used by any logic in Product 360 currently. Providing meaningful explanations about the enum param is recommended since the attribute might be used in the repository editor someday.
- name
The parameter name
- value
The value of the parameter

6.4.3.4 Permission Category

Category definition for the entity permissions

The attributes of categories

- identifier
The unique identifier of the category. All identifiers in the repository must be unique over the whole repository. There is no default naming convention for category identifiers.
- name
Language specific name of the category. See section for multi-language support.

6.4.3.5 Category

The user should not be bothered too much with the business-object-model and the hierarchy it defines. A typical Product 360 user is only interested in fields, because fields hold the data he needs to work with. Categories group fields logically together, just the way the customer would group the fields himself. In fact, the categories should be defined together with the customer.

The attributes of categories

- identifier
The unique identifier of the category. All identifiers in the repository must be unique over the whole repository. There is no default naming convention for category identifiers.
- name
Language specific name of the category. See section for multi-language support.
- order
The order in which the category should be visualized.

6.4.4 Multi-language support of the repository

All properties which have support for multi-language are treated the same way. Just add the language dependent property value to the corresponding property file. The language property files of the repository reside in the configuration area just like the Repository.repository file does. The default package is called Repository.properties and contains the German labels. If the consulting wants to overwrite these texts, it can either define a concrete property file which is not supplied already from the standard, or, if the customer does not have a multi-language environment, the texts can directly be changed in the repository xml file. Example: The name of the EAN field is defined as %field.Article.Ean which means, that the actual name of the field should be taken from the properties file. Without the '%' the string from the repository would be used directly.
Naming convention: %Identifier of the object.

6.4.4.1 Referencing on logical keys in field names

For some time there is the possibility of depositing with the name of a field a reference on logical keys. This takes effect by changing the value of a logical key. The field's name will change automatically, too. For example: "Short description (German)" will be "Short description (English)", because the logical key is changed and so the name of the field will change too.

Syntax: {[./]LogicalkeyIdentifier#Default} { ... } The curly brackets surround the whole wild card (they will automatically remove when the reference is solved). ./ is optional and means that the logical key isn't located in this entity, but an entity above. (At this time its not possible to refer two or more levels higher) LogicalKeyIdentifier is the explicit identifier of the logical key from the repository. Default Specifies a language dependent default value for the logical key, if no value could passed from the logical key (if not specified the default value is an empty string).

Examples

- Short description ({ArticleLangType.LK.Language})
Really simple, but could be difficult too, just like the value of an attribute:
- {[./ArticleAttributeType.LK.Name#Attribut]} ({ArticleAttributeValueType.LK.Language})
That's: "Length (German)" or "Width (English)"

Ever after which attribute the user wants, he feeds the name of the attribute in and specifies its language. If something is wrong configured here, the field name contains a corresponding hint like: !Repository is misconfigured, LK-Reference!

6.4.5 Picture-clause syntax

If the picture clause is omitted, the login locale's parameters will be used. Otherwise see the API documentation of the Java Development Kit

- Decimals: java.text.DecimalFormat
- Strings: javax.swing.text.MaskFormatter
- Dates: java.text.SimpleDateFormat (Please consider special week year syntax introduced with Java 7)
- Others: See the corresponding formater implementation which is contributed to the FormatUtils of Product 360.

6.4.6 Repository Manager

The Repository Manager (formerly known as Repository Editor) can be used to adjust the repository.

After you customized the repository to your needs you may want to validate the repository. This can be done either from the main menu 'Repository Editor' → 'Validate' or from the context menu entry 'Validate'. It is important to select the 'Repository' node or a node above.

The executed validations are the same as the ones executed during the server start which might lead to the server not running in case of a misconfiguration of the repository.

6.4.7 Assignment of Entity IDs

In Product360 each entity (including subentities) needs to have an entity id assigned.

The standard implementation uses ids between 1 and 20001. Below you can find an overview of the used id ranges for each root entity. Please don't use entity ids from this range for custom entities because we might use them for new standard entities in the future.

For custom entities please use an entity id between 20002 and 32768. In order to validate that the id hasn't already been used, validate your repository after your changes.

This entity id is stored as a short signed int, which can take positive values from 1 to 32768.



Each new standard entity should get an entity id within the respective id range, new non-standard entities get an id of the ranges defined below.

1 - 20001 (Steps of 1000 resp. 100)

Standard entities

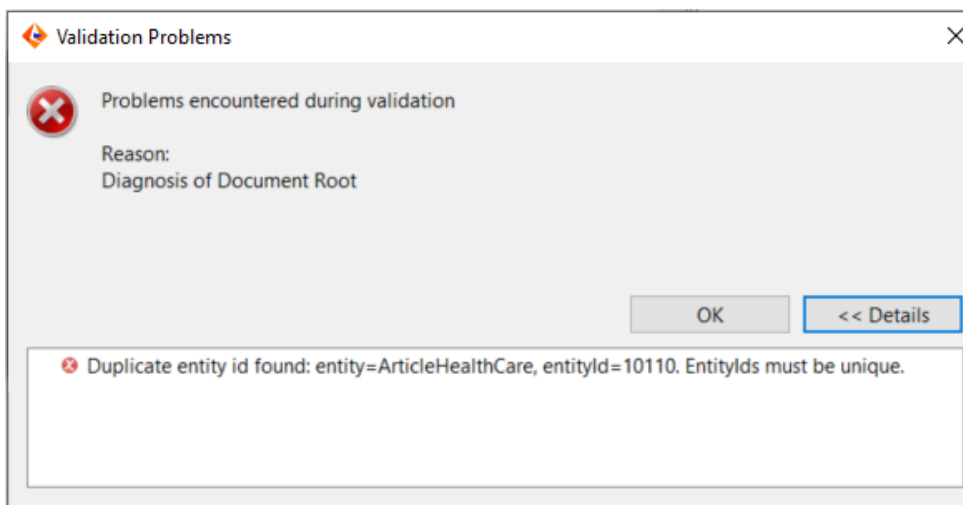
20002 - 32768

Used by custom entities

6.4.7.1 Validation

The 'Validate' function in the Repository Manager includes a check for duplicate entity ids.

Select 'Document Root' or 'Repository' and in the menu click on 'Repository editor' → 'Validate' to start the validation.



6.4.7.2 Entity id overview

EntityIDs currently reserved in standard HPM repository:

Entity	EntityType	EntityID range
Acl	AclType	4400-4401
AclDefaults	GenericDataEntityType	5700
Article	ArticleType	1000-1065; 10000-10131; 17102-17182; 20001
ArticleAssortment	AssortmentType	2100
ArticleSearch	GenericDataEntityType	3700
CatalogStructure	StructureType	2200-2201

Entity	EntityType	EntityID range
Certification	CertificationType	6500-6503
Channel	ChannelType	6800-6804
Characteristic	CharacteristicType	8000-8002
Customer	CustomerType	20000
DaaSProduct	DaaSProductType	1570-1573
DashboardTemplate	GenericDataEntityType	17450
Dictionary	DictionaryType	17000-17002
EntityReportQuery	GenericDataEntityType	5500-5501
ExportProfile	GenericDataEntityType	3900
ExportTemplate	GenericDataEntityType	3800
ImportProfile	GenericDataEntityType	5900-5901
InboxConfiguration	InboxConfigurationType	6200-6202
InboxGroup	InboxGroupType	6400-6401
Ingredient	IngredientType	6900-6930

Entity	EntityType	EntityID range
JobHistory	JobHistoryType	1;5400
KPIExecution	GenericDataEntityType	7500
KPIValue	KPIValueType	7400-7410
Lookup	LookupType	7200-7220
LookupValue	LookupValueType	7300-7340
MailTemplate	MailTemplateType	16100-16101
MasterCatalog	CatalogType	2900-2901
MediaAsset	MediaAssetType	2400-2404
MediaAssetCategory	MediaAssetCategoryType	7600
MediaAssetFile	MediaAssetFileType	2500-2502
Party	PartyType	2800-2801
PriceGroupPurchase	PriceGroupType	3300-3302
PriceGroupSales	PriceGroupType	3400-3402
ProblemLogEntry	ProblemLogEntryType	4500

Entity	EntityType	EntityID range
Product	EGDType	5100-5112
Product2G	ArticleType	1100-1165; 17103
Product2GAssortment	AssortmentType	6000
Product2GSearch	GenericDataEntityType	6100
ProductAssortment	AssortmentType	5200
ProductSearch	GenericDataEntityType	5300
QualityStatus	StatusType	2000-2021
Revision	RevisionType	5600-5602
RichTextFormatMap	GenericDataEntityType	4100
RichTextFormatTemplates	GenericDataEntityType	4000
StandardizationDictionary	DictionaryType	17200-17202
StandardizationValue	DictionaryEntryType	17300-17301
Structure	StructureType	2300-2302
StructureFeature	StructureFeatureType	4300-4303

Entity	EntityType	EntityID range
StructureGroup	StructureGroupType	3000-3011
StructureGroupSearch	GenericDataEntityType	3500
StructureValue	StructureValueType	4200-4202
SupplierCatalog	CatalogType	7000-7003
SystemMessage	GenericDataEntityType	5800-5801
Task	TaskType	3600-3650
TaskNotification	TaskNotificationType	16200
ThirdParty	ThirdPartyType	7100-7102
UiTemplate	GenericDataEntityType	17400
Unit	UnitType	3100-3104
UnitSystem	UnitSystemType	3200-3201
User	UserType	2600-2601
UserGroup	UserGroupType	2700-2702
Variant	ArticleType	1200-1265; 17104

Entity	EntityType	EntityID range
VariantAssortment	AssortmentType	6600
VariantSearch	GenericDataEntityType	6700
Word	DictionaryEntryType	17100-17101
Workflow2G	Workflow2GType	8700-8701

6.4.8 Lookups aka Dynamic Enumerations

Lookups (aka Dynamic Enumerations) can be used within characteristics to define lists of allowed values. In addition to that, this feature can also be used stand-alone to be able to have enumeration values dynamically modified in the application without the need to restart a server.

- [Background information on Enumerations](#) (see page 110)
 - [Key Class](#) (see page 110)
 - [EnumProvider](#) (see page 110)
- [Configure the Repository](#) (see page 111)
 - [Define the Enumeration in the Repository](#) (see page 111)
 - [LookupValueProxy](#) (see page 111)
 - [Long](#) (see page 111)
 - [String](#) (see page 111)
 - [Adjust Field Type](#) (see page 113)
 - [LookupValueProxy](#) (see page 113)
 - [Long or String](#) (see page 114)
 - [Create new Field](#) (see page 114)
- [Create the Lookup](#) (see page 115)
- [Proposal Enumerations](#) (see page 115)

Lookups can be used in combination with Characteristics and also for regular fields. Product 360 always had the concept of enumerations for fields. Every field can be configured to use an enumeration which limits the possible values of this field to the enumeration values.

One of the biggest issues of repository based enumerations is, that the actual enumeration entries are (usually) also stored in the repository. This is the most often used way to provide the values for the enumeration, especially in customizing situations as no coding is required. Unfortunately a modification of the repository requires a server (or cluster) restart which can be an issue for critical deployments and doesn't allow the business users to maintain values for a list independently.

The enumeration logic in Product 360 is prepared for dynamic enumerations which have a different persistence than the repository itself. Many enumerations already work in a dynamic way, like Units, Buyers, Suppliers or Catalogs and so on. The same concept is now used to combine the concept of lookups with enumerations.

6.4.8.1 Background information on Enumerations

Enumerations are defined in the repository's custom area and can be assigned to fields and logical keys. The enumeration in the repository can be seen as the descriptive part of the enumeration. It contains the name of the enumeration, a unique identifier and a description as well as the key class and the enum provider.

Key Class

The key class is the java class which represents the key's datatype of the enumeration entries. Some enumerations use `java.lang.String` so the key is a String, some use `java.lang.Integer` or `java.lang.Long` for numeric keys (like the language enumeration). Others use EntityProxy implementations like the `UnitProxy` or the `PartyProxy` for the Unit or Supplier/Buyer enumeration. The important thing here is that the key class must be equal to the class of the field to which the enumeration is assigned to.

So, if you have a field which is based on a field type which has a `UnitProxy` as class, the enumerations key class must also be a `UnitProxy`. Why is that? Pretty simple, since the key of a chosen enumeration entry will be stored as value for the field. If a user chooses a specific order unit for an item, the chosen order unit's key (which is an instance of `UnitProxy`) will be stored for this item's order unit field. This is the reason why the key-class and the field type's class must be equal in order to use an enumeration on any of the fields which belong to this field type.

As the class of a field type directly modified the business model of the application, it is not allowed to change it for standard fields at all. The business logic which is implemented for standard fields might break which results in critical exceptions. For reserved fields (`Res_*`) there usually is no such business logic so a change of the class is possible here (as long as it's a compatible change regarding the persistence class. You can't switch a `Res_Int_01` field to now use `java.lang.String` as there is no way we could convert the string into an integer. It would be possible the other way round of course).

EnumProvider

Every enumeration in the repository has a corresponding enum provider implementation. The `EnumProvider` interface is used in the application to work with the actual values of an enumeration. It provides access to the keys, labels and external codes of the enumeration entries. The enum provider can either be directly defined in the Enumeration element of the repository or it can be contributed for the enum provider extension point in the application. One or the other must be given for every enumeration and this will be checked during a server start!

So the enum provider is responsible for providing the enumeration entries keys and labels. One very common enum provider is the `StdEnumProvider` which uses the repository `EnumEntry` elements as data source. Other enum providers don't use the repository as data source. For example the enum provider for units or suppliers or catalogs. Those return the values for those entities from our database - as Units, Suppliers and Catalogs are actual Entities in the repository and there is a maintenance UI in the application.

So in order to combine the Enumerations with the Lookups we use special `EnumProvider` implementations which provide seamless access to the lookup values during runtime - just like we have it for Units, Suppliers or Catalogs.

6.4.8.2 Configure the Repository

When using a lookup for a regular field, we still need to adjust the repository, but only for defining which enumeration (lookup) belongs to which field and not for the actual entries of the enumeration any more. Those can be adjusted from within the application's lookup editing perspective.

To give you the maximum flexibility when configuring the repository we actually provided three different enum provider implementations for lookup values. Each of them has a different key class.

Define the Enumeration in the Repository

As described above the Enumeration in the repository is somewhat the descriptive part with identifier, name and description as, very important, the key class and the enum provider. In order to use a lookup for a field, you need to create an Enumeration for this lookup and then use the enumeration for the field.

LookupValueProxy

Here is an example of the custom manufacturers enumeration which uses the LookupValueProxy as key class.

```
<enum identifier="Enum.CustomManufacturers">
  <name>%enum.CustomManufacturers.name</name>
  <description></description>
  <class-name>com.heiler.ppm.lookup.core.enumeration.LookupValueEnumProvider</class-
name>
  <key-class-name>com.heiler.ppm.lookup.core.LookupValueProxy</key-class-name>
  <param name="lookupIdentifier" documentation="">CustomManufacturers</param>
</enum>
```

Long

Example to use it with `java.lang.Long` as key-class:

```
<enum identifier="Enum.CustomManufacturersByID">
  <name>%enum.CustomManufacturers.name</name>
  <description></description>
  <class-
name>com.heiler.ppm.lookup.core.enumeration.LookupValueInternalIDEnumProvider</class-
name>
  <key-class-name>java.lang.Long</key-class-name>
  <param name="lookupIdentifier" documentation="">CustomManufacturers</param>
</enum>
```

String

Example to use it with `java.lang.String` as key-class:

```
<enum identifier="Enum.CustomManufacturersByCode">
  <name>%enum.CustomManufacturers.name</name>
  <description></description>
  <class-name>com.heiler.ppm.lookup.core.enumeration.LookupValueCodeEnumProvider</
class-name>
  <key-class-name>java.lang.String</key-class-name>
  <param name="lookupIdentifier" documentation="">CustomManufacturers</param>
</enum>
```

Important attributes:

- **class-name**

The class name must be an implementation of the EnumProvider interface. Three possible enum providers are delivered for the Lookup integration:

- **com.heiler.ppm.lookup.core.enumeration.LookupValueEnumProvider** which corresponds with `com.heiler.ppm.lookup.core.LookupValueProxy` as key-class. This is the best choice if possible, but the field type needs to be adjusted for it to have `LookupValueProxy` as key class. However, it provides the full functionality including field transition in tables or export.
- **com.heiler.ppm.lookup.core.enumeration.LookupValueInternalIDEnumProvider** which corresponds with `java.lang.Long` as key-class. This is the second best choice. It still allows you to change the lookup value's code once it has been active, but it won't provide field transition in tables or export.
- **com.heiler.ppm.lookup.core.enumeration.LookupValueCodeEnumProvider** which corresponds with `java.lang.String` as key-class. **This should only be used if there is no way around it as in this case the Code of the lookup value will be stored as key. This means that you won't be able to change the code anymore as soon as it has been activated and is potentially being used.** Also field transitions are not supported with this one. This enum provider is actually better used for proposal enumerations.

- **key-class-name**

the class which must be equal to the `class-name` of the field's field type on which we want to use the enumeration (see above for details). Of course it also must correspond with the EnumProvider implementation.

- `com.heiler.ppm.lookup.core.LookupValueProxy`
- `java.lang.Long`
- `java.lang.String`

- **EnumParam "lookupIdentifier"**

Enumerations can have multiple enum params which typically provide additional settings for the enum provider implementations. In order to use a lookup as the data source for an enumeration we need to define which lookup that is. So there needs to be an enum-param with the name **lookupIdentifier** with the identifier of the lookup as value. No worries if this lookup does not yet exist. The application will automatically create it during server start in case it's not yet there.

Adjust Field Type

LookupValueProxy

If you are in the comfortable position to be able to use some `Res_Int` field for your custom field (and this field is not used already!) you can adjust the field type class.

By doing so, you enable the transition features in the table views or export. So you can access any field of the lookup value transparently from within the context of the item.

⚠ Changing the field type is not allowed for standard fields since this may break business logic. It is only allowed for the attributes mentioned here and only for reserved fields!

```
<field-type identifier="ArticleType.Res_Int_01" proxy-transition-entity-type-ref="LookupValueType">
  <object-name>res_Int_01</object-name>
  <class-name>com.heiler.ppm.lookup.core.LookupValueProxy</class-name>
  <persistence-model-class>com.heiler.ppm.article.db.model.ArticleDetail</persistence-model-class>
  <persistence-xpath>/articleRevisions/articleDetail/res_Int_01</persistence-xpath>
  <persistence-class-name>java.lang.Long</persistence-class-name>
  <fragment-column-access>ArticleDetail.Res_Int_01</fragment-column-access>
  <inactive>false</inactive>
  <searchable>true</searchable>
  <range-max></range-max>
</field-type>
```

In our example we use the `ArticleType.Res_Int_01` field type. It's actual datatype is `Long`. You can also use a `String` field for this, but we would recommend to stick with the `Int` fields as long as you can.

- `proxy-transition-entity-type-ref`
to signal that this field now points to objects of the Entity Type " `LookupValueType` "
- `class-name`
to define that we now store values of the `com.heiler.ppm.lookup.core.LookupValueProxy` class in this field (was `java.lang.Long`)
- `persistence-class`
to let the system know in which physical datatype the `LookupValueProxy` needs to be converted to, so this must be `java.lang.Long` (after all, the database column is still a `NUMBER/BigInt`) (was empty which means it's the same as the `class-name`)
- `inactive`
in order to be able to use the reserve field type, you need to set it to `inactive = false`

Long or String

In contrary to the use of the `LookupValueProxy` as key-class the `java.lang.Long` or `java.lang.String` requires no adjustment of the field type besides setting it to **inactive = false**

In our example we use the `ArticleType.Res_Int_01` field type:

```
<field-type identifier="ArticleType.Res_Int_01">
  ...
  <inactive>false</inactive>
  ...
</field-type>
```

- `inactive`
in order to be able to use the reserve field type, you need to set it to `inactive = false`

Create new Field

After adjusting the field type, you can create your custom field. It's important that this custom field points to the beforehand adjusted `FieldType`!

```
<field identifier="Article.CustomManufacturer" category-ref="master-data" enum-ref="Enum.CustomManufacturers" field-type-ref="ArticleType.Res_Int_01" proxy-transition-entity-ref="LookupValue" supports-dataquality="true">
  <name>%field.Article.CustomManufacturer.name</name>
  <name-from-top>%field.Article.CustomManufacturer.nameFromTop</name-from-top>
  <description></description>
  <editable>true</editable>
  <visible>true</visible>
  <visible-from-top>true</visible-from-top>
  <mergeable>true</mergeable>
  [...]
</field>
```

Important attributes:

- `enum-ref`
the identifier of the enumeration we created in the step before (in our example: `Enum.CustomManufacturers`)
- `field-type-ref`
the identifier of the field type we created at the beginning (in our example: `ArticleType.Res_Int_01`)
- `proxy-transition-entity-ref`
The identifier of the root entity to which this field now points (must be **LookupValue**)

⚠ This must only be set in case you used the `LookupValueProxy` as key-class and adjusted the field type accordingly (see above). Otherwise this attribute must not be empty as the field type doesn't support any transition!

6.4.8.3 Create the Lookup

There are generally two ways to create a new lookup. The first, and most obvious one is to create the lookup within the Desktop UI or the Service API. This is pretty straight forward and needs no further explanation here. However, since you can use lookups as enumerations we implemented a migration feature into the startup process of the server as well.

During startup the server scans the repository for all enumeration elements which do have the `lookupIdentifier` enum parameter. For those elements it checks if the lookup for this identifier already exists in the system. If it doesn't, it will use the information provided by the enumeration element to create this new lookup. In addition to that, it will not only create the lookup, but also create lookup values for each `enum-entry`. This is done only in case the lookup is not there yet.

This feature makes it very easy to provide default lookup values. Additionally, you don't need to open the Desktop UI after you configured the steps described above as the server will create the lookup automatically when you restart it. The users can directly start editing values for it.

i This feature can also be used as a migration feature. If you want to migrate a repository based enumeration (whose values are all in the repository as `enum-entries`), you just need to adjust this enumeration as described above. Just adjust the `class-name` and the `key` if necessary. If you do that, you also need to adjust the fields it's used for and so on. As soon as you have the enumeration adjusted you can restart the server and all the exiting entries in the repository will automatically be created as lookup values.

6.4.8.4 Proposal Enumerations

Pretty similar to the configuration of enumerations, you can also define proposal enumerations for `String` fields. Proposal enumerations allow the user to enter a value which is not part of the enumeration. So the enumeration entries merely work as a "proposal" for it. Because of this nature, proposal enumerations are only supported at `String` fields.

Technically it works the same way as with normal enumerations, but you don't set the `"enumeration"` attribute on the field, but the `"proposal-enumeration"` attribute. The field must be `String`, the proposal enumeration's `keyClass` must be `String` too.

For proposal enumerations you use also the `com.heiler.ppm.lookup.core.enumeration.LookupValueCodeEnumProvider` of course

6.4.9 How to register and use a characteristic value provider with parameters

- [Introduction](#) (see page 116)
- [Create Plugin using SDK](#) (see page 116)
 - [Add Extension in MANIFEST.MF](#) (see page 116)
 - [Contribute the characteristic value provider](#) (see page 117)
 - Create class with name: "SampleIntegerCharacteristicValueProviderFactory" in package: "com.heiler.ppm.customizing.sample.valueProvider.integer", implements interface: "CharacteristicValueProviderFactory". (see page 117)
 - Create class with name "SampleIntegerCharacteristicValueProvider" in package "com.heiler.ppm.customizing.sample.valueProvider.integer", implements interface: "CharacteristicValueProvider". (see page 118)
- [Deploy the plugin and restart Product 360 server](#) (see page 120)
- [Attaching value provider to a characteristic](#) (see page 120)
- [Characteristic UI showing the provided values](#) (see page 121)
- [DependentLookup Characteristic](#) (see page 121)
 - [Introduction](#) (see page 121)
- [Setup of Lookup Value Reference](#) (see page 122)
 - [Contribute the country to state dependent characteristics to value provider](#) (see page 122)
- [Characteristic modeling:](#) (see page 129)
 - [Assigning value provider to characteristics in modelling UI](#) (see page 130)
 - [Characteristic UI showing the provided values](#) (see page 130)

6.4.9.1 Introduction

Use the SDK to create new plugin, contribute value provider implementation and add value provider to a characteristic.

6.4.9.2 Create Plugin using SDK

Create new plug-in project similar to "com.heiler.ppm.customizing.sample.valueProvider" and add required dependencies.

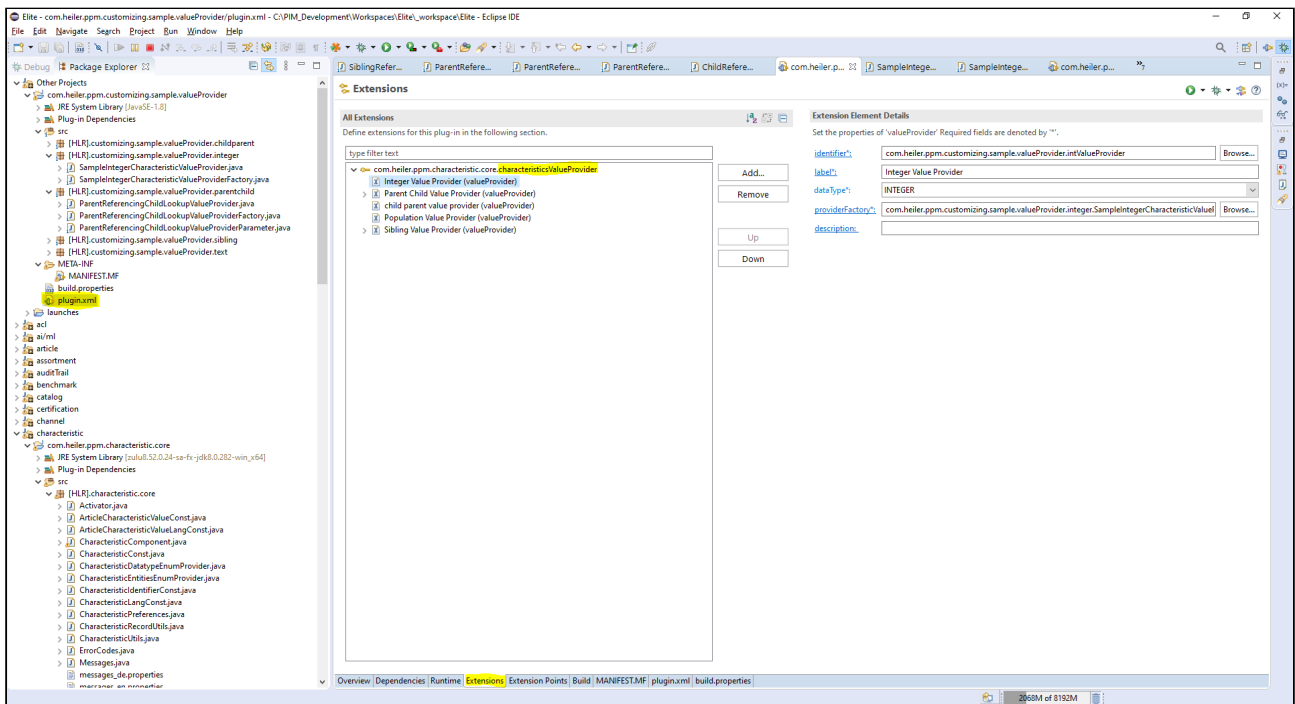
Sample plugin is present under sdk → examples → customizing.

Add Extension in MANIFEST.MF

Open Extensions

add extension for "com.heiler.ppm.charateristic.core.characteristicsValueProvider"

- Add valueProvider
 - Identifier : intgerValueProvider
 - Label : %valueProvider.text (Localized text)
 - dataType : INTEGER
 - providerFactory : SampleIntegerCharacteristicValueProviderFactory



Contribute the characteristic value provider

Create class with name: "SampleIntegerCharacteristicValueProviderFactory" in package: "com.heiler.ppm.customizing.sample.valueProvider.integer", implements interface: "CharacteristicValueProviderFactory".

CharacteristicValueProviderFactory: Creates new instance of CharacteristicValueProvider.

SampleIntegerCharacteristicValueProviderFactory

```

1  import
   com.heiler.ppm.characteristic.core.valueProvider.CharacteristicValueProvider;
2  import
   com.heiler.ppm.characteristic.core.valueProvider.CharacteristicValueProviderFactory;
3  import com.heiler.ppm.std.core.entity.list.EntityItemList;
4
5  public class SampleIntegerCharacteristicValueProviderFactory implements
   CharacteristicValueProviderFactory
6  {
7
8      @Override
9      public CharacteristicValueProvider createValueProvider( EntityItemList
   itemList )

```

```

10    {
11        return new SampleIntegerCharacteristicValueProvider( itemList );
12    }
13
14 }

```

Create class with name "SampleIntegerCharacteristicValueProvider" in package "com.heiler.ppm.customizing.sample.valueProvider.integer", implements interface: "CharacteristicValueProvider".

CharacteristicValueProvider: Provides characteristic values and check for a valid value.

Provide the implementation of interface Value inside the SampleIntegerCharacteristicValueProvider itself.

SampleIntegerCharacteristicValueProvider

```

1  import java.util.ArrayList;
2  import java.util.Collection;
3  import java.util.HashSet;
4  import java.util.List;
5  import java.util.Locale;
6  import java.util.Set;
7
8  import com.heiler.ppm.characteristic.core.record.CharacteristicRecord;
9  import
10
11  com.heiler.ppm.characteristic.core.valueProvider.CharacteristicValueProvid
12  er;
13  import com.heiler.ppm.lookup.core.LookupValue;
14  import com.heiler.ppm.std.core.entity.list.EntityItemList;
15
16  @SuppressWarnings( "unused" )
17  public class SampleIntegerCharacteristicValueProvider implements
18  CharacteristicValueProvider
19  {
20
21      public SampleIntegerCharacteristicValueProvider( EntityItemList
22  itemList )
23      {
24          //
25      }
26
27      @Override
28      public Collection< Value > getValues( LookupValue language,
29  CharacteristicRecord charRecord )
30      {
31          List< Value > values = createValues();
32          return values;
33      }
34
35      @Override

```

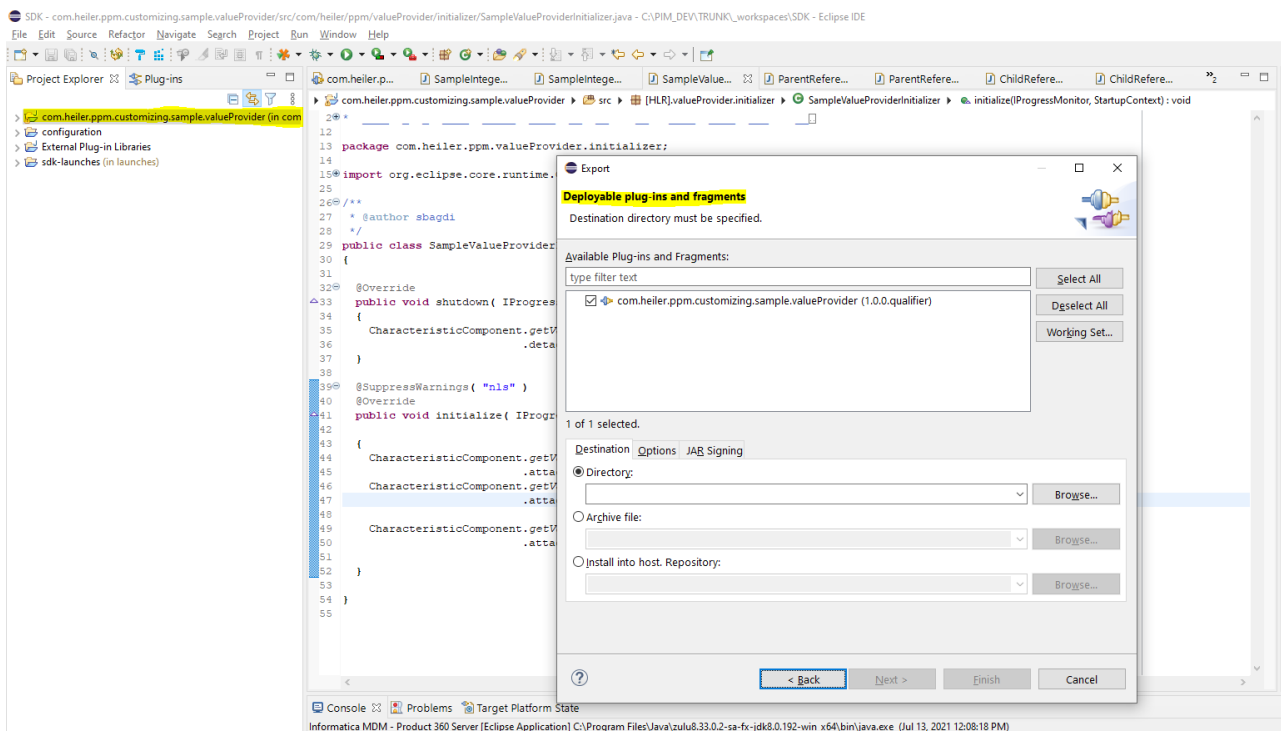
```

30     public boolean containsValue( LookupValue language, CharacteristicRecord
charRecord,
31                                     Object value /*actual charateristic record
value*/ )
32     {
33         // make sure values are of datatype Long for Integer type
characteristics
34         Set< Long > values = new HashSet();
35         values.add( 1024L );
36         values.add( 2048L );
37         values.add( 4096L );
38         return values.contains( value );
39     }
40
41     @SuppressWarnings( "nls" )
42     private List< Value > createValues()
43     {
44         List< Value > values = new ArrayList();
45         // make sure values are of datatype Long for Integer type
characteristics
46         Value v1 = new ValueImpl( 1024L, "1 MB" );
47         Value v2 = new ValueImpl( 2048L, "2 MB" );
48         Value v3 = new ValueImpl( 4096L, "4 MB" );
49         values.add( v1 );
50         values.add( v2 );
51         values.add( v3 );
52         return values;
53     }
54
55     public static class ValueImpl implements Value
56     {
57         private String label;
58         private Object value;
59
60         public ValueImpl( Object value, String label )
61         {
62             this.value = value;
63             this.label = label;
64         }
65     }
66
67     @Override
68     public Object getValue()
69     {
70         return this.value;
71     }
72
73     @Override
74     public String getLabel( Locale locale )
75     {
76         return this.label;
77     }
78 }

```

6.4.9.3 Deploy the plugin and restart Product 360 server

export the plugin as deployable plug-ins and fragments.

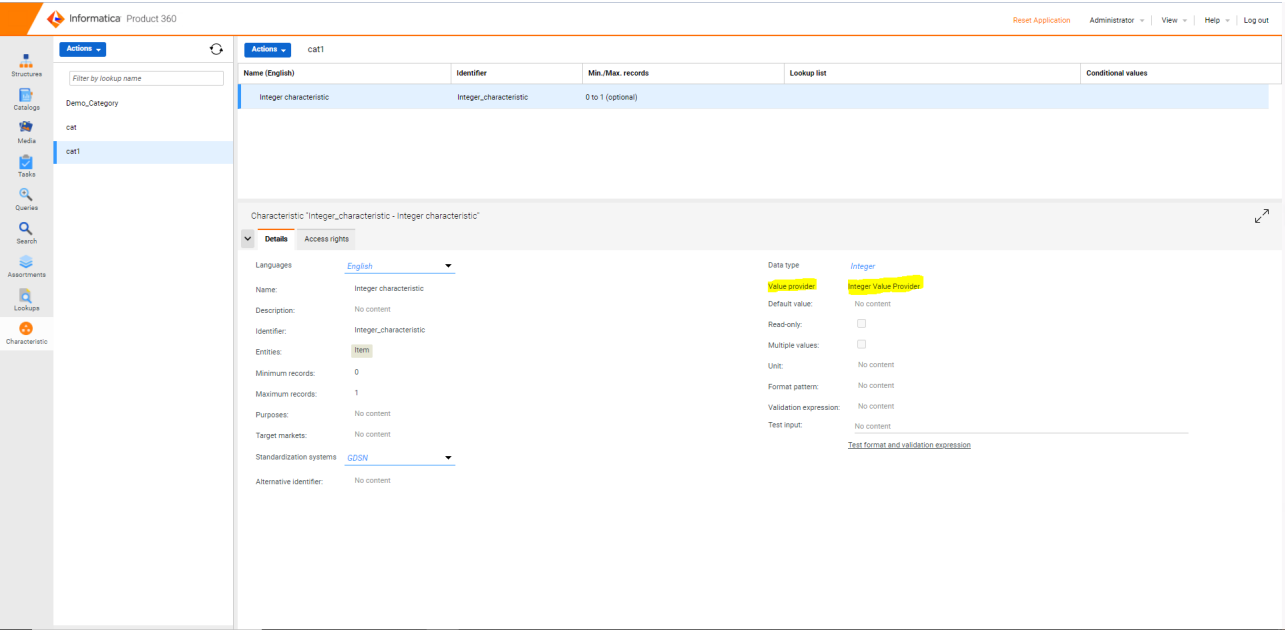


Deploy exported jar inside server → plugins folder

We are good to start the server

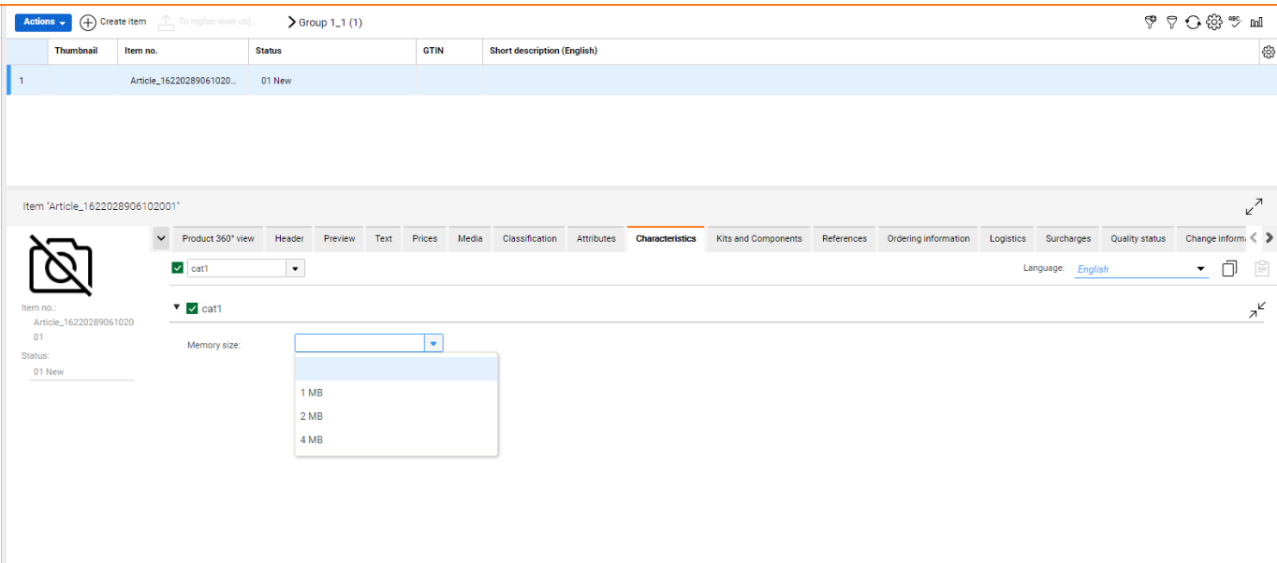
6.4.9.4 Attaching value provider to a characteristic

Characteristic value provider can be attached In characteristic modeling detail view.



6.4.9.5 Characteristic UI showing the provided values

Provided values will be available to select in the dropdown.



6.4.9.6 DependentLookup Characteristic

Introduction

Create lookups lookup values

- Country

- State

6.4.9.7 Setup of Lookup Value Reference

referencing lookup	referenced lookup	business case
Country	State	A country can have multiple states

Navigate to lookup context view

Add the referenced values for desired lookup value.

There are two ways you can add referenced values for a lookup value:

- Include values table : Only selected values can be referenced.
- Exclude values table : All the values can be referenced except selected values.

The screenshot displays the Informatica Product 360 interface. On the left, a sidebar contains navigation icons for Structures, Catalogs, Media, Tasks, Queries, Search, Assortments, Lookups, and Characteristic. The main area shows the 'Country' lookup context. A table lists countries with columns: MIME A, Code, Name (English), Description (English), and Active. The rows are: 1. can (Canada), 2. ger (Germany), 3. ind (India), and 4. usa (United States of America). The 'usa' row is highlighted. Below this, the 'Lookup value details' section for 'usa - United States of America' is shown. It includes a 'Referenced values' table with columns: Lookup list, MIME Attachment, Code, Name, Active, and Description. The referenced values are: 1. State (california, California, Yes) and 2. State (texas, Texas, Yes). There are also sections for 'Included values (0)' and 'Excluded values (0)', both currently empty.

Refer Importing lookups and lookup value references documentation for importing lookup references.

Contribute the country to state dependent characteristics to value provider

class name: "ParentReferencingChildLookupValueProviderFactory" package:
 "com.heiler.ppm.customizing.sample.valueProvider.parentchild" interface:
 "CharacteristicValueProviderFactory".

ParentReferencingChildLookupValueProviderFactory

```

1  package com.heiler.ppm.customizing.sample.valueProvider.parentchild;
2
3  import
4
5  com.heiler.ppm.characteristic.core.valueProvider.CharacteristicValueProvid
6  er;
7  import
8
9  com.heiler.ppm.characteristic.core.valueProvider.CharacteristicValueProvid
10 erFactory;
11 import com.heiler.ppm.std.core.entity.list.EntityItemList;
12
13 public class ParentReferencingChildLookupValueProviderFactory implements
14 CharacteristicValueProviderFactory
15 {
16
17     @Override
18     public CharacteristicValueProvider createValueProvider( EntityItemList
19 itemList )
20     {
21         return new ParentReferencingChildLookupValueProvider( itemList );
22     }
23 }

```

class name: "ParentReferencingChildLookupValueProvider" package:
 "com.heiler.ppm.customizing.sample.valueProvider.parentchild" interface: "CharacteristicValueProvider".

ParentReferencingChildLookupValueProvider

```

1  package com.heiler.ppm.customizing.sample.valueProvider.parentchild;
2
3  import java.util.Collection;
4  import java.util.HashSet;
5  import java.util.Locale;
6  import java.util.Set;
7  import java.util.stream.Collectors;
8
9  import org.apache.commons.logging.Log;
10 import org.apache.commons.logging.LogFactory;
11 import org.eclipse.core.runtime.CoreException;
12
13 import com.google.gson.JsonElement;
14 import com.google.gson.JsonObject;
15 import com.google.gson.JsonParser;
16 import com.heiler.ppm.characteristic.core.record.CharacteristicRecord;

```

```

17  import

com.heiler.ppm.characteristic.core.valueProvider.CharacteristicValueProvid
er;
18  import com.heiler.ppm.commons.RuntimeCoreException;
19  import com.heiler.ppm.lookup.core.Lookup;
20  import com.heiler.ppm.lookup.core.LookupValue;
21  import com.heiler.ppm.security.core.SecurityComponent;
22  import com.heiler.ppm.std.core.entity.list.EntityItemList;
23
24  public class ParentReferencingChildLookupValueProvider implements
CharacteristicValueProvider
25  {
26
27      private Locale          locale;
28      private static final String PARENT_CHARACTERISTIC =
"parentCharacteristic";                                     //
$NON-NLS-1$
29      private static final Log    LOG                =
LogFactory.getLog( ParentReferencingChildLookupValueProvider.class );
30
31      public ParentReferencingChildLookupValueProvider( @SuppressWarnings(
"unused" ) EntityItemList itemList )
32      {
33          this.locale = SecurityComponent.getImpersonator()
34                                  .getLoginToken()
35                                  .getLocale();
36      }
37
38      @Override
39      public Collection getValues( LookupValue language, CharacteristicRecord
characteristicRecord )
40      {
41          Set< LookupValue > dependentLookupValues =
fetchDependentLookupValues( language, characteristicRecord );
42          return dependentLookupValues.stream()
43                                  .map( lookupValue -> new
ValueImpl( lookupValue, lookupValue.getName( this.locale ) ) )
44                                  .collect( Collectors.toList() );
45      }
46
47      @Override
48      public boolean containsValue( LookupValue language, CharacteristicRecord
characteristicRecord, Object value )
49      {
50          Set< LookupValue > dependentLookupValues =
fetchDependentLookupValues( language, characteristicRecord );
51          return dependentLookupValues.contains( value );
52      }
53
54      private Set< LookupValue > fetchDependentLookupValues( LookupValue
language,

```



```

55 CharacteristicRecord characteristicRecord )
56 {
57     Lookup lookup = characteristicRecord.getCharacteristic()
58                                     .getLookup();
59     CharacteristicRecord parent =
60     getParentCharacteristicRecord( characteristicRecord );
61     Set< LookupValue > values = new HashSet();
62     if ( parent != null )
63     {
64         parent.getValues( language )
65             .forEach( lookupValue -> {
66                 addLookupValueReferences( lookup, values, lookupValue );
67             } );
68     }
69     //In case parent is null then by default all the values present in the
70     characteristic lookup should be returned
71     else if ( lookup != null )
72     {
73         Collection< LookupValue > lookupValues = lookup.getValues();
74         values.addAll( lookupValues );
75     }
76     return values.stream()
77                 .filter( v -> v.isActive() )
78                 .collect( Collectors.toSet() );
79 }
80
81 private void addLookupValueReferences( Lookup lookup, Set< LookupValue >
82 values, Object lookupValue )
83 {
84     LookupValue parentLookupValue = ( LookupValue ) lookupValue;
85     try
86     {
87         Collection< LookupValue > lookupValueReference =
88         parentLookupValue.getReferences( lookup );
89         values.addAll( lookupValueReference );
90     }
91     catch ( CoreException e )
92     {
93         throw new RuntimeException( e );
94     }
95 }
96
97 private CharacteristicRecord
98 getParentCharacteristicRecord( CharacteristicRecord characteristicRecord )
99 {
100     CharacteristicRecord parent;
101     if ( characteristicRecord.getCharacteristic()
102                                     .getValueProviderParameter() != null )
103     {
104         parent = getRequiredCharacteristicRecord( characteristicRecord,
105             characteristicRecord.getCharacteristic()

```

```

100         .getValueProviderParameter() );
101     }
102     else
103     {
104         parent = characteristicRecord.getParent();
105     }
106     return parent;
107 }
108
109 private CharacteristicRecord
getRequiredCharacteristicRecord( CharacteristicRecord
characteristicRecord,
110                                     String
parameter )
111 {
112     CharacteristicRecord parent = characteristicRecord.getParent();
113     JsonObject charParameter = jsonStringToJson( parameter );
114     String parentRefParam = ( charParameter != null
115                               &&
charParameter.get( PARENT_CHARACTERISTIC ) != null ) ?
charParameter.get( PARENT_CHARACTERISTIC )
116
117                               .getString()
118
119                               : null;
120     return getParent( parent, parentRefParam );
121
122 private CharacteristicRecord getParent( CharacteristicRecord parent,
String parentRefParam )
122 {
123     if ( parent == null )
124     {
125         return parent;
126     }
127     if ( parentRefParam != null && parent.getCharacteristic()
128                                           .getIdentifier()
129                                           .equals( parentRefParam ) )
130     {
131         return parent;
132     }
133     return getParent( parent.getParent(), parentRefParam );
134 }
135
136 private JsonObject jsonStringToJson( String input )
137 {
138     JsonObject jsonObject = null;
139     JsonParser parser = new JsonParser();
140     try
141     {
142         JsonElement element = parser.parse( input );
143         if ( element.isJsonObject() )

```

```

144     {
145         jsonObject = element.getAsJsonObject();
146     }
147 }
148 catch ( Exception e )
149 {
150     LOG.error( e.getMessage() );
151 }
152 return jsonObject;
153 }
154
155 public static class ValueImpl implements Value
156 {
157     private String label;
158     private Object value;
159
160     public ValueImpl( Object value, String label )
161     {
162         this.value = value;
163         this.label = label;
164     }
165
166     @Override
167     public Object getValue()
168     {
169         return this.value;
170     }
171
172     @Override
173     public String getLabel( Locale locale )
174     {
175         return this.label;
176     }
177 }
178 }
179 }

```

class name: "ParentReferencingChildLookupValueProviderParameter" package:
 "com.heiler.ppm.customizing.sample.valueProvider.parentchild" interface: "ValueProviderParameter".

ParentReferencingChildLookupValueProviderParameter

```

1 package com.heiler.ppm.customizing.sample.valueProvider.parentchild;
2
3 import java.util.ArrayList;
4 import java.util.Collection;
5 import java.util.Locale;
6
7 import com.heiler.ppm.characteristic.core.model.Characteristic;
8 import com.heiler.ppm.characteristic.core.record.CharacteristicRecord;

```

```

9      import
10      com.heiler.ppm.characteristic.core.valueProvider.ValueProviderParameter;
11
12      /**
13       * @author rkumhar
14       */
15      public class ParentReferencingChildLookupValueProviderParameter implements
16      ValueProviderParameter
17      {
18          @Override
19          public Collection getParameters( LookupValue language,
20      CharacteristicRecord characteristicRecord )
21      {
22          Collection< ParameterValue > paramList = new ArrayList< ParameterValue
23      >();
24          Collection< Characteristic > parentList = new ArrayList<>();
25          Characteristic characteristic =
26      characteristicRecord.getCharacteristic();
27          checkForParent( characteristic.getParent(), parentList );
28          parentList.forEach( parentCharacteristic -> paramList.add( new
29      ParameterValueImpl( parentCharacteristic.getIdentifier(),
30      parentCharacteristic.getName( Locale.ENGLISH ) ) ) );
31          return paramList;
32      }
33      Collection< CharacteristicRecord > checkForParent( CharacteristicRecord
34      characteristicRecord,
35      Collection<
36      CharacteristicRecord > parentCharRecordSetList )
37      {
38          if ( characteristicRecord != null )
39          {
40              parentCharRecordSetList.add( characteristicRecord );
41              if ( characteristicRecord.getParent() != null )
42              {
43                  checkForParent( characteristicRecord.getParent(),
44      parentCharRecordSetList );
45              }
46          }
47          return parentCharRecordSetList;
48      }
49
50      Collection< Characteristic > checkForParent( Characteristic
parentCharacteristic,

```

```

51                                     Collection< Characteristic
52 > parentCharList )
53 {
54     if ( parentCharacteristic != null )
55     {
56         parentCharList.add( parentCharacteristic );
57
58         if ( parentCharacteristic.getParent() != null )
59         {
60             checkForParent( parentCharacteristic.getParent(),
61                             parentCharList );
62         }
63     }
64     return parentCharList;
65 }
66
67 public static class ParameterValueImpl implements ParameterValue
68 {
69     private String label;
70     private Object value;
71
72     public ParameterValueImpl( Object value, String label )
73     {
74         this.value = value;
75         this.label = label;
76     }
77
78
79     @Override
80     public Object getValue()
81     {
82         return this.value;
83     }
84
85     @Override
86     public String getLabel( Locale locale )
87     {
88         return this.label;
89     }
90 }
91 }

```

6.4.9.8 Characteristic modeling:

Create characteristics like the following:

Characteristic	Name	Identifier	Data type	Lookup
Parent characteristic	Country -> state	Country_to_State	Lookup value	Country
Child characteristic	Parent referencing child lookup	parent_referencing_child_lookup	Lookup value	State

Assigning value provider to characteristics in modelling UI

- Value provider dropdown lists all the contributed value providers.
- Parameter dropdown list appropriate values for corresponding value provider

The screenshot displays the Informatica Product 360 Modelling UI. On the left, a sidebar contains navigation icons for Structures, Catalogs, Media, Tasks, Queries, Search, Assortments, and Lookups. The main area is titled 'Demo_Category' and shows a table with the following data:

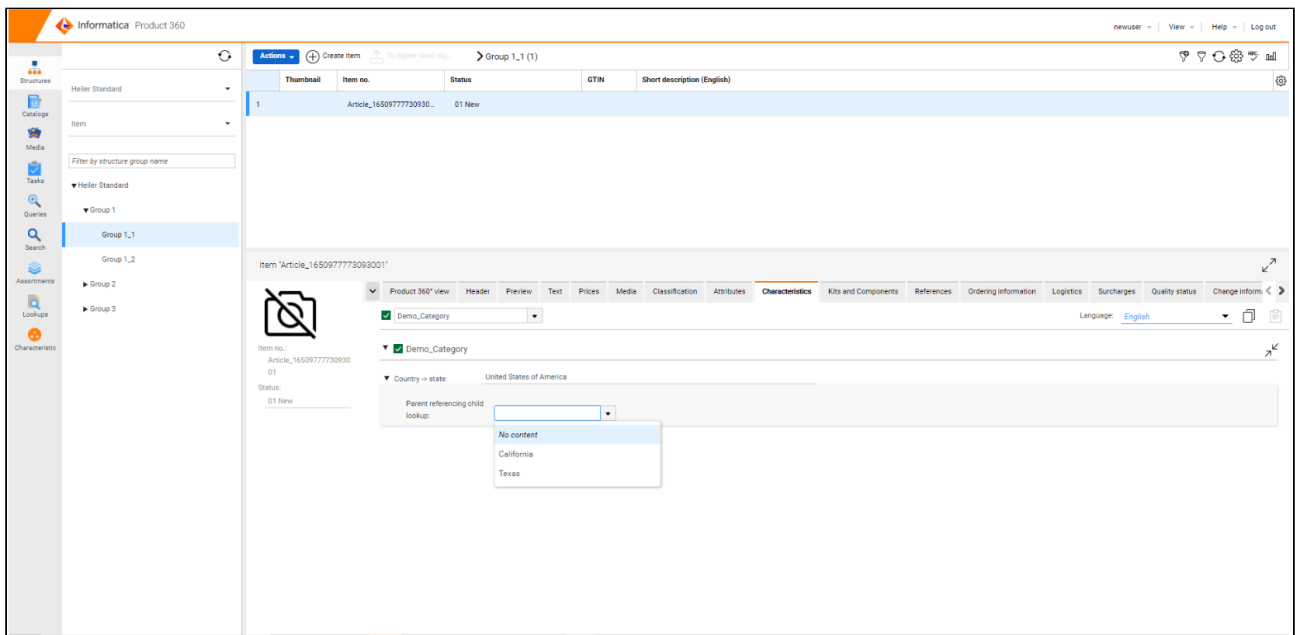
Name (English)	Identifier	Min./Max. records	Lookup list	Conditional values
Country -> state	Country_to_State	0 to 1 (optional)	Country	
Parent referencing child lookup	parent_referencing_child_lookup	0 to 1 (optional)	State	No restrictions

Below the table, the 'Details' tab for the characteristic 'parent_referencing_child_lookup - Parent referencing child lookup' is expanded. It shows the following configuration:

- Languages:** English
- Name:** Parent referencing child lookup
- Description:** No content
- Identifier:** parent_referencing_child_lookup
- Minimum records:** 0
- Maximum records:** 1
- Purposes:** No content
- Target markets:** No content
- Standardization systems:** GDSN
- Alternative identifier:** No content
- Data type:** Lookup value
- Value provider:** Parent Child Value Provider
- Parent Characteristic:** Country -> state
- Default value:** No content
- Read-only:** ☐
- Multiple values:** ☐
- Unit:** No content
- Validation expression:** No content
- Test input:** No content

Characteristic UI showing the provided values

Characteristic showing filtered values of state lookup.



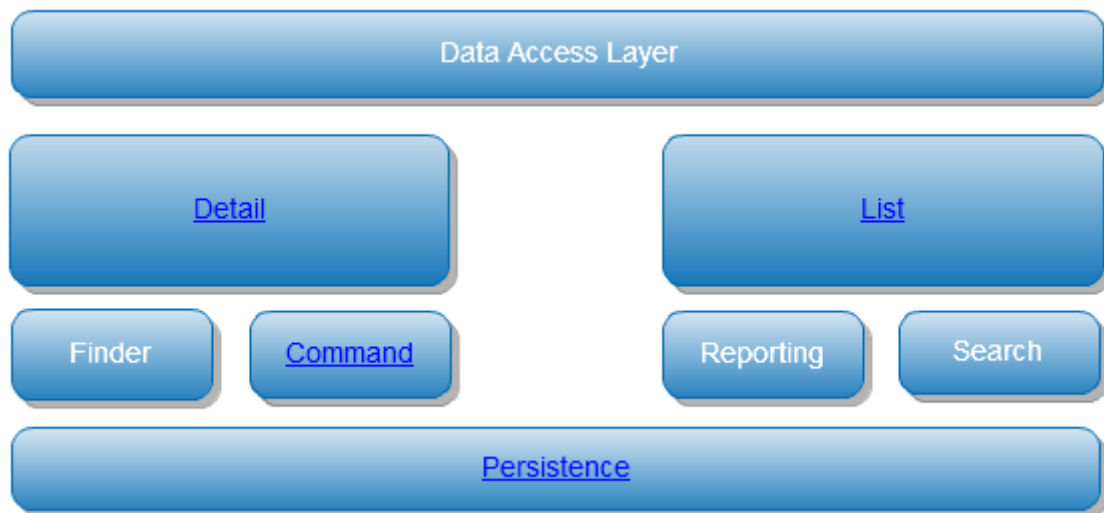
6.5 Data Access

- [Overview](#) (see page 131)
- [Detail access](#) (see page 132)
- [List access](#) (see page 132)

6.5.1 Overview

The data access layer consists of two main parts: entity detail access and list access. The entity detail access provides CRUD (Create Read Update Delete) operations and look up methods (finder framework) for the entities defined in the repository. The list access provides read-only methods to fetch entities data in a table form. The list access uses a supplementary reporting framework. The reporting framework is used to create a persistent set of object ids according to some business criteria. This set is supplied as a parameter to the list access methods.

Entity detail and list access methods provide API which operates with business model artifacts (i.e. EntityType, Field, FieldPath, Logical key etc.). This effectively isolates the data access layer's business logic from the persistence aspects. In most cases developers do not need to take care about persistence technologies like hibernate or jdbc.



Data access layer also has different extension points which can be used to adjust default persistence logic. Developer usually need to create extensions if the business model has been modified or extended in a non standard way. Before contributing extensions in the data access layer, please consider other possibilities like hpm command framework. Keep in mind that data access layer extensions should be used to adjust persistence logic and should not be used to implement business logic!

Most of the data access implementation logic uses meta information to perform data manipulation operations (deletion, search, update, hql queries generations etc). This meta information is composed from persistence fields defined in the repository (called mappings) and JPA persistence entity called persistence meta model. JPA entities itself are called persistence model. See further section for more details.

6.5.2 Detail access

Detail models can be accessed using `EntityManager`. On this level entity details are represented as EMF/SDO `EDataGraph` (data transfer object used in HPM platform). The `EntityManager` provides read, save, delete and find operations and isolates persistence layer from the application layer (business layer and platform services). It also takes care about detail model in memory caching, find results caching and change notification.

6.5.3 List access

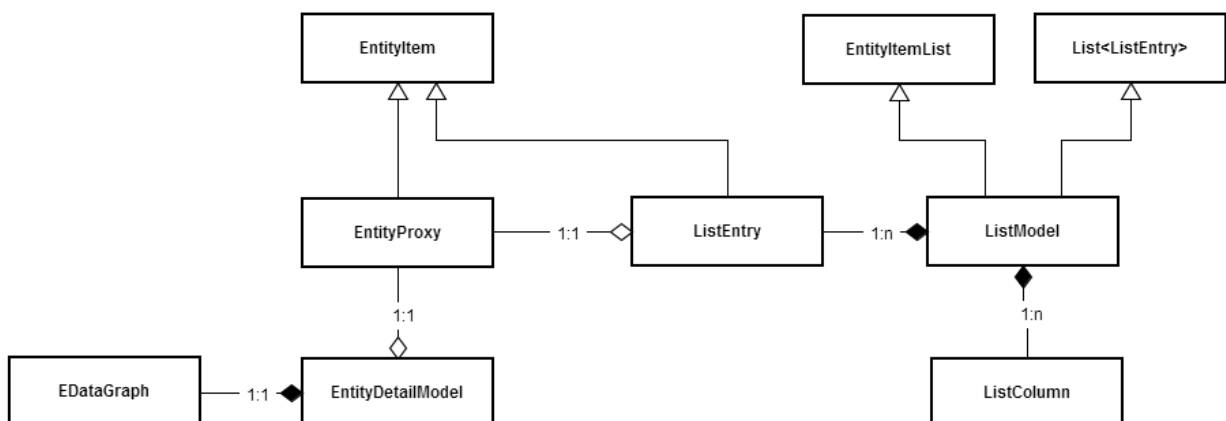
List models can be loaded using the `ListModelLoadService`. List access is a read-only method to get large sets of tabular data. List columns are defined using same terms as used in entity detail access (i.e. `FilePaths`, `EntityType` etc.). So the list access is used to get a flat table of the entity detail graphs. List access provides optimized version of list which is based on the lazy loading principal - table rows are loaded block-by-block as soon as they are really needed. It helps to reduce memory consumption. List access uses the reporting framework to define which rows should be loaded. Reporting framework provides a possibility to create sets of internal object ids according to some business criteria. These sets are persisted in the database and have identifiers (`reportId`). Business criterias (reports) are usually defined using Stored Procedures. However it is also possible to define reports in runtime using search.

6.5.4 Data Models

Gives a high level view on the most important classes and aspects regarding the Product Manager data access

- [Model Overview](#) (see page 133)
- [Entity Item](#) (see page 133)
- [Entity Proxy](#) (see page 134)
 - [Entity Proxy Revision](#) (see page 134)
- [Detail Model](#) (see page 134)
 - [Caching](#) (see page 134)
 - [Exclusive Access Pattern \(locking\)](#) (see page 134)
 - [Usability Shortcut for "read a single attribute"](#) (see page 136)
- [List Model](#) (see page 136)

6.5.4.1 Model Overview



6.5.4.2 Entity Item

The term "entity item" is used to identify a single business object which can be represented by different models.

An entity item is always a concrete real-world object which maps to a physical record in the database. (E. g. a *product* or *structure group* object.)

Additionally to that, there is also an interface `EntityItem` which inherits from `EntityItemList`. This interface has been created to unify different models in the application by the least common attributes. Since a single object can be seen just as a "special case of a list" - namely a list with just one entry - the `EntityItem` interface inherits from the `EntityItemList` interface. Therefore where ever you can use an `EntityItemList`, you can also use an `EntityItem`.

6.5.4.3 Entity Proxy

The `EntityProxy` represents an immutable proxy object pointing to an already persistent entity item or to an entity item that is going to be persisted soon. Depending on how it is created it contains an internal or external identifier of an entity item. Using the entity proxy you can obtain its corresponding singleton `EntityDetailModel` instance. Other than detail models, entity proxies are serializable and usually can be converted to a String representation as well as a Long representation (Long conversion is not available for entities which have a parent like `StructureGroup`, `Product` or `Article`).

Entity Proxy Revision

The `EntityProxyRevision` class combines an `EntityProxy` with a `RevisionToken`. Therefore it kind of qualifies the entity item by its revision.

6.5.4.4 Detail Model

The `EntityDetailModel` interface is a container for the actual data which needed for infrastructure needs. The detail model instances will be cached and the data will automatically be invalidated in case the entity item has been changed by someone on the network. For your convenience the `EntityDetailModel` also has methods to access the data of the entity item. In case it is currently in an invalidated state (thus, the internal data graph is null), the `EntityDetailModel` will trigger the reload of the data transparently. Modifications of entity items can (and should) only be done by modifying the detail model's data (usually by using the command framework). Exceptions to this rule only apply for specific mass data operations like the deletion of many entity items.

Caching

Loading detail models can be quite resource intensive. That is the reason why they are cached on the server and on the client. The detail model cache is managed by the

`EntityManager`. Detail model instances are designed as singletons. For a specific product or structure group, there is only one detail model in the client or server VM.

Exclusive Access Pattern (locking)

Since a detail model is a singleton, the access to the model must be synchronized in order to be thread safe. For example, the data of the detail model might be invalidated while you are currently working with it (because a different client modified the same object).

In order to prevent any concurrency problems you will need to obtain exclusive access on the detail model. For this purpose, acquire and release methods are available which provide reentrant high concurrency synchronization (multiple reads, single write).

It is absolutely necessary to strictly follow the following acquire/release pattern, otherwise serious problems may arise:

Concurrent Read Access Pattern

```

1 EntityDetailModel detailModel = ...;
2 detailModel.acquireRead();
3 try
4 {
5     //read data from the detail model or it's data object
6     //but, only READ, write is not permitted with a read lock!
7     //do NOT call asynchronous events or something similar, this might cause
    deadlocks!
8 }
9 finally
10 {
11     detailModel.releaseRead();
12 }
```

Exclusive Write Access Pattern

```

1 EntityDetailModel detailModel = ...;
2 detailModel.acquireWrite();
3 try
4 {
5     //read or write data from/to the detail model or it's data object
6     //do NOT call asynchronous events or something similar, this might cause
    deadlocks!
7     detailModel.save();
8 }
9 finally
10 {
11     detailModel.releaseWrite();
12 }
```

Why is this code block so important?

If you check the `acquireRead` method you will see that it declares a `CoreException` to be thrown in case of any problem.

In case a core exception really is thrown while you acquire the read lock, you **must not** release the lock - or you will run into an `IllegalMonitorStateException` like this:

```

java.lang.IllegalMonitorStateException: Non-owning thread tried to release
detail model for entity item
'com.heiler.ppm.std.core.entity.EntityProxyRevision@1d80945[EntityProxy=10001[[]
{#}18[[]],RevisionToken=RevisionToken[ID=1]]' which should never happen. Owning
write thread is '<unknown>'. Number of concurrent read threads '0'. Thread tried
```

to release is 'main'. Please make sure that you always follow the try/finally pattern with detail model's acquire and release.

You might believe, this is a very unlikely situation, because the `acquireRead` / `acquireWrite` methods surely won't throw an exception, but this is **wrong**.

`AcquireRead` will also throw a `CoreException` in case the current user has no read permission on this detail model - `acquireWrite` will throw it in case the current user has no write permission.

So you see, this might very well occur quite often.

On the other side, when the acquire methods succeed, you must make sure to call also the release methods (that's what the finally block is for).

So it's the best approach to have no other code line between the acquire call and the try.

Additionally to this most important pattern, you're also not allowed to perform an `acquireWrite` from within an `acquireRead`.

The detail model allows no lock upgrade since multiple threads can simultaneously read the detail model.



Always use the correct locking. If you really only read the detail model, then use `acquireRead`. `AcquireWrite` will give you exclusive access for the cost of performance.

Usability Shortcut for "read a single attribute"

Internally, the detail model is implemented in a thread safe manner, which means, the methods you may call on the detail model are internally synchronized in such a way that they can be treated as atomically. Like always when you have objects which are designed thread safe, multiple method calls to these objects aren't thread safe since a different thread could always interfere in between the method calls - that's why it's absolutely necessary to follow the pattern above when you call multiple methods on the detail model which must be treated together as "atomic operation" (like get data graph, modify data graph and save detail model (with the modified data graph))

However, sometimes you only need a single attribute and that's it, for this you can omit the complete acquire/release locking and just call `detailModel.getFieldValue(...)`.

6.5.4.5 List Model

The list model is an in-memory data structure comparable to a table. A column is represented by a `ListColumn` and a row by a `ListEntry`. The field values of a list entry can be accessed with the index of a specific column.

List models can either be created from scratch using the `ListModelFactory` or loaded from the data storage using the `ListModelLoadService`

ListModel

ListEntry

ListColumn

Artikel-Nr.	EAN	Hersteller Artikel-Nr.	Hersteller	Kurzbeschreibung (Deutsch)
1500038	8711428031836	XT000770039	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500039	0501002772599	XT000733466	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500040	9415442001243	XT000733490	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500041	5941544200725	GT500005548	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500042	5941544200728	DE999966216	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500043	5010027750616	DE9999667073	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500044	4001895639821	DE999966232	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500045	4001895613535	DE9999667081	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500046	4001895640506	DE999966257	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500047	4001895613658	DE9999667099	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500048	4001895643934	DE999966273	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500049	4001895613733	DE9999667107	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1500050	4001895643989	DE999966299	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1501236	4001895613771	DE9999667115	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1501237	4001895644016	DE999966489	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1501238	4001895613795	DE999966497	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1501239	4001895638961	DE999966539	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...
1501240	4001895639838	TE100024302	ATLAS ...	Top 300 EE, Sicherheitsstiefel mit G...

2 image2012-4-12 15:14:50.png

- Non-virtual list model
When loading a non-virtual list model, the whole content for the model (including all data/values) is loaded. This can take a long time in case of handling mass data and does not always make sense as there may be no need to access all the data in a list model all at once.
- Virtual list model
If using a virtual list model, the list model will not contain all data initially. Only a skeleton container of the model is loaded instead. The consequences are higher performance when obtaining the list model and a lower memory usage. If one tries to access data that is not yet available, the list model fetches the requested data (and a configured additional offset) from the database.

6.5.5 Data Selection

6.5.5.1 Entity Reporting

In general, a report can be understood as the "where" dimension of an sql query in a relational database - a predefined query with a fixed set of parameters whose result is persisted and therefore reusable.

- [Low-Level Reporting](#) (see page 137)
- [Entity Report](#) (see page 139)
 - [Example](#) (see page 140)
 - [Programmatic execution of entity reports](#) (see page 144)

Low-Level Reporting

When being executed, a report gathers a list of root object ids based on its fixed internal semantics, which can be defined by stored procedures, [search expressions](#) (see page 151) or an existing array of ids.

Subsequently, this list is persisted in the storage table, along with the report's id, allowing an N:M mapping

between reports and root entity objects. The purpose of this approach is to have fast reusable access on an entity set in [list model](#) (see page 133)s, where usually thousands of records are to be loaded and displayed at once, and all other occasions when dealing with lists of entity items. Therefore, only the root entities' ids are loaded and stored internally whereby the actual entity data fields (the select dimension of the query - which is defined by utilizing FieldPaths) are loaded separately via SQL joins and thereby reusing the report's result. Thus, only the requested data needs to be transferred to the caller, omitting unnecessary column data.

Reports carry the following information:

- unique id of the report
- data source: MAIN/MASTER/SUPPLIER
- purpose: TEMPORARY (report is going to be deleted after a specific amount of time) or PERMANENT
- type: root entity type (e.g. ArticleType)
- result table: Name of the storage table containing the report's object ids.
 Either `ReportStoreTempA[0..n]` / `ReportStoreTempB[0..n]` if temporary
 or `ReportStore` if permanent report.
- count: number of ids this report consists of

Depending on its purpose, the relevant mappings between root entity ids and reports are either stored in the `ReportStoreTempA[0..n]` / `ReportStoreTempB[0..n]` (for temporary reports - this is most often the case) or `ReportStore[0..n]` table (permanent reports).

For performance reasons, each table `ReportStoreTempA`, `ReportStoreTempB` and `ReportStore` exists `n` times, where `n` is configurable (default is 10). Reports are respectively stored in these `n` table instances following the round robin algorithm.

Temporary reports are rotationally stored in either `ReportStoreTempA[0..n]` tables or `ReportStoreTempB[0..n]` tables with a cleanup job continuously switching between those tables and executing a TRUNCATE statement of the table to be used henceforth, as well as deleting the report header data in Report table associated with this table.

In order to reuse and work with reports, they support additional operations like subtract, intersect, copy and delete via the `ReportService`.

ReportQuery: The `ReportQuery` class represents the input for a stored procedure based report. This implies the name of the data source, the EntityType of the item's to be reported, the query method (stored procedure or search expression), the purpose of the report, the origin of the report, etc.

Reporting stored procedures

Note that reporting stored procedures are deprecated, since they are database dependant and thus maintaining them is very expensive. Instead [search expressions](#) (see page 151) should be used. Nonetheless, reporting stored procedures must fulfill a special API contract so they can be used in the framework. Reporting stored procedures must have an OUT parameter called `FCA_ReportSQL` which holds the select statement returning the distinct IDs of the query. No order-by clause is allowed, since this is being done by the reporting framework. No DISTINCT is needed, because duplicates are removed by the reporting framework.

ReportResult: A report result is the consequence of an executed report query. It represents an *immutable report proxy*, referring to a report which has been executed and stored in the database. It holds the name of the data source (MAIN / MASTER / SUPPLIER) where the report has been executed and stored, the persisted

report's database-internal unique id, the report's type and purpose, the name of the report's result table name (ReportStoreTempA / ReportStoreTempB / ReportStore), as well as current amount of reported items.

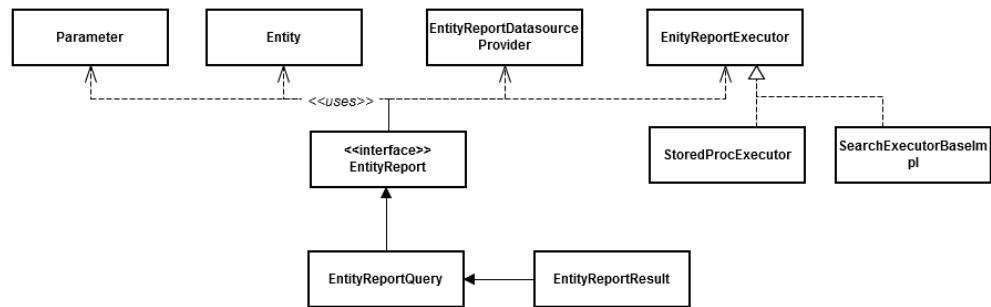
ReportUtils: This deprecated utility class provides methods for executing report queries and creating reports based on given item ids. It also includes a method for deleting non-temporary reports. Note: Since version 6.0 **ReportService** shall be used instead.

ReportCleanerJob: This job runs every hour and deletes temporary reports from the database.

Entity Report

In order to provide the means for generic report contributions, a new report approach has been elaborated which enhances the possibilities of the Low-Level reporting, thus it's not a replacement, it's an extension.

Class Diagram



EntityReport: The **EntityReport** represents an extension/contribution to the **com.heiler.ppm.entity.core.entityReports** extension point. It incorporates the needed information for providing generic reports:

- Typed parameters, which can have default values in case not mandatory
- Entity of the repository this report stores ids of
- The datasource provider for specifying which db is targeted (MAIN/MASTER/SUPPLIER)
- An **EntityReportExecutor** which basically contains the logic/semantic of the report query. The **StoredProcExecutor** can be used for wrapping the old **ReportQuery** (with parameter mapping, stored procedure name, ...). As a second executor, the Product Manager search expressions can be used in order to build HQL based queries.

EntityReportQuery: The query keeps a reference to the report plus the necessary information for a report to be executed, mainly the actual values for the report parameters.

EntityReportResult: The actual entity result report contains a reference to the executed query and represents an immutable result information of a persisted report, including the same attributes as in **ReportResult** (e.g. id, datasource, count, purpose)

EntityReportRegistry: This is the central point for registering reports to be contributed and accessing by alias etc



Custom entity reports can be contributed through the `com.heiler.ppm.entity.core.entityReports` extension point. To be able to access them through the report service they need to have an alias name defined.

Example

When contributing new entity reports, the following properties are most important. Note that a detailed property description is also available via mouse hover tool tips in the Eclipse extension menu.

Property Name	Purpose
id	unique id for the entity report
item-entity	entity type to be returned
alias	report alias to be used in service api as well as in report service
data source	data source the report has to be executed against either <code>FixDataSourceProvider</code> (MAIN/MASTER/SUPPLIER) or <code>com.heiler.ppm.catalog.core.report.DataSourceByCatalogProvider</code> for having the data source determined dynamically by parent catalog
supports-service-api	if on true, this report can be called from service api

As mentioned above, entity reports need an executor for the report's semantics, which can be specified in the executor node. Here, either add a class implementing `EntityReportExecutor`, or add a `sp-executor-spec` child node, where the name of the stored procedure to be called has to be specified. Entity report parameters can be added as additional nodes to the report node, with the most important properties being:

Property Name	Purpose
id	unique id of the parameter

Property Name	Purpose
name	parameter name, which can be externalized for i18n use
alias	service api alias for the parameter
visible	if set to true (default), parameter is visible in the ui. If set to false, parameter should be set to non-mandatory (see below) or must have a default value
mandatory	parameter mandatory for report execution or not
enum	repository based enum providing the possible values
collection	if set to true, a collection of values is expected. By default is false.

Additionally, a showForEntity key-value node can be added to the report node, which is used to decide whether this report shall be displayed in the UI context menu and if so, for which entity the entry shall appear (e.g. MasterCatalog or SupplierCatalog). This key-value is capable of taking more than just one entity, just separate the entities with a comma. (e. g. MasterCatalog,SupplierCatalog)

Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

com.heiler.ppm.value.core.dataTypes

com.heiler.ppm.entity.core.entityReports

Articles of Product (report)

Articles of Catalog (report)

Articles of Catalog in Structure Group (r

Articles of Catalog classified to (report)

Articles of Catalog not classified to (rep

Articles of Catalog by Status (report)

Artikel nach erreichten Status (repo

Catalog (parameter)

Status (parameter)

java.lang.Integer (value-class)

Status (sp-parameter-spec)

RevisionToken (parameter)

RevisionTokenCompare (parameter,

(executor)

PPM4_RepArt_GetByStatus (sp-e

(sp-hidden-parameter)

(sp-hidden-parameter)

showForEntity (key-value)

Articles of Catalog by Status not (report

Articles of Master Catalog (report)

Articles of Master Catalog classified to (

Articles of Master Catalog not classified

Add...

Remove

Up

Down

Extension Element Details

Set the properties of "report". Required fields are denoted by "*".

id*:

com.heiler.ppm.article.core.SupplierArticlesBySta

item-entity*:

Article

class:

Browse...

alias:

byStatusReached

data-source:

com.heiler.ppm.catalog.core.report.l

Browse...

parent-item-provider:

com.heiler.ppm.entity.core.report.ut

Browse...

permissions:

name:

Articles of Catalog by Status

order:

400

supports-service-api:

true

Overview

Dependencies

Runtime

Extensions

Extension Points

Build

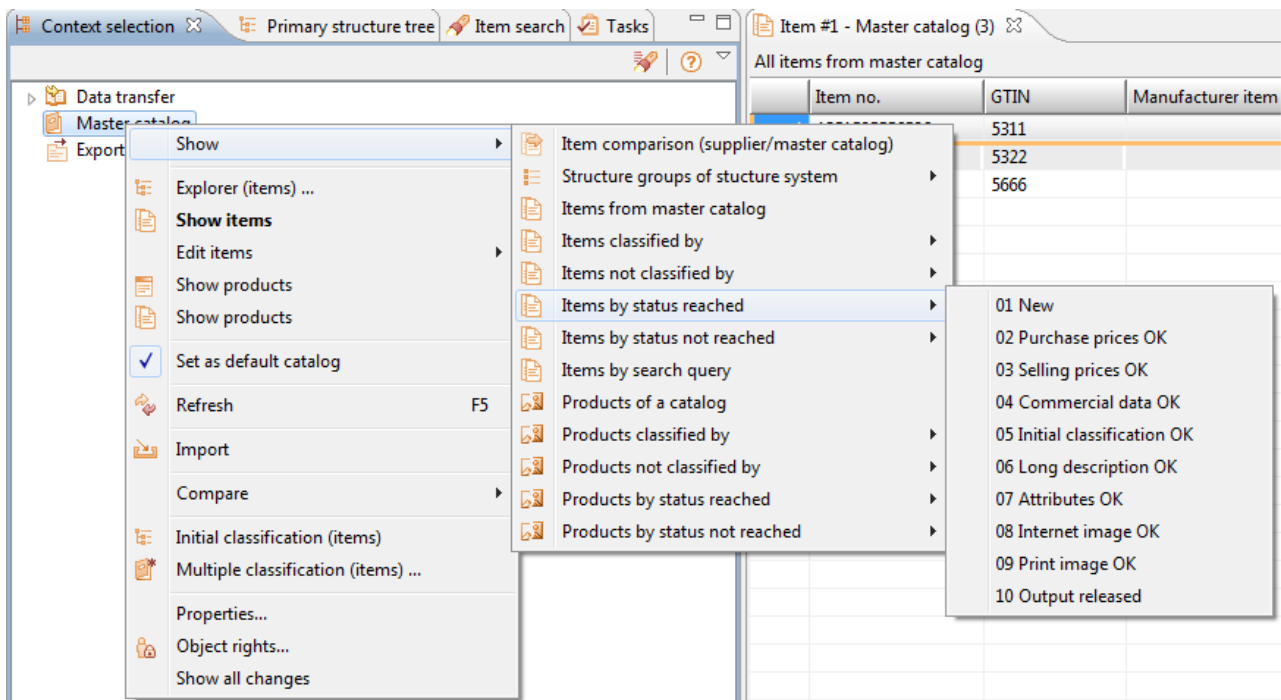
MANIFEST.MF

plugin.xml

build.properties

Contributed reports can be utilized/examined in the Product Manager client by calling predefined report, like in our example show-> items by status reached:

142



All the information, including the different status values for the respective parameter in our example, are pulled generically from the contribution. All changes taken within the contribution's plugin.xml are immediately reflected in the UI.

Edit predefined query

You can edit the predefined query here, e.g. change its name and specify arguments.

Template: Provides all items of the given catalog having the specified reached status

Name:

Description:

Parameter

Status:

Version comparison:

Catalog:

Version:

- 01 New
- 02 Purchase prices OK
- 03 Selling prices OK
- 04 Commercial data OK
- 05 Initial classification OK
- 06 Long description OK
- 07 Attributes OK
- 08 Internet image OK
- 09 Print image OK
- 10 Output released

As you can see, such entity report queries can also be edited by pressing "Edit query" in the table view menu, where e.g. a different status can be selected as a parameter.

Programmatic execution of entity reports

The following test code shows the example from above (MasterArticlesByStatus) to be called in programmatic manner via the `ListModelLoadService`, thereby executing the report implicitly and loading the relevant matching items in one shot.

Programmatic execution of entity reports via list model load service

```
1 package com.heiler.ppm.article.core.internal.characteristic.compare;
```

```

2
3  import org.eclipse.core.runtime.CoreException;
4  import org.eclipse.core.runtime.NullProgressMonitor;
5
6  import com.heiler.ppm.catalog.core.CatalogConst;
7  import com.heiler.ppm.entity.core.report.EntityReport;
8  import com.heiler.ppm.entity.core.report.EntityReportQueryBuilder;
9  import com.heiler.ppm.entity.core.report.EntityReportQueryItemList;
10 import com.heiler.ppm.entity.core.report.EntityReportRegistry;
11 import com.heiler.ppm.repository.EntityType;
12 import com.heiler.ppm.repository.path.FieldPath;
13 import com.heiler.ppm.repository.util.RepositoryUtils;
14 import com.heiler.ppm.std.core.entity.EntityProxy;
15 import com.heiler.ppm.std.core.entity.EntityProxyFactory;
16 import com.heiler.ppm.std.core.list.ListEntry;
17 import com.heiler.ppm.std.core.list.ListModel;
18 import com.heiler.ppm.std.core.list.ListModelLoadParams;
19 import com.heiler.ppm.std.core.list.ListModelLoadService;
20
21 @SuppressWarnings( "nls" )
22 public class ArticlesByStatusReportExecutor
23 {
24     private EntityReportRegistry entityReportRegistry;
25     private EntityProxyFactory   entityProxyFactory;
26     private ListModelLoadService listModelLoadService;
27     private EntityType           catalogEntityType;
28
29     public ArticlesByStatusReportExecutor( EntityReportRegistry
30 entityReportRegistry,
31                                           EntityProxyFactory
32 entityProxyFactory,
33                                           ListModelLoadService
34 listModelLoadService )
35     {
36         this.entityReportRegistry = entityReportRegistry;
37         this.entityProxyFactory   = entityProxyFactory;
38         this.listModelLoadService = listModelLoadService;
39         this.catalogEntityType    = RepositoryUtils.getRepositoryRootEntityType(
40 "CatalogType" );
41     }
42
43     public void executeArticlesByStatusReport() throws CoreException,
44 InterruptedException
45     {
46         //Obtain report from registry
47         EntityReport report = this.entityReportRegistry.getReport(
48 "com.heiler.ppm.article.core.MasterArticlesByStatus",
49                                     false );
50
51         //Create query builder
52         EntityReportQueryBuilder queryBuilder = new
53 EntityReportQueryBuilder( report );
54     }
55 }

```

```

48     // set parameters:
49     // obtain proxy for master catalog
50     EntityProxy catalogProxy = this.entityProxyFactory.createEntityProxy(
this.catalogEntityType, null,
51     CatalogConst.MASTER_ID );
52     // Items from master catalog
53     queryBuilder.setParamValue( "Catalog", catalogProxy );
54
55     // All items with status "new" -> key 100
56     queryBuilder.setParamValue( "Status", 100 );
57
58     // Entity item list for specifying list model's row results
59     EntityReportQueryItemList itemList = new
EntityReportQueryItemList( queryBuilder.toQuery() );
60
61     //Get List Model
62     // We want the ean field to be loaded for the list model
63     FieldPath[] fieldPaths = new FieldPath[] { new FieldPath(
"ArticleType.Ean" ) };
64
65     // Parameters for specifying list model's result. Rows by entity item
list, columns by field paths
66     ListModelLoadParams loadParams = new ListModelLoadParams( itemList,
fieldPaths );
67
68     // Load the list model, internally calls the report query executor for
running and persisting the entity report
69     ListModel lm = this.listModelLoadService.loadListModel( new
NullProgressMonitor(), loadParams );
70
71     // Obtain the list model's entries
72     for ( ListEntry entry : lm )
73     {
74         String ean = ( String ) entry.getFieldValue( 0 );
75         System.out.println( "item ean = " + ean );
76     }
77 }
78 }

```

A corresponding test executing this report code could look like the following:

Test injecting the dependencies and executing the report

```

1     @Test
2     public void runArticlesByStatusReportExecutor() throws CoreException,
InterruptedException
3     {
4         EntityReportRegistry entityReportRegistry =
EntityReportComponent.getReportRegistry();

```

```

5      EntityProxyFactory entityProxyFactory =
      EntityManagementComponent.getEntityProxyFactory();
6      ListModelLoadService listModelLoadService =
      EntityManagementComponent.getListModelLoadService();
7
8      ArticlesByStatusReportExecutor articlesByStatusReportExecutor = new
      ArticlesByStatusReportExecutor(
9
10                                     entityReportRegistry,
      entityProxyFactory, listModelLoadService );
11      articlesByStatusReportExecutor.executeArticlesByStatusReport();
12  }

```

Entity Report Filtering

By means of so called entity report filters it is possible to further narrow down the result of an entity report query. Such filters are applied after the execution of the report and can be contributed for either one specific entity report or as a general filter for all reports registered in the system. This offers the advantage of being able to reuse existing report query logic and simply adding general filtering steps for confining the result set.

- [Extension Point](#) (see page 147)
- [Parameter Grouping in PIM Rich Client](#) (see page 147)
- [Example](#) (see page 148)

Extension Point

Entity report filters can be contributed to *com.heiler.ppm.entity.core.entityReports* extension point, where the following information needs to be provided:

- **id**: A unique extension identifier of the contributed entity report filter.
- **filter-class**: An implementation of interface *EntityReportFilter*, containing the filter's logic as well as the programmatic steps to enhance an entity report with additional parameters etc.
- **entity-report-identifier** (optional): The identifier of the report to be filtered. If omitted, filter will be registered for all available entity reports.

Multiple filters registered for one report are NOT applied in a specific sequence and simply generate the intersection set of PIM objects adhering to the entity report result and corresponding filters.

Parameter Grouping in PIM Rich Client

For displaying entity report parameters within a special group in the PIM Rich Client's "edit query dialog", add a key/value pair to the corresponding parameter(s) like this:

```
parameter.setData( AbstractParameter.KEY_PARAMETER_GROUP, "Parameter group name" );
```

This is how the data quality status filter looks like for instance:

Example

One example for a convenient implementation example would be an assortment entity report filter, providing the means to filter each report result containing only PIM objects included in a given assortment.

AssortmentReportFilter

```

1  public class AssortmentReportFilter implements EntityReportFilter
2  {
3      public static final String PARAM_ASSORTMENT = "assortmentFilter";
4      /$NON-NLS-1$
5      private static final String VALUE_PARAM_GROUP =
6      "AssortmentReportFilter.paramGroup"; //$NON-NLS-1$
7
8      @Override
9      public EntityItemList filter( IProgressMonitor progressMonitor,
10      EntityItemList listToBeFiltered,
11      EntityReportQuery query ) throws
12      CoreException, InterruptedException
13      {
14          EntityReportQuery.NamingTypes queryNamingType =
15          EntityReportQuery.NamingTypes.NAME;
16          String queryName = query.getDisplayNaming( queryNamingType, null, true
17          );
18          String taskName = Messages.getString( "StatusReportFilter.task" ); //
19          $NON-NLS-1$
20          taskName = MessagesUtils.replace( taskName, false, queryName );
21          progressMonitor.beginTask( taskName, 1000 );
22          try
23          {

```



```

17      AssortmentProxy assortment = ( AssortmentProxy )
getParameterValue( PARAM_ASSORTMENT, query );
18      ReportResult assortmentReportResult =
assortment.getAssortmentDetailModel()
19                                          .getItemReportResult
();
20      ReportService reportingService =
ReportingComponent.getReportService();
21      ReportResult toBeFilteredReportResult =
listTobeFiltered.toReportResult( progressMonitor,
22
ReportPurposes.TEMPORARY );
23      ReportResult filteredReportResult =
reportingService.intersectWithReport( assortmentReportResult,
24
toBeFilteredReportResult, false,
25
ReportPurposes.TEMPORARY );
26      Entity itemEntity = query.getReport()
27                                  .getItemEntity();
28      EntityItemList filteredItems = new
ReportResultEntityItemList( itemEntity, filteredReportResult );
29      return filteredItems;
30  }
31  finally
32  {
33      progressMonitor.done();
34  }
35  }
36
37  @Override
38  public void adaptEntityReport( EntityReport entityReport )
39  {
40      AbstractEntityReport report = ( AbstractEntityReport ) entityReport;
41      String entityIdentifier = report.getItemEntity()
42                                  .getIdentifier();
43      Parameter assortmentParam = null;
44      switch ( entityIdentifier )
45      {
46          case "Article": //$NON-NLS-1$
47              assortmentParam = createAssortmentParameter(
"Enum.AllArticleAssortments", "Article", //$NON-NLS-1$ //$NON-NLS-2$
48
"com.heiler.ppm.article.assortment.core.permission.Read" ); //$NON-NLS-1$
49              break;
50          case "Product2G": //$NON-NLS-1$
51              assortmentParam = createAssortmentParameter(
"Enum.Product2GAssortments", "Product2G", //$NON-NLS-1$ //$NON-NLS-2$
52
"com.heiler.ppm.product2g.assortment.core.permission.Read" ); //$NON-
NLS-1$
53              break;
54          case "Variant": //$NON-NLS-1$

```

```

55         assortmentParam = createAssortmentParameter(
56             "Enum.VariantAssortments", "Variant", //$NON-NLS-1$ //$NON-NLS-2$
57             "com.heiler.ppm.variant.assortment.core.permission.Read" ); //$NON-NLS-1$
58             break;
59             default:
60                 return;
61         }
62         report.addParameter( assortmentParam );
63     }
64     @Override
65     public void clearEntityReport( EntityReport entityReport )
66     {
67         AbstractEntityReport report = ( AbstractEntityReport ) entityReport;
68         Entity itemEntity = report.getItemEntity();
69         String entityIdentifier = itemEntity.getIdentifier();
70         if ( entityIdentifier.equals( "Article" ) || entityIdentifier.equals(
71             "Product2G" ) //$NON-NLS-1$ //$NON-NLS-2$
72             || entityIdentifier.equals( "Variant" ) ) //$NON-NLS-1$
73         {
74             report.removeParameter( PARAM_ASSORTMENT );
75         }
76     }
77     @Override
78     public boolean isApplicable( EntityReportQuery query )
79     {
80         try
81         {
82             return getParameterValue( PARAM_ASSORTMENT, query ) != null;
83         }
84         catch ( CoreException shouldNotHappen )
85         {
86             String msg = "Unexpected error while obtaining parameter value for
87                 assortment report filter."; //$NON-NLS-1$
88             throw new IllegalStateException( msg, shouldNotHappen );
89         }
90     }
91     protected Object getParameterValue( String paramIdentifier,
92         EntityReportQuery query ) throws CoreException
93     {
94         Object result = null;
95         Parameter parameter = query.getReport()
96             .getParameter( paramIdentifier );
97         if ( parameter != null )
98         {
99             result = query.getParamValue( parameter, false );
100         }
101         return result;
102     }

```

```

103     private Parameter createAssortmentParameter( String
assortmentIdentifier, String entityIdentifier, String permission )
104     {
105         ParameterImpl parameter = new ParameterImpl( PARAM_ASSORTMENT );
106         parameter.setAlias( PARAM_ASSORTMENT );
107         parameter.setMandatory( false );
108         parameter.setVisible( true );
109         parameter.setOrder( 15000 );
110         parameter.setMessagesHelper( Messages.messagesHelper );
111         parameter.setNameOrResourceKey(
"%AssortmentReportFilter.param.assortment.name" ); //$NON-NLS-1$
112         parameter.setDescriptionOrResourceKey(
"%AssortmentReportFilter.param.assortment.description" ); //$NON-NLS-1$
113         parameter.setValueClassProvider( new
FixClassProvider( AssortmentProxy.class ) );
114         parameter.setEnumIdentifier( assortmentIdentifier );
115         parameter.setEntityIdentifier( entityIdentifier );
116         parameter.setData( AbstractParameter.KEY_PARAMETER_GROUP,
Messages.getString( VALUE_PARAM_GROUP ) );
117         parameter.setPermissions( new String[] { permission } );
118         return parameter;
119     }
120 }

```

6.5.5.2 Entity Search

Item search is used to find root entities which satisfy arbitrary conditions. Conditions are defined in runtime on root entity fields and sub-entities fields. Search result is a report which contains a list of found root entity ids (see reporting framework for more details) or an array of entity proxies. Item search is in fact a generic API to reporting framework and can (and will) replace most of the named reports.

In HPM 5.3 item search implementation has been redesigned and can be safely used to perform complex search queries over large datasets. Search API is defined in `com.heiler.ppm.search.core` plugin and can be used platform wide.

Search expression

Search conditions are defined using boolean expressions and field operands. Search expression example using pseudo code:

```

(ArticleType.Ean = 9780321349606) AND ( (ArticleLangType.DescriptionShort not
contains "myDescription") OR NOT ( ArticleType.LastModified > "2012.12.12 13:14" ) )

```

Available conditions and operands can be found in java package `com.heiler.ppm.search.core.expression`

Search filters

- ACL filter. By default search result contains only items for which current user has READ ACL permission.
- Alive only filter. By default search results do not contain soft-deleted items

- Arbitrary filter. You can provide report result to define subset of entries to search in. This may improve performance if a subset is small (usefull when searching in a predefined assortment)

Examples

HPM 6.0 API

Starting from HPM 6.0 please use *SearchService* interface instead of *ExecuteSearchEx* utility. Platform wide search instance can be obtained using *EntitySearchComponent* singleton. Consider IoC principal and pass the *SearchService* instance in constructor (or using setter) instead of calling *EntitySearchComponent.getSearchService()* every time you need a search service instance.



Starting from HPM 6.0 you can use *SearchExpressionBuilder* to reduce amount of the boiler-plate code used to create search expression. It provides a kind of internal DSL to build search expression. See java doc for more detail.

Search for an article with ean = 9783898426350 in master catalog

```

1  //repository fields
2  FieldPath catalogFieldPath = new FieldPath( "ArticleType.CatalogProxy" );
3  FieldPath eanFieldPath = new FieldPath( "ArticleType.Ean" );
4
5  //search expression
6  EqualExpression equalCatalogExpression = new EqualExpression( new
7      FieldPathExpression( catalogFieldPath ),
8      new
9      ValueExpression( CatalogProxy.MASTER_CATALOG_PROXY ) );
10 EqualExpression equalEanExpression = new EqualExpression( new
11     FieldPathExpression( eanFieldPath ),
12     new
13     ValueExpression( "9783898426350" ) );
14 Expression rootExpression =
15     equalCatalogExpression.and( equalEanExpression );
16
17 //search parameters
18 SearchParameters searchParameters = new
19     SearchParameters( DataSource.MASTER.getIdentifier(), "ArticleType",
20     rootExpression );
21 ExecuteSearchEx search = new ExecuteSearchEx( searchParameters );
22 search.runWithCoreException( new NullProgressMonitor() );
23
24 //search result
25 ReportResult searchResult = search.getReportResult();

```

Find all ExportTemplates - entity search using classifier field

```

1  EntityType genericDataEntityType =
RepositoryUtils.getRepositoryEntityType( "GenericDataEntityType" );
2  Field classifierField = RepositoryUtils.getRepositoryField(
"GenericDataEntityType.Classifier" );
3  FieldPath classifierFieldPath =
RepositoryUtils.getFieldPath( classifierField );
4
5  EqualExpression equalClassifierExpression = new EqualExpression( new
FieldPathExpression( classifierFieldPath ),
6                                     new ValueExpression(
"ExportTemplate" ) );
7  searchParameters = new SearchParameters( DataSource.MAIN.getIdentifier(),
"GenericDataEntityType", equalClassifierExpression );
8  search = new ExecuteSearchEx( searchParameters );
9  search.runWithCoreException( new NullProgressMonitor() );
10 searchResult = search.getReportResult();

```

Search for article revision 100

```

1  SearchParameters searchParameters = new
SearchParameters( DataSource.MASTER.getIdentifier(), "ArticleType",
2
rootExpression );
3  searchParameters.setRevisionFilter( new
RevisionSearchFilter( RevisionTokenFactory.create( 100 ) ) );
4  ExecuteSearchEx search = new ExecuteSearchEx( searchParameters );
5  search.runWithCoreException( new NullProgressMonitor() );

```

Search for articles in master catalog which have "not available" string in the short description in english or were modified in last 24 hours

```

1  //repository fieldpaths
2  Field shortDescriptionField = RepositoryUtils.getRepositoryField(
"ArticleLang.DescriptionShort" );
3  Field modificationDateField = RepositoryUtils.getRepositoryField(
"ArticleType.LastModified" );
4  FieldPath shortDescriptionFieldPathEN =
RepositoryUtils.getFieldPath( shortDescriptionField );
5  FieldPath lastModifiedFieldPath =
RepositoryUtils.getFieldPath( modificationDateField );
6

```

```

7 shortDescriptionFieldPathEN.getEntityPath()
8         .setLogicalKeyValue( "ArticleLangType.LK.Language", 9L );
9
10 //expression
11 Expression containsShortDescENExpression = new ContainsExpression( new
12     FieldPathExpression( shortDescriptionFieldPathEN ),
13     new
14     ValueExpression( "not available" ) );
15
16 GreaterExpression lastDayModificationExpression = new GreaterExpression(
17     new FieldPathExpression( lastModifiedFieldPath ),
18     new ValueExpression(
19     new Timestamp( System.currentTimeMillis() -86400000 ) ) );
20
21 equalCatalogExpression = new EqualExpression( new
22     FieldPathExpression( catalogFieldPath ),
23     new
24     ValueExpression( CatalogProxy.MASTER_CATALOG_PROXY ) );
25
26 Expression expression =
27     equalCatalogExpression.and( containsShortDescENExpression.or( lastDayModif
28     icationExpression ) );
29
30 //search
31 searchParameters = new
32     SearchParameters( DataSource.MASTER.getIdentifier(), "ArticleType",
33     rootExpression );
34
35 search = new ExecuteSearchEx( searchParameters );
36 search.runWithCoreException( new NullProgressMonitor() );
37
38 //search result
39 searchResult = search.getReportResult();

```

Search for all articles in master catalog filtered by assortment report (improves performance on large catalogs)

```

1 ReportResult assortmentReportResult // = ... create assortment report
2 EntityType articleEntityType = RepositoryUtils.getRepositoryEntityType(
3     "ArticleType" );
4 equalCatalogExpression = new EqualExpression( new
5     FieldPathExpression( catalogFieldPath ),
6     new
7     ValueExpression( CatalogProxy.MASTER_CATALOG_PROXY ) );
8
9 searchParameters = new
10     SearchParameters( DataSource.MASTER.getIdentifier(), "ArticleType",
11     rootExpression );

```

```

7  search = new ExecuteSearchEx( searchParameters,
8                                assortmentReportResult );
9  search.runWithCoreException( new NullProgressMonitor() );
10
11 //search result
12 searchResult = search.getReportResult();

```

Search for all articles in master catalog which are NOT in assortment report

```

1  assortmentReportResult = null; // = ... create report ...
2  articleEntityType = RepositoryUtils.getRepositoryEntityType( "ArticleType"
3  );
4  Field articleInternalIdField = RepositoryUtils.getRepositoryField(
5  "ArticleType.Id" );
6  FieldPath articleInternalIdFieldPath =
7  RepositoryUtils.getFieldPath( articleInternalIdField );
8
9  equalCatalogExpression = new EqualExpression( new
10  FieldPathExpression( catalogFieldPath ),
11  new ValueExpression( CatalogProxy.MASTER_CATALOG_PROXY ) );
12  NotExpression idsNotInAssortmentExpression = new NotExpression( new
13  InExpression( new FieldPathExpression( articleInternalIdFieldPath ),
14  new ValueExpression( assortmentReportResult ) ) );
15  expression = equalCatalogExpression.and( idsNotInAssortmentExpression );
16  searchParameters = new
17  SearchParameters( DataSource.MASTER.getIdentifier(), "ArticleType",
18  rootExpression );
19  search = new ExecuteSearchEx( searchParameters);
20  search.runWithCoreException( new NullProgressMonitor() );
21
22 //search result
23 searchResult = search.getReportResult();

```

Limitations

Search has some limitations.

- Cross datasource search is not supported (i.e. you can't build a single query to search for master and supplier articles simultaneously).
- Only root entities can be searched
- Search operates on EntityType repository level and generic search on Entity level is not possible. However if some kind of entity classifier is defined then you can use it in search expression (see ExportTemplate Example).

- Search is not public API so far. If you are using search api you have to implement junit test to check if results are correct. This is especially true if you are using complex expressions with different logical keys
- Non-standard logical keys may not work in search expressions or deliver wrong results. Such logical keys are not field based i.e. they do not have 'Field Type' attribute in the repository.

Debug

In case of error or wrong search results you can trace the internal search query and search filter by enabling '**com.heiler.ppm.search.server.internal.service**' log category in the server log4j.xml.

Debug search

```
<category name="com.heiler.ppm.search.server">
  <priority value="TRACE"/>
</category>
```

HSQ - Heiler Search Query language

HSQ is a search expression language which can be used to define entity search queries in a textual form (similar to SQL but more domain model oriented). Since it is currently only used in the context of the Rest Service API, its syntax is described in [REST Search Query Language](#) (see page 447).

You can use the HSQParser class to parse a hsq query and build a SearchExpression which can then be used with the SearchService.

HSQ Example

```
1    ...
2
3    ConvertUtils convertUtils = ConvertUtils.getInstance();
4    EntityProxyFactory entityProxyFactory =
5    EntityManagementComponent.getEntityProxyFactory();
6    EntityManagerRegistry managerRegistry =
7    EntityManagementComponent.getManagerRegistry();
8    PermissionService permissionService =
9    EntityManagementComponent.getPermissionService();
10   EntityProxyParser proxyParser = new EntityProxyParser( convertUtils,
11   entityProxyFactory, managerRegistry );
12   RepositoryService repositoryService =
13   RepositoryComponent.getRepositoryService();
14   NavigationService navigationService =
15   RepositoryComponent.getNavigationService();
16   EnumProviderRegistry enumProviderRegistry =
17   EntityManagementComponent.getEnumProviderRegistry();
18   FieldParser fieldParser = new FieldParser( repositoryService,
19   navigationService, convertUtils, enumProviderRegistry, proxyParser,
20   permissionService );
```



```

13 HSQParser hsqParser = new HSQParser( fieldParser, convertUtils );
14
15 String hsq = "Article.EAN in (\"123456789\", \"987654321\",
    \"abcdefghij\") AND not (Article.Identifier = \"abc\" OR
    Article.Identifier equals \"xyz\")";
16
17 Expression expression = hsqParser.parseSearchQuery( hsq );
18
19 SearchParameters searchParameters = new
    SearchParameters( dataSource.getIdentifier(), entityType.getIdentifier(),
    expression );
20 searchParameters.setRevisionFilter( new
    RevisionSearchFilter( revisionToken ) );
21
22 SearchService searchService = EntitySearchComponent.getSearchService();
23
24 ReportResult reportResult = searchService.searchAsReport( progressMonitor,
    searchParameters, reportResultFilter );

```

Search entities using characteristic criteria

When searching for entities based on characteristic records the new Expression `CharacteristicExpression` should be used.

Characteristic expression

A characteristic expression can be used programmatically in the following way:

```

Expression characteristicExp = new
    CharacteristicExpression( entityProxyOfTheCharacteristic );

```

It can then be used in the same manner as any other `BinaryExpression`:

```

Expression expression = new EqualExpression( characteristicExp, valueExp );

```

The result will be all entities which have a **characteristic record** for the characteristic defined by the `characteristicExp` and with the value defined in the `valueExp`.

Example

We want to build a search query which searches for items which contain the value "Flour" in a record for the characteristic "Ingredient".

```

Characteristic ingredient = this.modelService.getCharacteristic("Ingredient"); //
Characteristic with Datatype "Text"
EntityProxy characteristic = ingredient.getEntityProxy();

Expression characteristicExp = new CharacteristicExpression( characteristic );
Expression valueExp = new ValueExpression( "Flour" );
Expression expression = new EqualExpression( characteristicExp, valueExp );

SearchParameters searchParameters = new SearchParameters( DataSource.SUPPLIER,
ArticleConst.ENTITY_TYPE,
                                expression );
EntityProxy[] searchResult = this.searchService.searchAsEntityProxies( new
NullProgressMonitor(), searchParameters,
                                null );

```

We have these articles with characteristic records:

ItemIdentifier	Characteristic	Characteristic record value
Item1	Ingredient	"Salt"
Item2	Ingredient	"Flour"
Item3	Ingredient	"Egg"
Item4	BakingMaterials	"Flour"

The query above will only return **Item2**.

Searching for LookupValues of a characteristic

When searching for Lookup values, a `LookupValueProxy` has to be in the `ValueExpression`.



When using enumeration based lookup values the method `getKey()` returns the `EntityProxy`.

Example

We want to find all articles which have the LookupValue "Wool" in the record for the characteristic "animalIngredient".

```
Characteristic animalIngredients= this.modelService.getCharacteristic(
"animalIngredient"); //Characteristic with Datatype "Lookup"
EntityProxy characteristic = animalIngredient.getEntityProxy();

LookupValue wool = ingredientLookup.getValueByKey( "Wool");

Expression characteristicExp = new CharacteristicExpression( characteristic );
Expression valueExp = new ValueExpression( wool.getKey() );
Expression expression = new EqualExpression( characteristicExp, valueExp );

SearchParameters searchParameters = new SearchParameters( DataSource.SUPPLIER,
ArticleConst.ENTITY_TYPE,
expression );
EntityProxy[] searchResult = this.searchService.searchAsEntityProxies( new
NullProgressMonitor(), searchParameters,
null );
```

We have these articles with characteristic records:

ItemIdentifier	Characteristic	Characteristic datatype	Characteristic record value
Item1	animalIngredient	Lookup	Wool
Item2	Ingredient	Text	"Wool"
Item3	animalIngredient	Lookup	Leather
Item4	BakingMaterials	Lookup	Flour

The query above will only return **Item1**, as Item2 has a different characteristic which uses the datatype String.

Language dependent characteristic records



When not setting a specific language, the search service will search in all languages.

If we want to only search for values in a specific language it is possible to add a language to the `CharacteristicExpression` as a second parameter.

The optional second parameter is a List containing values of the java datatype `Long`. The values represent the languages like defined in the `Enum.Languages` from the repository.

Here is shown how to only get results where the characteristic record has the value "Text" in a specific language.

```
Characteristic shortDescription = this.modelService.getCharacteristic(
    "ShortDescription"); //Characteristic with Datatype "Text"
EntityProxy characteristic = animalIngredient.getEntityProxy();

List< Long > languagesToSearch = new ArrayList<>();
languagesToSearch.add( ( Long ) englishValueByCode.getKey() );

Expression characteristicExp = new CharacteristicExpression( languageDependent,
    languagesToSearch );

Expression valueExp = new ValueExpression( "Text" );
Expression expression = new EqualExpression( characteristicExp, valueExp );
```

We have these articles with characteristic records:

ItemIdentifier	Characteristic	Language	Characteristic record value
Item1	ShortDescription	English	"Text"
Item2	LongDescription	German	"Text"

The query above will only return **Item1**, as Item2 has a the same value but for a different language.

Search values regardless of a specific characteristic

It is also possible to search for values contained in a characteristic record without specifying a concrete characteristics.

This example shows how to see all items that have the LookupValue "Milk" somewhere in their characteristic records.

```
LookupValue milk = IngredientLookup.getValueByKey( "Milk"); //Characteristic with
Datatype "Lookup"
```

```

Expression characteristicExp = new CharacteristicExpression(null);
Expression valueExp = new ValueExpression( milk.getKey() );
Expression expression = new EqualExpression( characteristicExp, valueExp );

SearchParameters searchParameters = new SearchParameters( DataSource.SUPPLIER,
ArticleConst.ENTITY_TYPE,
                                expression );
EntityProxy[] searchResult = this.searchService.searchAsEntityProxies( new
NullProgressMonitor(), searchParameters,
                                                                    null );

```

Setting the first parameter of the `characteristicExpression` to `null` will result in searching for characteristic records which do have the specified value from the `valueExp` in any characteristic.

We have these articles with characteristic records:

ItemIdentifier	Characteristic	Characteristic record value
Item1	Ingredient	Salt
Item2	Ingredient	Milk
Item3	Ingredient	Egg
Item4	BakingMaterials	Milk

The query above will return **Item2** and **Item4** as both records use the same LookupValue.

Descendant Expression

A characteristic expression can be used programmatically in the following way:

```

Expression descendantExp = new DescendantExpression( parentExpression,
childExpression );

```

Note that there is also a constructor with multiple childExpressions available.

The result will be all entities which have **characteristic records** matching the parent expression as well as the child expression in the same hierarchy. The following example will make this clearer.

Example

In this example items are composed of different ingredients from different countries. To model this there is a superordinate characteristic called ingredient and a subordinate characteristic called Country of origin.

We want to find all items containing sugar from Brazil. If we used the characteristic expressions from above and connect them with and, we'd not only find the items with sugar from Brazil but also items that contain sugar from India and meat from Brazil. In order to search for multiple values in a direct hierarchy we can use the descendant expression.

```
//Superordinate characteristic with data type "Lookup"
Characteristic ingredientCharacteristic = this.modelService.getCharacteristic(
"Ingredient" );
EntityProxy ingredient = ingredientCharacteristic.getEntityProxy();
LookupValue sugar = ingredientsLookup.getValueByCode( "SUGAR" );

Expression ingredientExp = new CharacteristicExpression( ingredient );
Expression sugarExp = new ValueExpression( sugar.getKey() );
Expression parentExpression = new EqualExpression( ingredientExp, sugarExp );

//Subordinate characteristic with data type "Lookup"
Characteristic originCharacteristic = this.modelService.getCharacteristic(
"CountryOfOrigin" );
EntityProxy origin = originCharacteristic.getEntityProxy();
LookupValue brazil = originCountriesLookup.getValueByCode( "BRAZIL" );

Expression originExp = new CharacteristicExpression( origin );
Expression brazilExp = new ValueExpression( brazil.getKey() );
Expression childExpression = new EqualExpression( originExp, brazilExp );

Expression descendantExpression = new DescendantExpression( parentExpression,
childExpression );

SearchParameters searchParameters = new SearchParameters( DataSource.MASTER,
ArticleConst.ENTITY_TYPE,
descendantExpression );

EntityProxy[] searchResult = this.searchService.searchAsEntityProxies( new
NullProgressMonitor(), searchParameters,
null );
```

We have these items with characteristic records:

Item Identifier	Characteristic	Lookup of Characteristic	Characteristic record value
Item1	Ingredient	Ingredients	Sugar

Item Identifier	Characteristic	Lookup of Characteristic	Characteristic record value
Item1	CountryOfOrigin	OriginCountries	Thailand
Item2	Ingredient	Ingredients	Sugar
Item2	CountryOfOrigin	OriginCountries	India
Item2	Ingredient	Ingredients	Meat
Item2	CountryOfOrigin	OriginCountries	Brazil
Item3	Ingredient	Ingredients	Sugar
Item3	CountryOfOrigin	OriginCountries	Brazil

The query above will return **Item3**.

6.5.6 Data Navigation

6.5.6.1 Entity Relations

6.5.7 Data Security

6.5.8 Command Framework

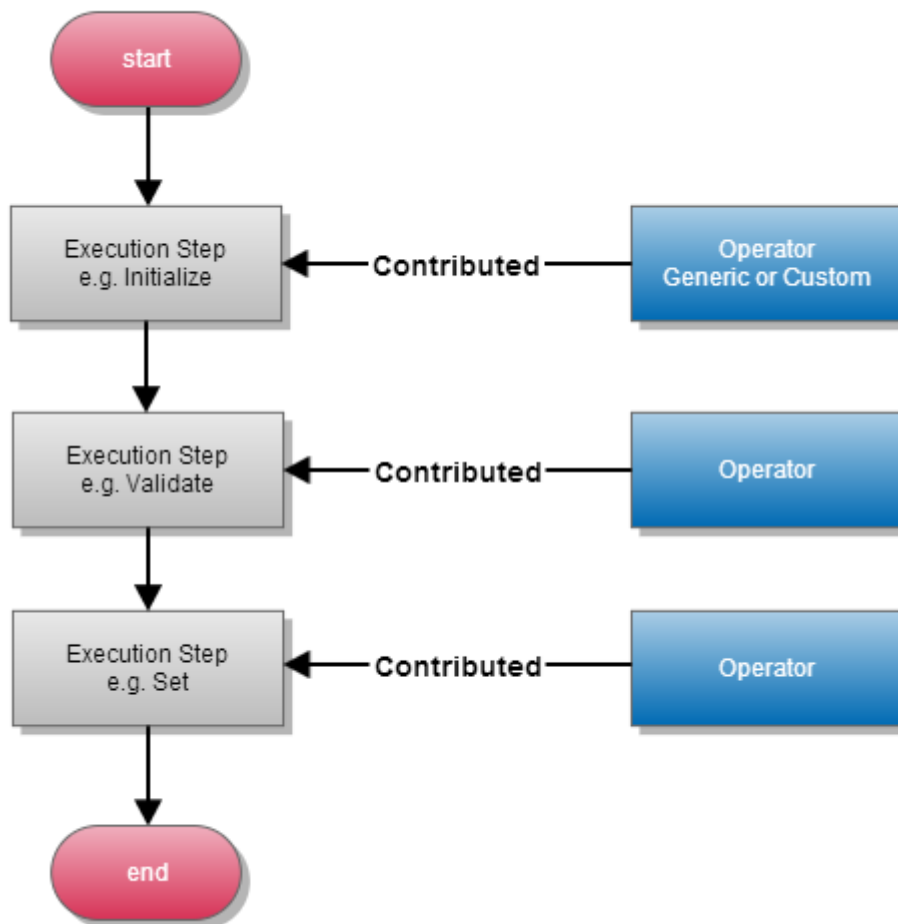
- [General](#) (see page 164)
 - [Command Context](#) (see page 165)
 - [Feedback Processors](#) (see page 165)
- [Commands](#) (see page 166)
- [Operators](#) (see page 166)

6.5.8.1 General

The command framework provides an extensible command pattern based on the template design. The commands can be seen as templates providing a static workflow using single, atomic operations as execution step. Usually, the commands fit for nearly all data modification tasks we have. You can create new data objects, add or remove them to/from their parent, you can set single properties of a data object and you can trigger separate validation rules. Additionally to this, even new commands can be contributed to the framework - either for new purposes (e.g. import or merge command), or as a different implementation for specific entities.

The amount of flexibility comes not without the cost of complexity. A

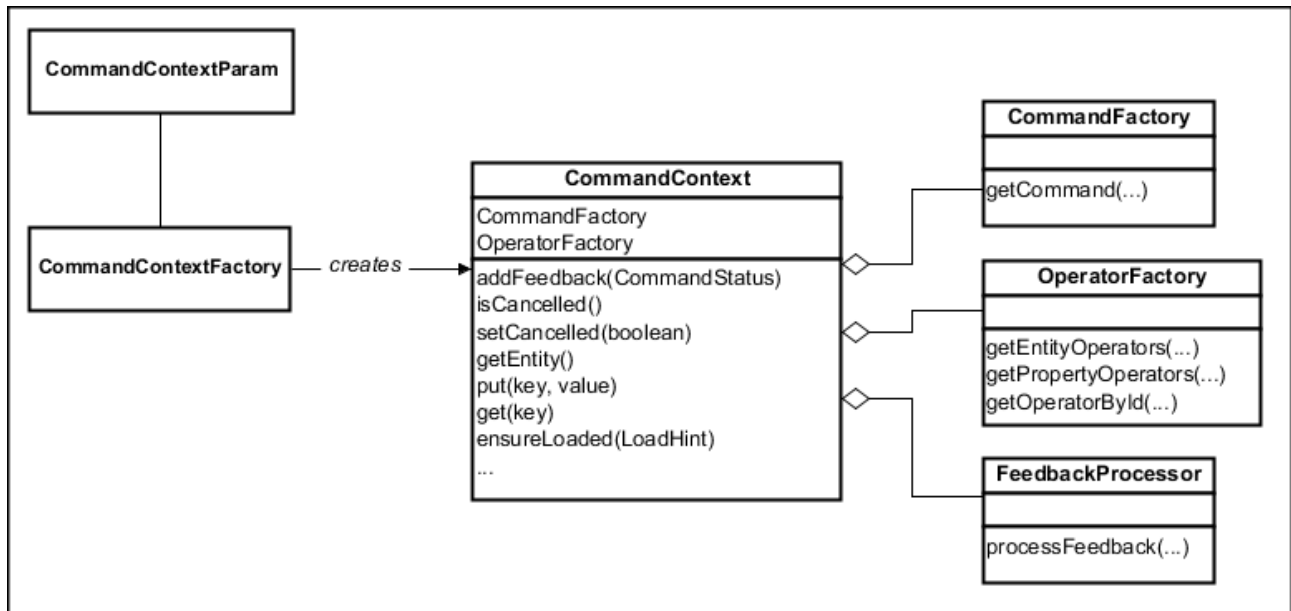
"simple" `article.setDescriptionShort(english, "my description")` is currently not provided by the platform, but we encourage you to build your own data access objects which themselves use the generic command framework.



Every operator which is part of a command follows a well known interface. This interface always provides access to the `CommandContext` and additional parameters which might be helpful to fulfill the tasks the operator is designed for. Some operators are contributed by default from the command framework, others have no default implementation and are usually only for entity or customizing specific implementations.

In case a `CommandContext` is cancelled, the following operators will not be executed and the command execution is aborted.

Command Context



The `CommandContext` is the central interface of the command framework. Clients must obtain an instance from the `CommandContextFactory` using the `CommandContextParam` object to parameterize the context. By using the `CommandContext` you can obtain the `CommandFactory` as well as the `OperatorFactory` which provides you the actual command and operator instances for an Entity or Field.

Feedback Processors

All operators have access to the `CommandContext` and can - at any time - provide feedback to the framework. Feedback can be an error or a warning. To provide feedback you need to create a `CommandStatus` object and call the corresponding `addFeedback` method. Depending on the severity of the `CommandStatus` the `FeedbackProcessor` will decide if the command context should be cancelled or not.

In general there are two kinds of `FeedbackProcessor` implementations possible. The one supports and enforces to have the possibility to interact with a user (e.g. through a dialog), the other doesn't. For example, when editing in the client's table view, the feedback processor which is used `needsUserInteraction` since it opens an error or warning dialog (depending on the severity of the command status). On the other hand, during the automated execution of an import or merge, we can not expect to have the possibility of a user interaction during the command run - therefore the used processor will only log the feedback and - in case of a warning - will always assume a "yes" - thus warnings will not cancel the command status.

You can define your own `FeedbackProcessor` in the `CommandContextParams` in case you're calling the command by yourself. Other areas in the system like Jobs (e.g. Import, Merge) use their own feedback processor implementation which can not be replaced as the job logic relies on them.

6.5.8.2 Commands

Platform commands are all commands which are used by the core platform. They build the fundamental set of data modification commands all other parts of the application use.

Create Command

Creates a new data object and initializes all its direct properties with the default values from the repository.

Add Command

Adds a data object to its parent data object. Prior before doing this it executes several validation operators including the validation for duplicates.

Put Command

Typically used to set values of single fields of a data object. But in contrast to only allow a simple set operation on a field, it also is able to navigate through the object model to find the correct data object on which to set a value. This operation is called put, of course it's far more complex.

Validate Command

Validates a single data object and its children. The default operators for this command only validate for mandatory fields and mandatory children. Field length and other field based validations are performed during the put or set operation of the PutCommand and must not be executed again. Customizing can contribute additional validation.

Remove Command

Removes a data object from its parent. Before that, validations are executed. For example one could check if it is allowed to remove the "last price" of an item. By default there are no extra validations implemented for this command.

6.5.8.3 Operators

Most operators are used in multiple commands (like the `PropertyInitializer` which is used in the `AddCommand` as well as the `CreateCommand`)

Operators are a highly generic concept. The command implementations are free to decide which kind of operators they want to use. There is no common base interface for all operators at all. So how can you contribute your own operator? For this you need the operator's type information which is either documented here, or in the class javadoc of the operator interface. The operator type has meta information which are documented as follows:

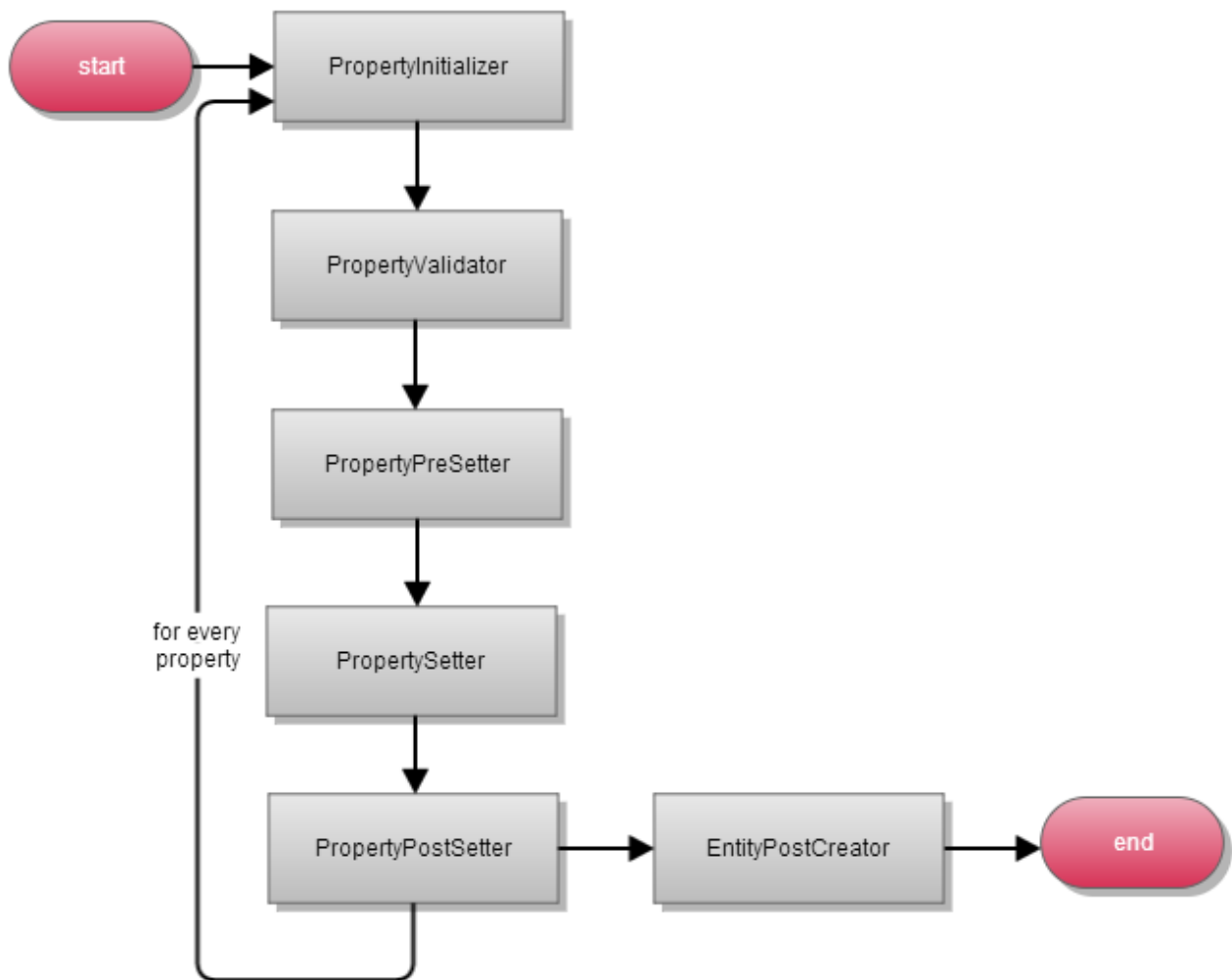
- **Type Identifier**
Unique identifier of the operator type, usually equals to the simple class name of the operator interface.

- **Allows Combination**
When yes, this signals that the command can handle multiple contributions for the same property, but on different repository levels. For example: You could then contribute an operator for the field type, e.g. `ArticleLangType.DescriptionShort` as well as for the field (custom area in the repository) `ArticleLang.DescriptionShort`. In this case, both operators will be executed in order of their repository level. That is: Type elements before Custom elements, Entities before Fields.
- **Allows Multiple**
When true, multiple contributions for **the same** repository identifier are possible. The operators will then be executed in the order of their `order` attribute
- **Generic Operator Identifier**
The identifier of the generic implementation of the operator (generic means, no repository specific contribution, thus this operator will be executed for every field/entity)

The list of supported operators can be found on each command page.

6.5.8.4 Create Command

Creates a new data object and initializes all its direct properties with the default values from the repository.



Operators

Type Identifier	Allows Combination	Allows Multiple	Generic Operator Identifier
PropertyInitializer	no	yes	n/a
<p>Initializes an item property for command usage. Typical usage is to manipulate/resolve the <i>property value</i> based on the actual item. For example, the main supplier which has an artificial ID of -10 will be resolved by the actual main supplier of the item using this operator. This operator should <i>not</i> modify the data object.</p>			

Type Identifier	Allows Combination	Allows Multiple	Generic Operator Identifier
PropertyValidator	yes	yes	ppm.std.operator.propertyValidator
<p>Validates a single item property. The generic default implementation validates for everything which is defined in the repository. Lengths, ranges, enumerations, lower- and upperbound etc. Since multiple contributions on multiple levels might be available for the same property, implementors are <i>not allowed to change the value of the item property during a validation</i>. Other validators would not be able to work correctly in case the value changes depending on the order of validators. If you need to modify the value, you can do this with the PropertyInitializer operator.</p>			
PropertyPreSetter	yes	yes	n/a
<p>Called before an item property will be set. Modification of the property value or the data object are possible here.</p>			
PropertySetter	no	no	ppm.std.operator.propertySetter
<p>Sets the value of an item property to the data object's corresponding attribute using generic eSet methods on the EMF data objects.</p>			
PropertyPostSetter	yes	yes	n/a
<p>Called after an item property has been successfully set</p>			
EntityPostCreator	yes	yes	n/a
<p>Is called after all properties have been set to the data object. Can be used to perform entity validations which need to have access to multiple property values.</p>			

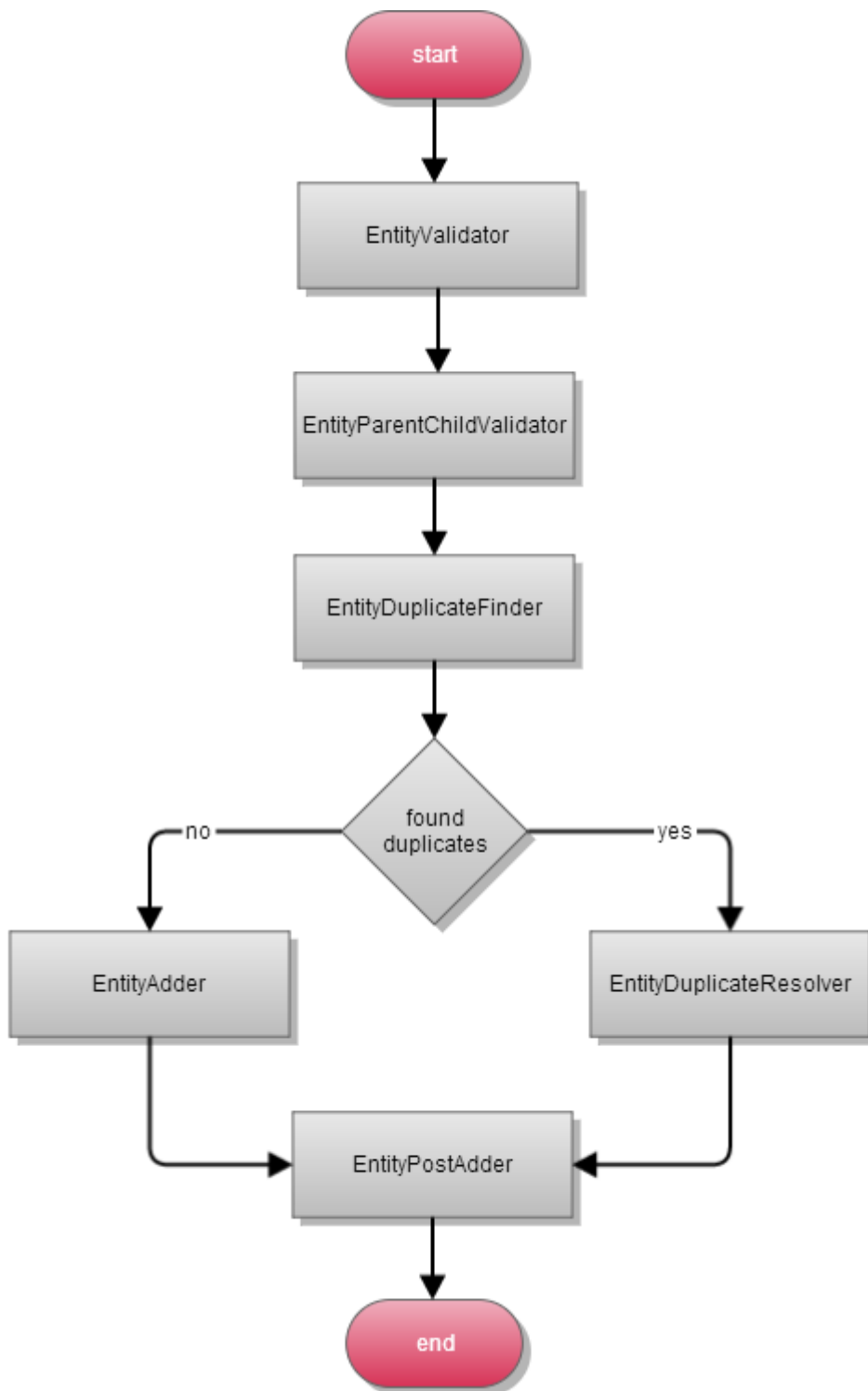
Use Cases

The create command is used by the `ImportCommand` as well as the `PutCommand` (see page 173) . Please see the respective documentation of these command for details.

The desktop client calls the `CreateCommand` as soon as you click on the "add" button in a table view to create a new row.

6.5.8.5 Add Command

Adds a data object to it's parent data object. Prior before doing this it executes several validation operators including the validation for duplicates.



Operators

Type Identifier	Allows Combination	Allows Multiple	Generic Operator Identifier
EntityValidator	yes	yes	ppm.std.operator.entityValidator
<p>Validates a single entity data object. Implementations should not validate the children of a data object by themselves.</p> <p>A modification of the data object is not allowed in this operator, since subsequent operators might rely on the original value and the order of operators might not always be guaranteed</p>			
EntityParentChildValidator	yes	yes	ppm.std.operator.entityParentChildValidator
<p>Validates the parent/child relationships. For example, this operator might validate if it is allowed to add a second price to an item, etc. So it usually validates the cardinality of children for a parent.</p>			
EntityDuplicateFinder	no	no	ppm.std.operator.entityDuplicateFinder
<p>Finds any duplicate records before the data object is added to its parent. Depending on the results of this operator either the <code>EntityAdder</code> or the <code>EntityDuplicateResolver</code> will be called.</p> <p>The generic implementation uses <code>LogicalKeyEvaluator</code> instances to search for a duplicate in the parent object.</p>			
EntityAdder	no	no	ppm.std.operator.propertySetter
<p>Adds the child to the parent by either setting it as value of the corresponding attribute (1:1 relationships), or by adding it to the corresponding list of the parent (1:n relationships)</p>			

Type Identifier	Allows Combination	Allows Multiple	Generic Operator Identifier
EntityDuplicateResolver	no	no	ppm.std.operator.entityDuplicateResolver
<p>Resolves the duplicates which have been found by the EntityDuplicateFinder . The generic implementation will create a corresponding command feedback which usually aborts the command execution.</p> <p>Other implementations might override the duplicate or merge the values, etc.</p>			
EntityPostAdder	yes	yes	n/a
<p>Executed after the child has been added to the parent. The post adder will be called after the EntityAdder and also after the EntityDuplicateResolver - unless the command context has been canceled before. (Which the default implementation of the resolver does!)</p>			

Use Cases

The ImportCommand uses the AddCommand to add new child objects to their parent.

The desktop client calls this command when you click on the save button in a detail table view.

6.5.8.6 Put Command

Typically used to set values of single fields of a data object. But in contrast to only allow a simple set operation on a field, it also is able to navigate through the object model to find the correct data object on which to set a value. This operation is called put , of course it's far more complex.

Set Operation

The set operation applies a single value of a property to the data object it directly belongs to. No hierarchical navigation is performed. The property value is validated and applied with additional permission checks and custom operators to perform your own logic.

Operators

Type Identifier	Allows Combination	Allows Multiple	Generic Operator Identifier
EntityPermissionChecker	yes	yes	ppm.std.operator.entityPermissionChecker
Checks permissions on field or entity level. The generic implementation validates against the qualified field permissions (since object permissions are validated in the detail model and action rights in UI)			
PropertyInitializer	no	yes	n/a
<p>Initializes an item property for command usage. Typical usage is to manipulate/resolve the <i>property value</i> based on the actual item.</p> <p>For example, the main supplier which has an artificial ID of -10 will be resolved by the actual main supplier of the item using this operator.</p> <p>This operator should <i>not</i> modify the data object.</p>			
PropertyValidator	yes	yes	ppm.std.operator.propertyValidator
<p>Validates a single item property. The generic default implementation validates for everything which is defined in the repository. Lengths, ranges, enumerations, lower- and upperbound etc.</p> <p>Since multiple contributions on multiple levels might be available for the same property, implementors are <i>not allowed to change the value of the item property during a validation</i>.</p> <p>Other validators would not be able to work correctly in case the value changes depending on the order of validators. If you need to modify the value, you can do this with the <code>PropertyInitializer</code> operator.</p>			
PropertyPreSetter	yes	yes	n/a
Called before an item property will be set. Modification of the property value or the data object are possible here.			

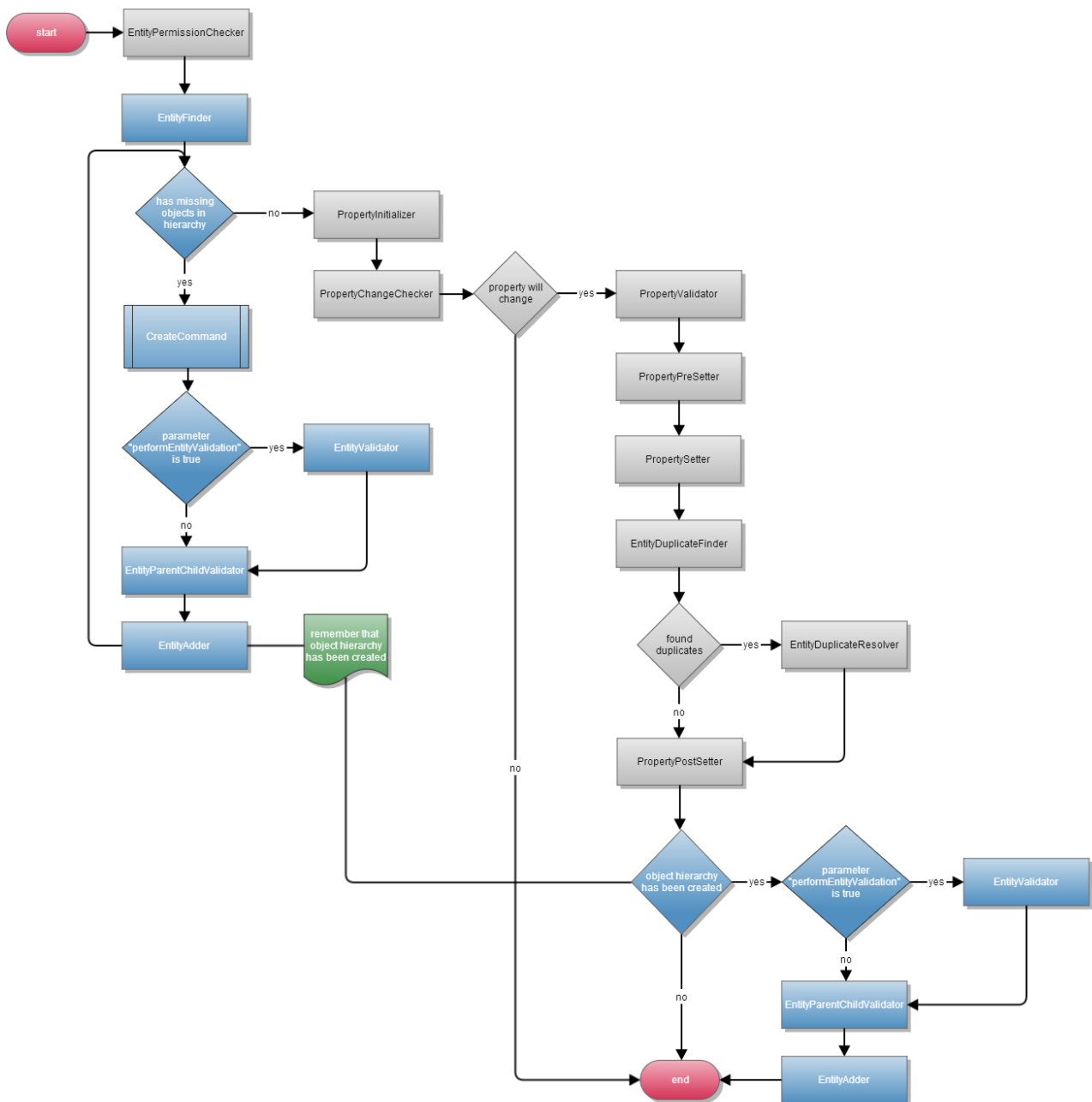
Type Identifier	Allows Combination	Allows Multiple	Generic Operator Identifier
PropertySetter	no	no	ppm.std.operator.propertySetter
Sets the value of an item property to the data object's corresponding attribute using generic eSet methods on the EMF data objects.			
PropertyPostSetter	yes	yes	n/a
Called after an item property has been successfully set			
EntityPostCreator	yes	yes	n/a
Is called after all properties have been set to the data object. Can be used to perform entity validations which need to have access to multiple property values.			

Use Cases

Put Operation

The put operation first tries to navigate to the correct target data object inside of the hierarchy. For this it uses the `EntityFinder` operator, which itself will use `LogicalKeyEvaluator` implementations to find the correct records based on the logical keys of the entity types which are "on the way" down the hierarchy.

The `PutCommand` will validate input values automatically based on the generic settings of the repository. This included length checks for text fields as well as range checks for numeric or date fields. It checks if the value is part of an enumeration as well as if the value is mandatory or not. The mandatory field check will also be executed for field values which have not (yet) been applied by calling the put operation - unless this has been disabled when calling the `PutCommand`. This might be a problem in case you want to execute multiple put commands on the same data object since the first put operation will then return an error feedback indicating that the value of a mandatory field is missing (namely the value of the other mandatory fields). For these scenarios you need to disable this check in the `PutCommand` and execute the `ValidateCommand` after you're finished with the `PutCommands`, or disable it for all calls but the last one.



Operators

Type Identifier	Allows Combination	Allows Multiple	Generic Operator Identifier
EntityPermission Checker	See Set Operation (see page 173)		

Type Identifier	Allows Combination	Allows Multiple	Generic Operator Identifier
PropertyInitializer	See Set Operation (see page 173)		
PropertyValidator	See Set Operation (see page 173)		
PropertyPreSetter	See Set Operation (see page 173)		
PropertySetter	See Set Operation (see page 173)		
PropertyPostSetter	See Set Operation (see page 173)		
EntityPostCreator	See Set Operation (see page 173)		

Use Cases

6.5.8.7 Validate Command

Validates a single data object and its children. The default operators for this command only validate for mandatory fields and mandatory children. Field length and other field based validations are performed during the put or set operation of the `PutCommand` and must not be executed again. Customizing can contribute additional validation.

6.5.8.8 Remove Command

Removes a data object from it's parent. Before that, validations are executed. For example one could check if it is allowed to remove the "last price" of an item. By default there are no extra validations implemented for this command.

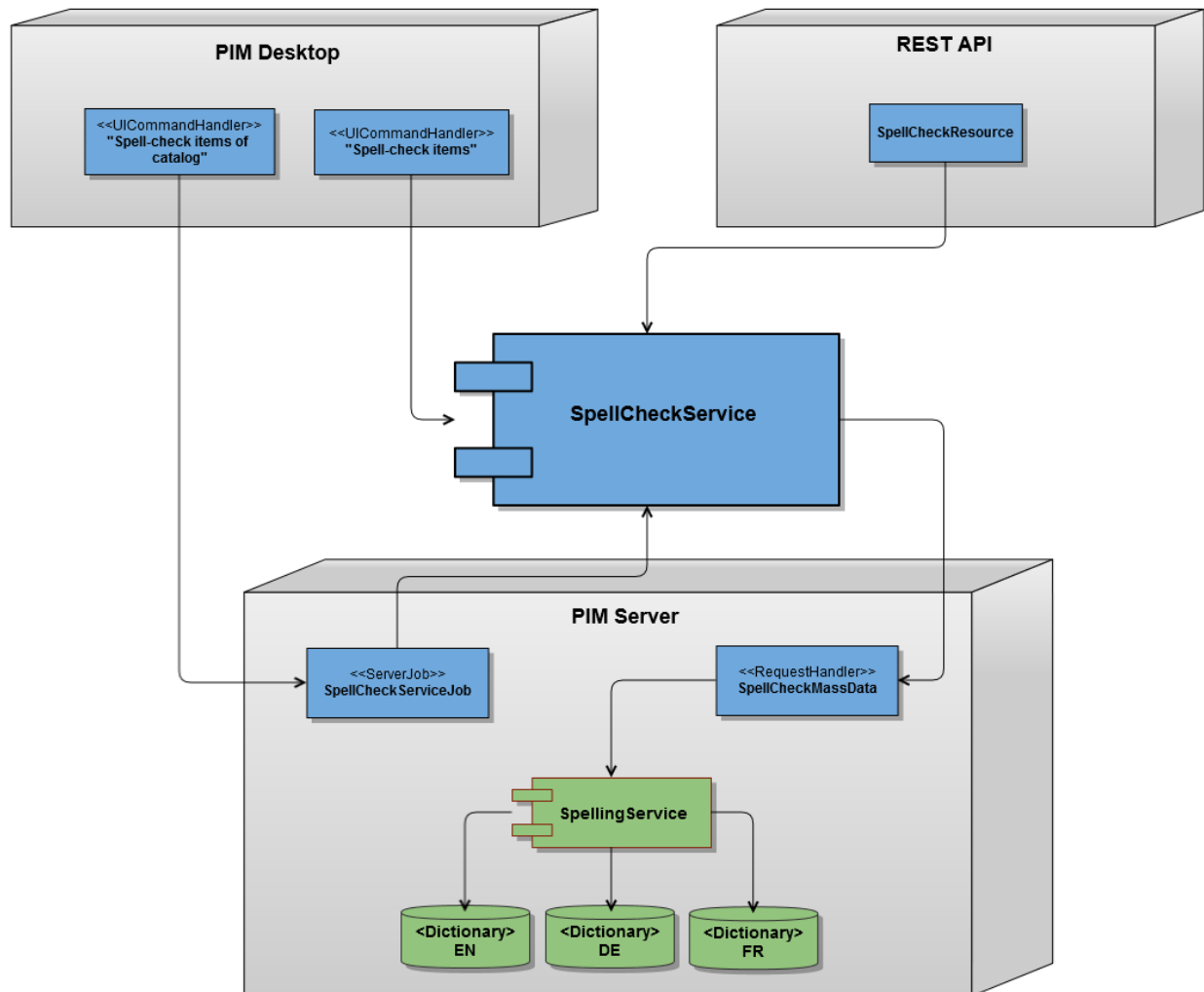
6.5.9 Spellchecking API

6.5.9.1 General

With the Spellchecking API you can spell-check mass data using a service which can be performed in the background (e.g. as a server job). The *SpellCheckService* supports spell-checking of the plain texts as well as *EntityItems*. Depending on the input for the spell-checking you get also a corresponding result with the objects with the incorrect words. The *SpellCheckService* is placed in PIM Core so it can be used on the server as well as on the client, in PIM Desktop as well as in PIM Web. The service delegates all requests to the corresponding *RequestHandler* on the server, so the spell-checking itself will be always performed server-side. See also the architecture picture below.

6.5.9.2 Architecture

SpellCheckService Architecture



The *SpellCheckService* uses the *RequestHandler* to spell-check the input data on the PIM Server. The *RequestHandler* again uses for each object to spell-check the already existed *SpellingService* (which is also used from the UI to spell-check a text from the text field "on the fly" when editing). So the *SpellCheckService* uses the same *Dictionaries* as the spell-checking process in the UI.

6.5.9.3 Usage

Plain texts

The simplest usage of the *SpellCheckService* is to the spell-check one or many plain texts:

```
String[] textsToCheck = { "Text1", "Text2", "Text3" };
```

```

SpellCheckService spellCheckService = SpellCheckComponent.getSpellCheckService();
TextSpellCheckServiceResult result = spellCheckService.check( progressMonitor,
ThreadLocale.getDefault(),
                                                                    textsToCheck );

```

As result you receive an instance of type *TextSpellCheckServiceResult*. It provides methods to determine whether every single text is spelled correctly. You can also get the *ResultDetails* for each incorrect text which contains all incorrect words of the text (including the start and the end index of each incorrect word relative to the whole text).

EntityItems

For more advanced usage there are methods to spell-check values of specific fields of one or many *EntityItems*:

```

EntityProxy[] entityProxies = new EntityProxy[] { article1, article2, article3 };
EntityItemList itemList = new ArrayEntityItemList( articleEntity, entityProxies );
FieldPath[] fieldPaths = getFieldPathForSpellChecking();
SpellCheckService spellCheckService = SpellCheckComponent.getSpellCheckService();
RevisionToken revisionToken = RevisionContext.getInstance()
                                                                    .getRevisionToken();
EntityItemSpellCheckServiceResult result = spellCheckService.check( progressMonitor,
revisionToken,
ThreadLocale.getDefault(), itemList, fieldPaths );

```

NOTE: the *FieldPaths* using for the spell-checking should be full-qualified and contain the current revision, otherwise the spell-checking can not be performed properly.

As result you receive an instance of type *EntityItemSpellCheckServiceResult*. It provides methods to get all entityItems which contain incorrect texts. You can also get for each item and for each fieldPath the corresponding *ResultDetails* which contains all incorrect words (including the start and the end index of each incorrect word relative to the whole text).

SpellCheckServiceJob

If you want to run the spell-checking process as a server job, use the corresponding *SpellCheckServiceJob* instead of using the *SpellCheckService* directly:

```

RevisionToken revisionToken = RevisionContext.getInstance()
                                                                    .getRevisionToken();
SpellCheckServiceParam param = new SpellCheckServiceParam( null, revisionToken,
itemList, fieldPaths );
Serializable2DomPersistableAdapter serverJobParam = new
Serializable2DomPersistableAdapter( param );
ServerJobScheduler.getInstance()
                                                                    .schedule( "SpellCheckService", serverJobParam, null, null, null );

```


The result of the spell-checking process you can see in the corresponding process overview category "Spellcheck". All items with incorrect words will be listed in the problemLog of the corresponding server job.

REST API

The *SpellCheckService* can be also used via REST Service API. Currently only one REST support for the spell-checking is implemented: spell-checking of the plain text. So you can send a POST request using specific URL including the text which should be spell-checked as a request body:

The screenshot shows a REST client interface with the following details:

- Environment:** Normal, Basic Auth, Digest Auth, OAuth 1.0, No environment (selected).
- URL:** http://DEW182131:1501/rest/V1.0/manage/spellcheck/en
- Method:** POST
- Form Data:** form-data, x-www-form-urlencoded, raw (selected), Text (selected).
- Request Body (raw):**

```

1 Representational sttae transfer (REST) is a software architectural style consisting of a coordinated
2 set of architectural constraints applied to components, connectors, and data elements, within a distributed
3 hypermedia system. REST ignores the details of component implementation and protocol syntax in order to
4 focus on the roels of components, the constraints upon their interaction with other components, and their
5 interpretation of significant data elenents.
6

```
- Buttons:** Send, Preview, Add to collection, Reset.
- Response:**
 - Status:** 200 OK
 - Time:** 53 ms
 - Body (Pretty):**

Representational **sttae** transfer (REST) is a **software** architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system. REST ignores the details of component implementation and protocol syntax in order to focus on the **roels** of components, the **constraints** upon their **interaction** with other components, and their interpretation of significant data **elenents**.

As result you get a HTML string which contains the input text including the highlighting HTML-tags around the incorrect words.

For the future also other REST support for the spell-checking is proposed. Fill free to inform the PIM dev team which kind of REST API for the spell-checking would be useful.

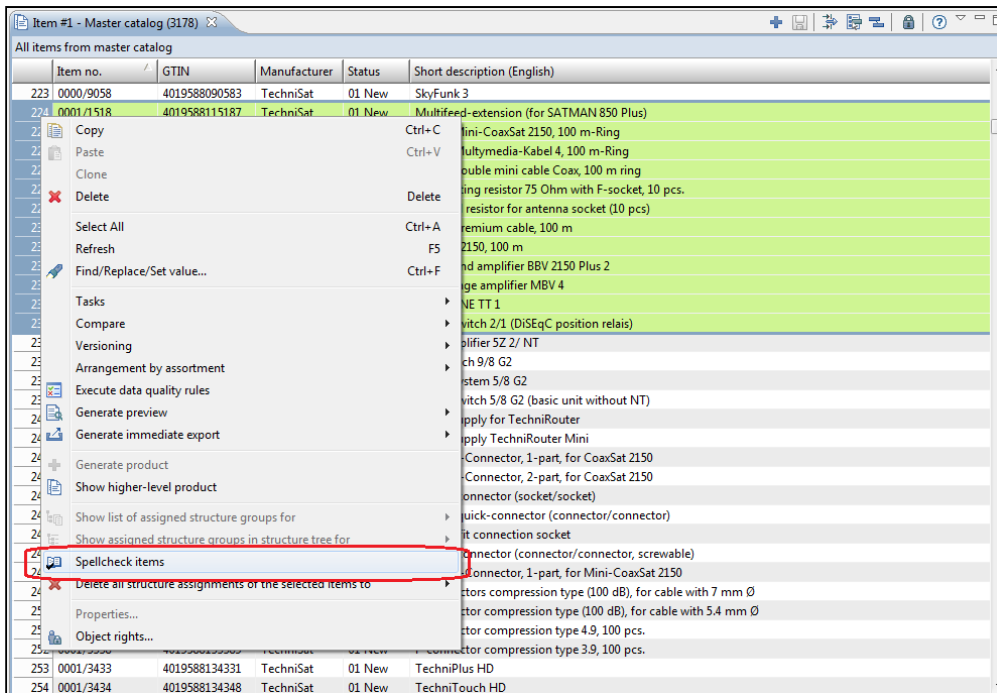
6.5.9.4 Examples

As examples of the usage of the *SpellCheckService* there are two UI commands which are contributed for several views and can be performed from the user to spell-check corresponding entity items.

Both examples can be found in the SDK examples:

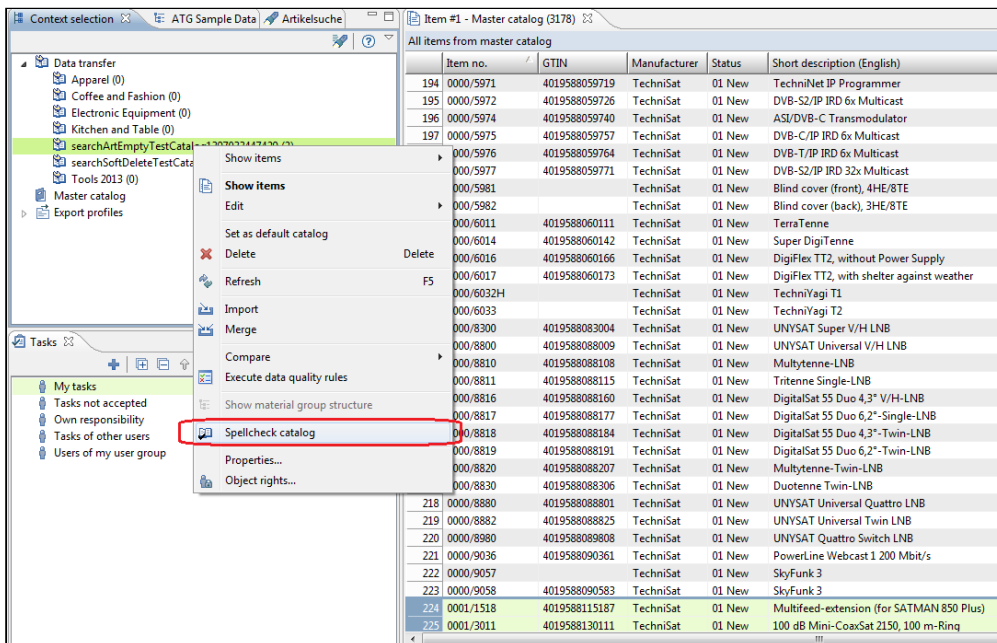
sdk\examples\customizing\com.heiler.ppm.customizing.spelling.ui

Spellcheck items



The command handler for this menu contribution uses the *SpellCheckService* directly and let spell-check the short and long descriptions of all selected items. The result will be shown as a *ProblemLog-Dialog*.

Spellcheck catalog



The command handler for this menu contribution performs a server job - *SpellCheckServiceJob* - and let spell-check the short and long descriptions of all items from the selected catalog. The result can be viewed in the process overview.

6.6 Data Audit

Detailed information about the extension points and facilities for auditing data changes. It distinguishes between an Audit Log and an Audit Trail.

The audit log functionality stores the time and user of creation, modification and deletion of objects. This is stored for the root item in a channel specific way, and for every database table also.

The Audit Trail functionality stores also the value which has been changed in a historical way. So you can see not only who and when the record has been changed, but also what has been changed (which column), what was the value before etc.

6.6.1 Audit Trail

With Version 10.1 a completely new audit trail feature has been introduced which fully replaces the old audit trail facilities. Old and previously deprecated APIs have been removed and a new persistence layer has been created for the audit trail feature.

See the Knowledge Base for more general overview of the feature.

The new Audit Trail architecture is tightly integrated in the persistence layer of Product 360. For every create, modify or delete operation an EntityItemChange Document will be created and stored within the Elastic document storage.

Please see the Configuration Manual for details on how the Audit Trail feature can be configured.

The Migration Guide on details about the migration job which reads the old audit trail data from the relational database and moves it to the new storage.

There are a few [Audit Trail Extension points](#) (see page 187) which allow a customizing of the feature.

6.6.2 Audit Log

Audit logging is an important feature in an application which has a lot of users and data transaction every day. Providing delta reports for 3rd party systems like web-shops was one of the main functional focuses on the audit log released with the version 5.3. Therefore we enhanced the complete [Persistence Model](#) (see page 216) and provided also a new extension point specifically designed to be implemented by consulting in order to be prepared for the future and be most flexible in this area.

6.6.2.1 Physical audit log

Nearly all persistence objects support the `Auditable` (see page 183) base class (starting with the version 5.3). Therefore all the corresponding database tables have been enhanced accordingly (please see [Persistence Model](#) (see page 216) for details). The persister framework checks every persistence object which is going to be saved in the database and updates the creation/modification timestamps automatically. Clients must do just nothing for this feature except supporting the needed columns in their persistence tables.

6.6.2.2 Logical audit log

The requirements for logical audit logging of root entities is being driven by use cases like "Show all items which have been created by user 'abc' since yesterday", or "Show all items which have been modified since last week". To be able to fulfill these requirements we need to store audit log data not only in every table (like with the physical audit log), but also on the business root object. Unless that, we would need to join nearly all tables for a specific entity item so we can determine if the item has been changed. (Maybe only a price has changed, which would not be reflected in a change of the Article (or ArticleRevision) table. Additionally to that, the definition of "changed, created and deleted" may differ depending on specific scenarios which use this information. For example, regarding an ERP replication process only certain fields may be relevant for replication - therefore as seen from the ERP system, an object has been changed when one of those fields change only.

Up from the version 5.3 we returned from the approach of storing this data in a generic table, since in fact, the Product 360 server has no dynamic [Persistence Model](#) (see page 216) - we only have a dynamic [Business Model](#) (see page 183). Introducing a generic model which stores data for multiple different entity types was confusing and error prone, thus we refactored this part in 5.3 again.

AuditLogWriter

The persister calls the `AuditLogWriter` (see page 183) after the persistence object has been modified by the mediators, but before flush is being called on the `JPA EntityManager`. With this we ensure two things:

- Limit the number of database round trips and provide hibernate with the possibility to streamline the SQL statements it sends to the database.
- Make sure that the audit log records are written in a synchronous and transaction safe manner. Thus we make sure that the audit log records are not written in case the entity item can not be saved due any reason.



It's currently not possible to customize the `AuditLogWriter` itself, please use the `AuditLogProvider` for this

AuditLogProvider (deprecated)



Deprecation Notice

Starting with Version 8.1 users can configured a list of fields (optionally qualified) for the channel entity directly in the Desktop UI's channel management. The `AuditLogProvider` extension point is deprecated starting with version 10.1 and we do not recommend to use it any longer. Customers should use channels and their field configuration for this purpose.

Since direct interaction with the persistence layer is not a recommended customizing, we provide an extra interface which is specifically designed to be customized. The `AuditLogProvider` (see page 183) interface provides methods to implement special custom logic. The `AuditLogWriter` will read the current audit log record and calls then the `AuditLogProvider` which itself is responsible to set the creation/modification/deletion user and timestamp in the given `AuditLogRecord` (see page 183) object.

The `AuditLogProvider` implementation however **must not** save this record to the database in any case! It's the `AuditLogWriter` which is responsible to build the bridge between `Persister` and `AuditLogProvider` and will save the `AuditLogRecord` in the most effective way. Please note that the `AuditLogProvider` interface also provides methods for bulk modifications of entity items. These methods will be called in case of bulk updates which occur in deletion scenarios mostly.

An entity item can have multiple logical audit log records which are distinguished by the classifier field. So you could have an audit log record specific for different export channels like ERP system or Web-Shop etc.

`AuditLogProviders` can be contributed with the `com.heiler.ppm.persistence.auditLog` extension point. If you want to have multiple audit log records, you will need to contribute also an appropriate classifier in this extension point.

In case you have multiple classifiers and therefore would have multiple audit log providers, we recommend to implement the `MultiAuditLogProvider` interface which makes it possible to provide all audit log values for all those classifiers in one call - this significantly improves performance depending on the needed logic in the audit log provider.

Sample implementation for MultiAuditLogProvider

```

1  package com.heiler.ppm.persistence.server.audit;
2
3  import java.util.Collection;
4  import java.util.Set;
5
6  import org.apache.commons.logging.Log;
7  import org.apache.commons.logging.LogFactory;
8  import org.eclipse.emf.ecore.sdo.EDataObject;
9
10 import com.heiler.ppm.persistence.server.mediator.BusinessChangeSummary;
11 import com.heiler.ppm.repository.EntityType;
12 import com.heiler.ppm.repository.core.RepositoryComponent;
13 import com.heiler.ppm.repository.core.RepositoryService;
14 import com.heiler.ppm.revision.common.RevisionToken;
15
16 public class CustomMultiAuditLogProvider extends
17     MultiAuditLogProviderBaseImpl
18 {

```

```

18     private static final Log log =
LogFactory.getLog( CustomMultiAuditLogProvider.class );
19
20     private RepositoryService repositoryService;
21
22     public CustomMultiAuditLogProvider()
23     {
24         this.repositoryService = RepositoryComponent.getRepositoryService();
25     }
26
27     @Override
28     public void onCreate( AuditLogData logData, Collection< AuditLogRecord >
logRecords, RevisionToken revisionToken,
29                     BusinessChangeSummary changeSummary, EDataObject
businessObject )
30     {
31         if ( doTrigger( changeSummary ) )
32         {
33             log.info( "trigger onCreate( )" );
34         }
35     }
36
37     private boolean doTrigger( BusinessChangeSummary changeSummary )
38     {
39         boolean doTrigger = true;
40         EntityType mediaAssetDocumentEntityType = this.repositoryService.getEn
tityTypeByIdentifier( "MediaAssetDocumentType" );
41         Set< EDataObject > mediaAssetDocuments =
changeSummary.getDataObjectsForEntityType( mediaAssetDocumentEntityType );
42
43         for ( EDataObject mediaAssetDocument : mediaAssetDocuments )
44         {
45             String quality = mediaAssetDocument.getString( "quality" );// find
"object name" in types area of repository
46             if ( "web".equals( quality ) )
47             {
48                 doTrigger = false;
49                 break;
50             }
51         }
52         return doTrigger;
53     }
54
55     @Override
56     public void onChange( AuditLogData logData, Collection< AuditLogRecord >
logRecords, RevisionToken revisionToken,
57                     BusinessChangeSummary changeSummary, EDataObject
businessObject )
58     {
59         if ( doTrigger( changeSummary ) )
60         {
61             log.info( "trigger onChange( )" );
62         }

```

```

63     }
64
65     @Override
66     public void onBulkChange( AuditLogData logData, Collection<
67         AuditLogRecord > logRecord, RevisionToken revisionToken,
68         AuditChangeSummary changeSummary, Object[]
69         primaryKeys )
70     {
71         // ...
72     }
73
74     @Override
75     public void onBulkDelete( AuditLogData logData, Collection<
76         AuditLogRecord > logRecord, RevisionToken revisionToken,
77         AuditChangeSummary changeSummary, Object[]
78         primaryKeys )
79     {
80         // ...
81     }
82 }

```

HPM Provider

Since in previous Product 360 releases we already had a root entity audit log feature enabled for certain entities we needed to provide a default implementation which fulfills this known feature. The default implementation writes a "created, changed and deleted" value based on the physical events. Thus in case of any change, the default implementation will write the modification date/user and in case of delete it will write the deleted date/user.

This default implementation is contributed for all entity types and the classifier: "HPM".

Repository Definition

The types area of the repository has been enhanced to have a 1:n sub-entity for every root entity for all entity types. Logical key will be the mentioned classifier. The custom entity uses an enumeration of available classifiers and a default qualification for this key with "HPM". So the fields can be re-qualified by the user to show the audit log information for a different classifier. The used classifier enum provider uses the classifier elements from the `com.heiler.ppm.persistence.auditLog` extension point.

6.6.3 Audit Trail Extension Points

6.6.3.1 The extension point *ChangeSummaryTransformer*

For the history view it is possible to customize the visualization of the change summary tree. For a unique customer experience it may be required to have a special visualization for some entities or fields. One example is the attributes, where the values should be formatted depending on the attribute data type. The extension point `com.heiler.ppm.audittrail.server.changeSummaryTransformer` provides the possibility to customize the display of (custom) fields or (custom) entities.

In the screenshot below you can see the change summary tree for a changed short description. As mentioned above, this visualization could be changed by implementing a custom `ChangeSummaryTransformer`

Product "1595113807461016 - Television 4K HDMI"		
<div>Product 360° view</div> <div>Header</div> <div>Preview</div> <div>Text</div> <div>Prices</div> <div>Media</div> <div>Classification</div> <div>Attributes</div> <div>Variants of the product</div> <div>References</div> <div>Quality status</div> <div>Change information</div> <div>Translation</div> <div>All texts</div> <div>History</div>		
<div>Product no.: 1595113807461016</div> <div>Status: 01 New</div>		
<div>▼ Administrator</div> <div>Wednesday, 7/22/2020 11:26:07 AM (Moments ago) Initiated by User</div>		
Field	Old value	New value
▼ Header data		
▼ Language-specific data (English)		
Short description	Television	Television 4K HDMI

In the screenshot below you can find an example of an extension for the attribute value field.

All Extensions

Define extensions for this plug-in in the following section.

type filter text

com.heiler.ppm.audittrail.server.changeSummaryTransformer

Attribute value: value (transformer)
Attribute value: second value (transformer)
Characteristic records (ArticleType) (transformer)
Characteristic record lang (transformer)
Characteristic record value (transformer)

Add...
Remove
Up

Extension Element Details

Set the properties of 'transformer' Required fields are denoted by '*'.

shortIdentifier*: value
transformerFactory*: com.heiler.ppm.article.server.audittrail.transformer.AttributeV Browse...
label: Attribute value: value
parentEntity:
parentEntityType: ArticleAttributeValueType

Please note that all parameters are described in detail in the extension point itself. Please have a look there in case you need more details about a specific parameter.

Implementing a custom `ChangeSummaryTransformer`

If you want to implement a change summary transformer that transforms a json node from an `EntityItemChangeDocument` object into a `ChangeDocumentNode` object which is the model object for the visual representation in the UI, you need the transformer and a factory that is able to create instances of that transformer.

`ChangeSummaryTransformerFactory`

The factory has to implement the interface `ChangeSummaryTransformerFactory`, a constructor without any parameters is mandatory to be able to initialize the class. This factory has to be registered at the extension point `changeSummaryTransformer`.

The factory interface contains one method to create the transformer:

`ChangeSummaryTransformerFactory` interface method

```
ChangeSummaryTransformer getTransformer( String shortIdentifier, ChangeDocumentNode parentNode );
```


ChangeSummaryTransformer

The transformer implementation has to implement the interface `ChangeSummaryTransformer`. You may inherit from an OOB class `FieldTransformer` or `EntityTransformer` in case you have only small adjustments.

The interface contains the method

ChangeSummaryTransformer interface method

```
ChangeDocumentNode transform( JsonNode value, Locale locale, DocumentFilter
documentFilter ) throws CoreException;
```

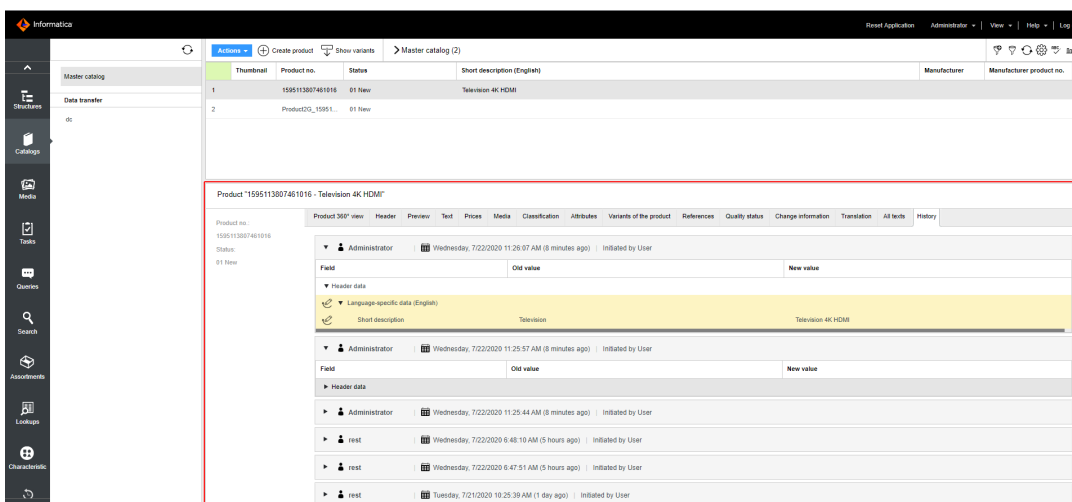
which mainly does the work. It returns a `ChangeDocumentNode` which is also an interface. You can use the OOB classes `FieldNode` or `EntityNode` and subclass them, or write your custom node.

Please find more detailed information in the java doc of the corresponding interfaces.

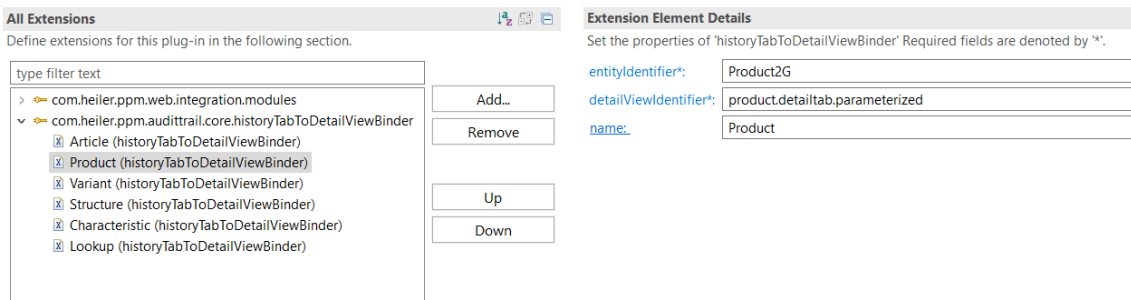
6.6.3.2 The extension point *historyTabToDetailViewBinder*

In case you want to display a history tab for a custom entity, you need to bind the history tab to the corresponding web detail view. This can be done via the extension point `com.heiler.ppm.audittrail.core.historyTabToDetailViewBinder`.

In the screenshot below you can see the history tab for the "Product" entity. The detail view for products is marked with a red border.



In order to make this work, we contributed the extension shown below in our standard code. If you want to show a history tab for your custom entity you have to do the same thing, of course with adjusted entity identifier as well as `detailViewIdentifier`.



In addition to the "Product" entity, also some other entities (Article, Variant, Structure group, Characteristic and Lookup value) are already contributed out of the box and **should not be overridden**. Please find more detailed information about the extension point as well as its parameters in the extension point definition.

6.6.4 Entity item change document

The entity item change document contains all changes which are persisted in a specific save operation including the old and new values. In case this save operation would only modify a single field, the change document only contains this field. In case a full item is created or modified with sub-entities and hundreds of fields, the document will have all that in it. The document is hierarchical like the repository entities. Its basic structure is derived from there. One exception to this rule are the characteristics which have an inherent hierarchical nature and thus will also be provided fully hierarchical in the document.

6.6.4.1 Datatypes

- ENTITY_ITEM objects always contain the `_entityId`, `_internalId` and `_externalId` attributes
 - `_internalId` = Service API syntax with internal numeric IDs - includes the container in case the entity item has one
 - `_externalId` = Service API syntax with the alphanumeric identifier of the entity item. Includes the container as well in case the entity item has one
 - `_entityId` = The numeric id of the repository entity
- MIME_VALUE objects always contain the relative `_filePath`, the `_label` and the `_mimeType`
- timestamp: ISO 8601, e.g.: 2012-04-23T18:25:43.511Z
- date: ISO 8601, e.g.: 2019-07-04
- numbers: standard JSON
- "no value" is either provided as null, or the attribute is not in the document.
- Empty string is returned as null, which is omitted when possible!
- Enumeration fields
 - `_key` : always holds the unique key of the enumeration entry
In case the enumeration field is of datatype ENTITY_ITEM, then the key will be provided with the ENTITY_ITEM syntax (see above)
 - `_code` : the external code of the enumeration entry, if the enum entry has one, and it differs from the key!

6.6.4.2 Meta attributes

Meta attributes are prefixed with an underscore.

<code>_module</code>	The area of the application which has triggered the modification: IMPORT, MERGE, CLONE, SERVICEAPI, REVISION, ENVIRONMENT_TRANSFER, UI, OTHER.
<code>_user</code>	The entity item of the user which executed the CRUD operation
<code>_eventTimestamp</code>	The timestamp at which this document is created
<code>_revision</code>	The entity item of the revision in which the operation has been performed
<code>_changeType</code>	<p>The change type of the entity record. The change type of the parent record applies if no change type is given.</p> <ul style="list-style-type: none"> • CREATED = the entity record has been created (this implies that all children of this also have been created) • CHANGED = the entity record has been changed. Also it's children MIGHT have been changed, but at least the entity itself has changed • CHANGED_CHILD = only one or multiple children of this entity record have changed, not the record itself • DELETED = the entity record has been marked as deleted or has been physically deleted.
<code>_qualification</code>	The qualification of the entity. The combination of the fields in the qualification object define this record as unique within it's parent. Qualification values do not have current and old values as a change in the qualification implies a create/delete of the record even if that is not the case in our database.
<code>_entity</code>	The root entity which is changed
<code>_identifier</code>	The identifier of the entity item

<code>_container</code>	The container of the root entity record
<code>_entityItem</code>	<p>The root entity record's proxy information</p> <ul style="list-style-type: none"> In case the identifier of an item is changed, the externalId of the item also changes, starting with the next change document. So when you search for it, you will at least find the document with the old identifier and see that the last change for this is a change of the identifier. This is identical for the <code>_identifier</code> attribute of course.
<code>_changedFields</code>	<p>An array of fields which got changed (fields are without qualifications)</p> <ul style="list-style-type: none"> We use the field identifier, not the short identifier for this element. This way sub-entity fields are unique as well In case characteristics are affected, the <code>ArticleCharacteristicValueLang.Values</code> field is included
<code>_changedEntities</code>	An array of entity identifiers which are part of the change summary (but only in case the entity is created, changed or deleted - not for <code>changed_child!</code>)
<code>_transactionStatus</code>	<p>State of the current audit trail transaction. Every record has one of the following status</p> <ul style="list-style-type: none"> COMPLETE = the whole transaction was successfully completed INCOMPLETE = transaction has still not completed yet. This can happen as we store the document while parallelly persisting record in the database. This signifies <i>transaction under processing</i> stage. If these documents stay in this state for too long, then they would be picked up by the watchdog. COMPLETE_BY_WATCHDOG = this status signifies that an incomplete document that was picked up by the watchdog has been processed and marked as valid transaction. INVALID = this status signifies that the transaction for which this document was created didn't really process successfully. POTENTIALLY_INVALID = this status signifies that it can't be guaranteed that the transaction did happen successfully. EXCEPTION = an exception while validating that the transaction was valid or not.
<code>_invalidReason</code>	Defines the reason why the document was marked as invalid or had an exception

<code>_relationshipType</code>	<p><i>Internal attribute</i></p> <p>This defines what type of document it is. It can be of either of these 2 types</p> <ul style="list-style-type: none"> • <code>changeSummaryDoc</code> = contains the change summary specific document • <code>triggerFiredDoc</code> = contains the trigger related information
--------------------------------	---

6.6.4.3 Change Summary

All data elements are located within the `_changeSummary` attribute. The `_current` and `_old` attributes for fields provide the current and old value, if one of them is omitted, it means there was/is no corresponding value. Qualifications do not have current and old values, in case a qualification field is modified, the whole record is twice in the document, one time with the old value marked as deleted, one time with the new one, marked as created.

In case an entity item has been deleted, the change document does not contain a change summary!

Field or entity names

A new repository property has been added in the custom section of the repository. The new "shortIdentifier" can be generated by a function in the repository editor in case no value is set for it. However, it should never be changed for standard fields. There might be specially contributed logic which is linked to the short identifier. On the other hand, when enabling a reserved field in the custom section, the short identifier must be defined. It should reflect the logical content of the field, and not it's field type. So, instead of having "res_Text_200" as short identifier, chose something more meaningful.

The short identifier must be unique within it's parent entity. This short identifier is available for entities, fields and logical keys. In case of logical keys which are bound to a field, it should be identical.

Please see the [Domain Model \(Repository\)](#) (see page 79) documentation on details on allowed characters and syntax.

What is not part of the change summary

- Object permissions. If someone modifies the object permission of an entity item, this is not recorded in the audit log
- Password fields: fields which are marked as password fields are not recorded in the audit trail
- Fields/Entities without the " `shortIdentifier` " attribute in the repository
- Qualifications which have a fixed value and are not modifiable by the user - so called "disabled" logical keys
- Fields/Entities which has `supportsAuditTrail` set to false
- Root Entities which do not have an `AuditTrailSettings` child element, or in which the retention policy has been set to "NO_RETENTION"

Examples

CREATE	CHANGED
<p>Item is created with a name and description in English and French</p> <pre> { "_module": "IMPORT", "_user": { "_entityId": 2600, "_internalId": "47", "_externalId": "\u0027abuehler\u0027" }, "_eventTimestamp": "2020-05-28T23:28:56.782Z", "_revision": { "_entityId": 5600, "_internalId": "1", "_externalId": "\u0027HEAD\u0027" }, "_entity": "Item", "_identifier": "MyItem", "_container": { "_entityId": 2900, "_internalId": "1", "_externalId": "\u0027MASTER\u0027" }, "_entityItem": { "_entityId": 1000, "_internalId": "345@1", "_externalId": "\u0027MyItem\u0027@\u0027MASTER\u0027" }, "_changeType": "CREATED", "_changedEntities": ["Article", "ArticleLang"], "_changedFields": ["Article.GTIN", "ArticleLang.Name", "ArticleLang.Description"], "_changeSummary": { "item": { "gtin": { </pre>	<p>Item is changed. New value for German, modified value for English and the French value is removed</p> <pre> { "_module": "UI", "_user": { "_entityId": 2600, "_internalId": "47", "_externalId": "\u0027abuehler\u0027" }, "_eventTimestamp": "2020-05-28T23:28:56.782Z", "_revision": { "_entityId": 5600, "_internalId": "1", "_externalId": "\u0027HEAD\u0027" }, "_entity": "Item", "_identifier": "MyItem", "_container": { "_entityId": 2900, "_internalId": "1", "_externalId": "\u0027MASTER\u0027" }, "_entityItem": { "_entityId": 1000, "_internalId": "345@1", "_externalId": "\u0027MyItem\u0027@\u0027MASTER\u0027" }, "_changeType": "CHANGED", "_changedEntities": ["Article", "ArticleLang"], "_changedFields": ["ArticleLang.Name", "ArticleLang.Description"], "_changeSummary": { "item": { "gtin": { </pre>

CREATE	CHANGED
<pre> "_current": "11112222333" }, "lang": [{ "_qualification": { "language": { "_key": 9, "_code": "eng" } }, "name": { "_current": "spicy cookie" }, "description": { "_current": "yummy cookie" } }, { "_qualification": { "language": { "_key": 12, "_code": "fra" } }, "name": { "_current": "somethingInFrench" }, "description": { "_current": "somethingInFrench" } }] }, "_transactionStatus": "COMPLETE", "_relationshipType": { "name": "changeSummaryDoc" } } } </pre>	<pre> "_current": "4711239283", "_old": "11112222333" }, "lang": [{ "_changeType": "CREATED", "_qualification": { "language": { "_key": 7, "_code": "deu" } }, "name": { "_current": "Spekulazius" }, "description": { "_current": "Lecker Kekse!" } }, { "_changeType": "CHANGED", "_qualification": { "language": { "_key": 9, "_code": "eng" } }, "name": { "_current": "spiced cookie", "_old": "spicy cookie" }, "description": { "_old": "yummy cookie" } }], { "_changeType": "DELETED", "_qualification": { "language": { "_key": 12, "_code": "fra" } }, "name": { "_old": "somethingInFrench" } }, </pre>

CREATE	CHANGED
	<pre> "description": { "_old": "somethingInFrench" }] } }, "_transactionStatus": "COMPLETE", "_relationshipType": { "name": "changeSummaryDoc" } }</pre>

DELETED

Item is deleted in the system.

```
{
  "_module": "UI",
  "_user": {
    "_entityId": 2600,
    "_internalId": "47",
    "_externalId": "\u0027abuehler\u0027"
  },
  "_eventTimestamp": "2020-05-28T23:28:56.782Z",
  "_revision": {
    "_entityId": 5600,
    "_internalId": "1",
    "_externalId": "\u0027HEAD\u0027"
  },
  "_entity": "Item",
  "_identifier": "MyItem",
  "_container": {
    "_entityId": 2900,
    "_internalId": "1",
    "_externalId": "\u0027MASTER\u0027"
  },
  "_entityItem": {
    "_entityId": 1000,
    "_internalId": "345@1",
    "_externalId": "\u0027MyItem\u0027@\u0027MASTER\u0027"
  },
  "_changeType": "DELETED",
  "_transactionStatus": "COMPLETE",
  "_relationshipType": {
    "name": "changeSummaryDoc"
  }
}
```

Characteristic Values

Characteristic values are rendered in a specialized structure in order to honor their hierarchical nature. This structure is not identical to the repository as it is fully hierarchical. This is the reason for extra meta attributes which help to not get in conflict with repository based data.

Additional Meta Attributes

<code>_characteristicRecords</code>	Top level element for all characteristic records
<code>_recordLanguage</code>	Language specific record values. In case the characteristic is not language specific, the qualification for "not language specific" is returned (-1 or xxz)
<code>_children</code>	Children of the current characteristic record
<code>_datatype</code>	The characteristic data type - to be able to interpret the values even in case the characteristic is no longer in the system
<code>_formatPattern</code>	The format pattern of the characteristic - to be able to format the values even in case the characteristic is no longer in the system

Examples

```
{
  "_module": "IMPORT",
  "_user": {
    "_entityId": 2600,
    "_internalId": "47",
    "_externalId": "\u0027abuehler\u0027"
  },
  "_eventTimestamp": "2020-05-28T23:28:56.782Z",
  "_revision": {
    "_entityId": 5600,
    "_internalId": "1",
    "_externalId": "\u0027HEAD\u0027"
  },
  "_entity": "Item",
  "_identifier": "MyItem",
  "_container": {
    "_entityId": 2900,
    "_internalId": "1",
    "_externalId": "\u0027MASTER\u0027"
  },
}
```

```

"_entityItem": {
  "_entityId": 1000,
  "_internalId": "345@1",
  "_externalId": "\u0027MyItem\u0027@\u0027MASTER\u0027"
},
"_changeType": "CREATED",
"_changedEntities": [
  "Article",
  "ArticleLang",
  "ArticleCharacteristicValue",
  "ArticleCharacteristicValueLang"
],
"_changedFields": [
  "Article.GTIN",
  "ArticleLang.Name",
  "ArticleLang.Description",
  "ArticleCharacteristicValueLang.Values"
],
"_changeSummary": {
  "item": {
    "gtin": {
      "_current": "11112222333"
    },
    "lang": [
      {
        "_qualification": {
          "language": {
            "_key": 9,
            "_code": "eng"
          }
        },
        "name": {
          "_current": "spicy cookie"
        },
        "description": {
          "_current": "yummy cookie"
        }
      }
    ]
  },
  "_characteristicRecords": [
    {
      "_changeType": "CREATED",
      "_qualification": {
        "characteristic": {
          "_key": {
            "_entityId": 8000,
            "_internalId": "7",
            "_externalId": "\u0027AnimalIngredient\u0027"
          },
          "_code": "AnimalIngredient"
        },
        "recordKey": "0000.0000.RK",

```

```

    "parentRecordKey": "root"
  },
  "_dataType": "LOOKUP",
  "order": {
    "_current": -32767
  },
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": -1,
          "_code": "zxx"
        }
      },
      "values": {
        "_current": [
          {
            "_entityId": 7300,
            "_internalId": "23@5",
            "_externalId":
"\u0027Down\u0027@\u0027AnimalIngredient\u0027"
          }
        ]
      }
    }
  ],
  "_children": [
    {
      "_qualification": {
        "characteristic": {
          "_key": {
            "_entityId": 8000,
            "_internalId": "8",
            "_externalId": "\u0027CertDown\u0027"
          },
          "_code": "CertDown"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "0000.0000.RK"
      },
      "_dataType": "LOOKUP",
      "order": {
        "_current": -32766
      },
      "_recordLang": [
        {
          "_qualification": {
            "language": {
              "_key": -1,
              "_code": "zxx"
            }
          },
          "values": {
            "_current": [
              {
                "_entityId": 7300,
                "_internalId": "23@5",
                "_externalId":
"\u0027Down\u0027@\u0027AnimalIngredient\u0027"
              }
            ]
          }
        }
      ]
    }
  ]
}

```

```

    "values": {
      "_current": [
        {
          "_entityId": 7300,
          "_internalId": "25@6",
          "_externalId": "\u0027Other\u0027@\u0027CertDown\u0027"
        }
      ]
    }
  },
  "_children": [
    {
      "_qualification": {
        "characteristic": {
          "_key": {
            "_entityId": 8000,
            "_internalId": "11",
            "_externalId": "\u0027CertDownExpDate\u0027"
          },
          "_code": "CertDownExpDate"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "0000.0000.RK"
      },
      "_dataType": "DATE",
      "order": {
        "_current": -32765
      },
      "_recordLang": [
        {
          "_qualification": {
            "language": {
              "_key": -1,
              "_code": "zxx"
            }
          },
          "values": {
            "_current": [
              "1977-05-28"
            ]
          }
        }
      ]
    }
  ],
  {
    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,

```

```

        "_internalId": "8",
        "_externalId": "\u0027CertDown\u0027"
    },
    "_code": "CertDown"
},
"recordKey": "0000.0000.RK",
"parentRecordKey": "0000.0000.RK"
},
"_dataType": "LOOKUP",
"order": {
    "_current": -32766
},
"_recordLang": [
    {
        "_qualification": {
            "language": {
                "_key": -1,
                "_code": "zxx"
            }
        },
        "values": {
            "_current": [
                {
                    "_entityId": 7300,
                    "_internalId": "25@6",
                    "_externalId": "\u00270ther\u0027@\u0027CertDown\u0027"
                }
            ]
        }
    }
],
"_children": [
    {
        "_qualification": {
            "characteristic": {
                "_key": {
                    "_entityId": 8000,
                    "_internalId": "11",
                    "_externalId": "\u0027CertDownExpDate\u0027"
                },
                "_code": "CertDownExpDate"
            },
            "recordKey": "0000.0000.RK",
            "parentRecordKey": "0000.0000.RK"
        },
        "_dataType": "DATE",
        "order": {
            "_current": -32765
        },
        "_recordLang": [
            {
                "_qualification": {

```

```

        "language": {
            "_key": -1,
            "_code": "zxx"
        }
    },
    "values": {
        "_current": [
            "1977-05-28"
        ]
    }
}
]
}
],
"_children": [
{
    "_changeType": "CREATED",
    "_qualification": {
        "characteristic": {
            "_key": {
                "_entityId": 8000,
                "_internalId": "8",
                "_externalId": "\u0027CertDown\u0027"
            },
            "_code": "CertDown"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "0000.0000.RK"
    },
    "_dataType": "LOOKUP",
    "order": {
        "_current": -32766
    },
    "_recordLang": [
        {
            "_qualification": {
                "language": {
                    "_key": -1,
                    "_code": "zxx"
                }
            },
            "values": {
                "_current": [
                    {
                        "_entityId": 7300,
                        "_internalId": "25@6",
                        "_externalId": "\u00270ther\u0027@\u0027CertDown\u0027"
                    }
                ]
            }
        }
    ]
}
]
}

```

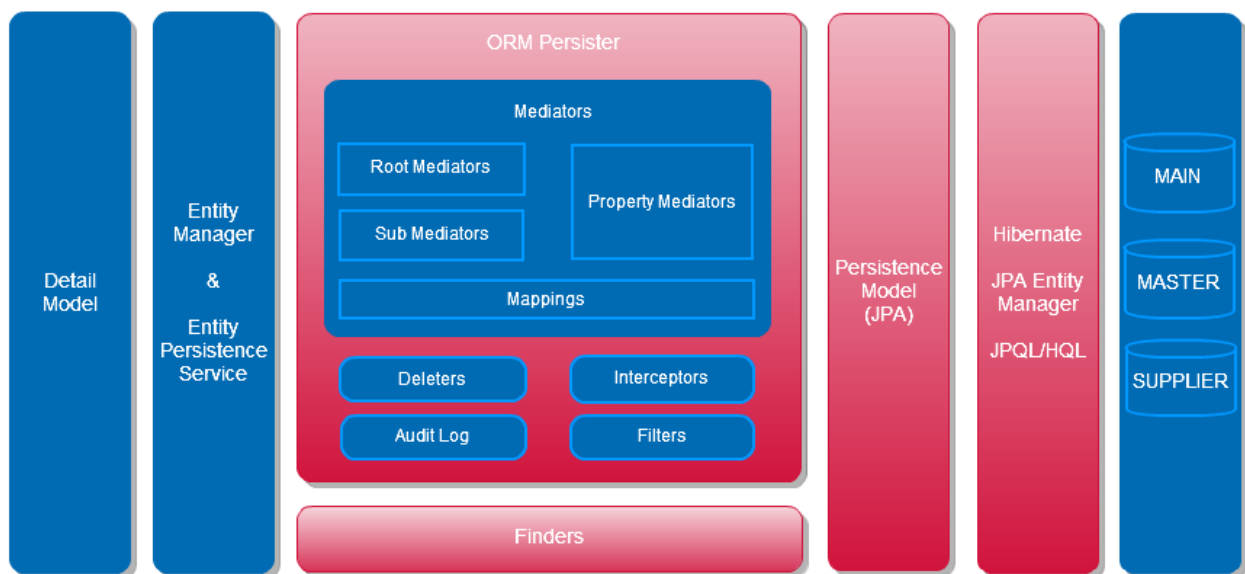

- [Persistence mapping in repository](#) (see page 207)
 - [Entity Type attributes](#) (see page 207)
 - [Field Type attributes](#) (see page 208)
- [Persistence meta model](#) (see page 209)

The persistence layer has different extension points which can be used to adjust the default persistence logic. Developers usually need to create extensions if the business model has been modified or extended in a non standard way. Before contributing extensions in the persistence layer, please consider other possibilities like the [command framework](#) (see page 163). Keep in mind that persistence layer extensions should only used to adjust persistence logic but not to implement business logic!

Most of the persistence layer implementation logic uses meta information to perform data manipulation operations (deletion, search, update, hql queries generations etc). This meta information is composed from persistence fields defined in the repository (called mappings) and JPA persistence entities called persistence meta model. JPA entities itself are called persistence model.

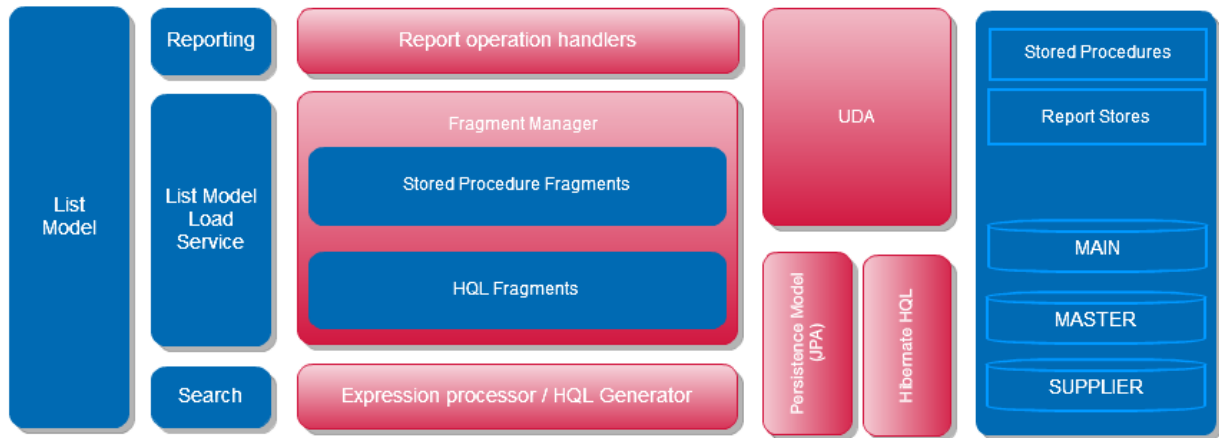
6.7.1 Detail persistence

If you need to adjust the detail model persistence logic, the property mediators extension point is a good place to start. If you need to contribute custom data modification audit logic, please use the AuditLog extension point. Using deleters, finders and interceptors you can adjust deeper aspects of the persistence process. In rare cases you can implement custom mediators. If you create new entity types you will also need to create new persistence model entities.



6.7.2 List persistence

If you need to adjust list access please have a look at the Fragments extension point. If you create new EntityTypes you need to set column data in the repository and contribute respective fragment (in most cases generic HQL fragment implementation will do the work). To define new reports you will need to create PL/SQL or T-SQL stored procedures, or you can simply use search expressions.



6.7.3 Business and persistence models

HPM has a well defined separation between the persistence and business aspects of the domain model. Basically both models contain the same data but in the different forms and with different access methods. Business model provides generic access methods (for example using XPath-like queries), has meta information which is used for presentation, validation, business rules definition and a lot more. Business model also provides DTO capabilities like transferring disconnected data model sub-graphs over network and change logging. On the other hand persistence model is simple a set of JavaBeans enriched with JPA annotations. Persistence model is simple used to store/load data from DB using Hibernate and JPA. This approach provides a flexible and extendable model on business side and easy to implement and debug persistence technology on the other side. Data transfer between models is the task of the mediator framework. Mediators use mappings between business and persistence models which are defined in the repository.

Since the persistence model is also normalized, one could think it is exactly the same as the business model - in fact, it's not.

For example

- the business model stores instances of `EntityProxy`, where the persistence model only stores the numeric id of the entity proxy.
- customizing can introduce a 1:n field (a list) where as the persistence model just has a big string field (values will be stored by some sort of string separation).
- parts of the business model might just be calculated from other fields or 3rd party systems
- in business model every entity has version property, where in persistence model entity and entity revision are different objects

The central approach of the Heiler Product Manager - being easily customizable - is bound to the fact that we have a separate business model.

6.7.4 Mapping

As mentioned, the persistence model is quite similar to the business model. Because of this we're able to provide a lot of generic functionality which mediate between the business and the persistence model automatically - always being able to intercept and customize this of course.

For this we established a mapping model which holds the mapping information between business model and persistence model. You can obtain this model from the

```
PersistenceComponent ( PersistenceComponent.getMappingModelRegistry())
```

6.7.4.1 Persistence mapping in repository

The repository defines several attributes which map the business model to the persistence model. Please see the Persister Framework documentation for details on the used mapping algorithms, the following list gives only a short summary of the repository attributes.

Entity Type attributes

Persistence Class Name	All entity types	Fully qualified path to the persistence model's class which directly corresponds to the business entity type on which the attribute is defined
Persistence XPath	All entity types	Path to the persistence object starting from the root. Path segments are separated by a slash '/' and must also start with a slash and are build using the object names of the corresponding members. E.G / articleRevision/articleLangs (defines the articleRevision set in the Article object (root), followed by the articleLangs set.
Identifying Field Type	All entity types	The field type which uniquely identifies the record in the business as well as in the persistence model

Deletion Mode	All entity types	<p>Defines what deletion modes the entity supports.</p> <ul style="list-style-type: none"> • HARD = only hard (aka physical) deletion possible • SOFT = only soft (aka mark as deleted) deletion possible Note, in this case the persistence class must inherit from the <code>Deletable</code> class! • BOTH = both supported.
Deletion Date Field Type	All entity types	<p>Field type which reflects the deletion date property of the entity type. Empty in case the entity type doesn't support soft delete at all. Used for the generic deletion algorithms.</p>
Revision Persistence Class	Only root entity types	<p>Fully qualified class name of the persistence class which holds the revision information for this entity type. This attribute is empty in case the root entity type doesn't support revisions</p>
Revision Property	Only root entity types	<p>Name of the property which gives access to the set of revision objects (which based on the <code>Revision Persistence Class</code>)</p>
Audit Log Entity Type	Only root entity types	<p>The entity type which implements the audit log functionality, empty in case the root entity type doesn't support the audit log feature</p>

Field Type attributes

Fragment Column Access	<p>Only used for SP based fragment access. Defines the table name and the column name of the corresponding database column, separated by a dot '.'. Both names are case sensitive!</p>
-------------------------------	--

Persistence class name	The java data type of the physical column in the database. Values will automatically be converted to and from the class name (which reflects the java data type in the business model). In case you have your own classes here, you need to provide a converter through the <code>ConverterUtils</code> in the <code>com.heiler.ppm.commons</code> bundle.
Persistence model class	Fully qualified path to the persistence model's class which directly corresponds to the business attribute. If empty, this defaults to the persistence class name of the field type's entity type.
Persistence xPath	Path to the persistence property starting from the root persistence object (In case of Collections in the persistence model, only the first value will be used!)
Physical Column Is Big	true in case the database column is some LOB object or similar (like text, ntext, nvarchar(max), clob, nclob, etc.)

6.7.4.2 Persistence meta model

The meta model of the JPA provider was not able to satisfy all requirements which we had in order to provide the generic mapping algorithms. Therefore we created an own meta model which combines information from the JPA provider as well as the persistence classes and the annotations them self. Using this persistence model we could create generic JPQL generators for deletion of items as well as selecting data for list model access. The persistence meta model can be obtained through the

`DataSourceRegistry` which itself is available through the `PersistenceComponent`.

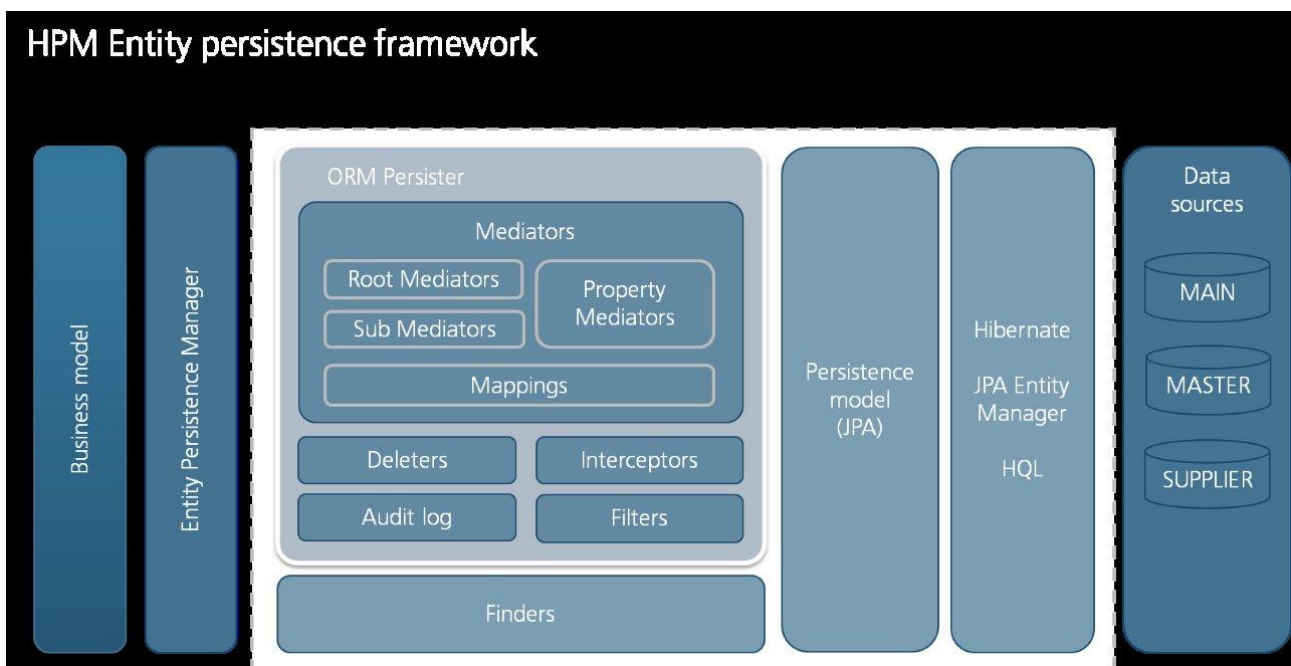
6.7.5 Detail Model Persistence

- [Persister](#) (see page 210)
 - [Persister Interceptors](#) (see page 211)
- [Mediators](#) (see page 211)
 - [Mediator Session](#) (see page 212)
 - [Root Mediator](#) (see page 212)
 - [Sub Mediator](#) (see page 212)
 - [Property Mediator](#) (see page 212)
 - [Persistence Object Deleter](#) (see page 212)
 - [Persistence Object Finder](#) (see page 213)

- [Mediator Interceptors](#) (see page 213)
- [Audit Log](#) (see page 213)
 - [Physical audit log](#) (see page 214)
 - [Logical audit log](#) (see page 214)
 - [AuditLogWriter](#) (see page 214)
 - [AuditLogProvider](#) (see page 215)
 - [HPM Provider](#) (see page 215)
 - [Repository Definition](#) (see page 215)

The persister framework is responsible for the physical interaction with the database storage. It loads, saves and deletes business models by mediating them to the corresponding [Persistence Model](#) (see page 216) and utilizing the Java Persistence API (JPA) via Hibernate to manipulate the database by corresponding DML statements.

Usually application developers should never use the persister framework directly, it is used by the [Entity Manager](#) (see page 209) of the [Business](#) (see page 209) automatically, but it's always good to know how things work in case you need to do some troubleshooting. This section gives an overview of the persister framework and the used algorithms as well as extension possibilities. Please refer to the linked javadoc for details on the available methods and classes.



6.7.5.1 Persister

The persister instance for a specific business model can be obtained from the `PersisterRegistry` which itself must be obtained from the `PersistenceComponent`. Currently the only available implementation of the persister interface is the `ORMPersister` which uses the object relational mapping functionality of JPA via Hibernate. Please note that the `ORMPersister` implementation itself is internal and must not be subclassed or directly used in any kind. It is also not (yet) possible to contribute an own persister, although the platform has been prepared for this.

Persister Interceptors

Currently the HPM platform provides two interceptors which are called (and must be called!) directly within the persister implementation.

-

`SavePreCommitInterceptor` is being called after the persistence model has been flushed to the database, but before the transaction has been committed! Therefore, any exception thrown in this interceptor will lead to an immediate rollback of the transaction!

-

`SavePostCommitInterceptor` is being called after the persistence model has been flushed to the database, and the transaction has been committed. You can perform tasks here which must be executed after the transaction has been committed - note that any exception thrown in this interceptor will not affect the written data, a rollback is no longer possible!

6.7.5.2 Mediators

The core task of a mediator is the mediation between the [Persistence Model \(see page 216\)](#) and the [Business Model \(see page 133\)](#). Mediators should not execute database operations by themselves, they should only work with the models provided. It's the persisters task to call the corresponding JPA flush methods etc. Mediators can be contributed for a specific entity or field type (depends on the mediator type) using the

`com.heiler.ppm.persistence.server.persister` extension point. The HPM platform provides generic implementations (`GenRootMediator`, `GenSubMediator`,

`PropertyMediatorImpl`) which will always be used as long as no specialized mediator has been contributed.



Mediators can not be exchanged by contribution!

Mediator Session

It is the persister's responsibility to create and initialize the `MediatorSession` which will be used in every mediator. The `MediatorSession` provides access to the root entity type, data source, mapping model, log data (user and timestamp) the JPA entity manager as well as the revision which should be used. Additionally the `MediatorSession` is the only way of communicating between different mediator / interceptor instances by it's get/set data interface. All objects in the persister framework are cached singletons and must not hold any state other than their registries and caches.

Root Mediator

The `RootMediator` is responsible for root entity types. The root mediator loads and writes the root data and then calls the property and sub mediators for the immediate child entity types and field types.

Sub Mediator

The `SubMediator` is analyzing the `BusinessChangeSummary`. Based on this it's determining the corresponding persistence objects and calls the property mediators for each of them. Last but not least it's calling the `SubMediator` recursively for each of it's own child entity types.

Property Mediator

The `PropertyMediator` reads the field's persistence value from the persistence object, converts it to the business datatype using the `ConvertUtils` and applies it to the business object. On the other hand it reads the business value, converts it to the persistence data type and applies it to the [Persistence Model](#) (see page 216). Summarized you could say it reads and writes a single property (or field type).

Persistence Object Deleter

`PersistenceObjectDeleter` implements the actual mass deletion logic for an entity type. In contrary to the Mediators, the `PersistenceObjectDeleter` is executing bulk update or delete JPQL statements to do it's work. The generic implementation is building a dynamic JPQL statement, so in most cases it's not necessary (or even a good idea) to interfere with this.

However, you can still contribute an own `PersistenceObjectDeleter` using the `com.heiler.ppm.persistence.server.persister` extension point.

Persistence Object Finder

`PersistenceObjectFinder` is responsible to find the persistence objects based on a given business object. It's utilized in the mediators and must sometimes be implemented in a special way for some more complex business model aspects. Just like the `PersistenceObjectDeleters` you should not interfere with those implementations and mostly the generic implementation is just fine.

Mediator Interceptors

The mediators do have a lot of interceptors which can be used to customize their behavior or to add additional, needed persistence logic. Please note that you should not try to implement business logic using the persistence framework at all - business logic should always be implemented using the [Command Framework](#) (see page 163) if possible.

-

`PostReadInterceptor` is being called after the business object for whose entity type the interceptor has been contributed has been read from the [Persistence Model](#) (see page 216) (thus standard read operations have been performed and the business model has been filled)

- `PostReadPropertyInterceptor` (see page 209) is being called after the business property for which the interceptor has been contributed has been read. This interceptor must be contributed for a single field type.
- `PostWriteInterceptor` (see page 209) is being called after the business objects changes have been applied to the persistence object. This interceptor must be contributed for an entity type.
- `PostWritePropertyInterceptor` (see page 209) is being called after the business property has been applied to the persistence object. This interceptor must be contributed for a single field type.
- `PreFilterInterceptor` (see page 209) is being called before the hibernate persistence filters are being activated and parameterized. Contributors can provide own logic to manipulate this.

6.7.5.3 Audit Log

Audit logging is an important feature in an application which has a lot of users and data transaction every day. Providing delta reports for 3rd party systems like web-shops was one of the main functional focuses on the audit log implemented in HPM 5.3. Therefore we enhanced the complete [Persistence Model](#) (see page 216)

and provided also a new extension point specifically designed to be implemented by consulting in order to be prepared for the future and be most flexible in this area.



An audit log must not be mixed up with an audit trail! The audit log functionality stores creation, modification and deletion of objects. But only the time and the user will be stored. Every record/object thus has only one record with these data. In contrary an audit trail functionality stores also the value which has been changed in a historical way. So you can see then not only who and when this record has been changed the last time, but also what has been changed (which column), what was the value before etc. Audit trail is way more complex then audit log and has not yet been implemented in the HPM standard.

Physical audit log

Nearly all persistence objects in HPM support the `Auditable` (see page 209) base class (starting with HPM 5.3). Therefore all the corresponding database tables have been enhanced accordingly (please see [Persistence Model](#) (see page 216) for details). The persister framework checks every persistence object which is going to be saved in the database and updates the creation/modification timestamps automatically. Clients must do just nothing for this feature except supporting the needed columns in their persistence tables.

Logical audit log

The requirements for logical audit logging of root entities is being driven by use cases like "Show all items which have been created by user 'abc' since yesterday", or "Show all items which have been modified since last week". To be able to fulfill these requirements we need to store audit log data not only in every table (like with the physical audit log), but also on the business root object. Unless that, we would need to join nearly all tables for a specific entity item so we can determine if the item has been changed. (Maybe only a price has changed, which would not be reflected in a change of the Article (or ArticleRevision) table. Additionally to that, the definition of "changed, created and deleted" may differ depending on specific scenarios which use this information. For example, regarding an ERP replication process only certain fields may be relevant for replication - therefore as seen from the ERP system, an object has been changed when one of those fields change only.

With HPM 5.3 we returned from the approach of storing this data in a generic table, since in fact, HPM has no dynamic [Persistence Model](#) (see page 216) - we only have a dynamic Business Model. Introducing a generic model which stores data for multiple different entity types was confusing and error prone, thus we refactored this part in 5.3 again.

AuditLogWriter

The persister calls the `AuditLogWriter` (see page 209) after the persistence object has been modified by the mediators, but before flush is being called on the `JPA EntityManager`. With this we ensure two things:

- Limit the number of database round trips and provide hibernate with the possibility to streamline the SQL statements it sends to the database.
- Make sure that the audit log records are written in a synchronous and transaction safe manner. Thus we make sure that the audit log records are not written in case the entity item can not be saved due any reason.



It's currently not possible to customize the `AuditLogWriter` itself, please use the `AuditLogProvider` for this

AuditLogProvider

Since direct interaction with the persistence layer is not a recommended customizing, we provide an extra interface which is specifically designed to be customized. The `AuditLogProvider` (see page 209) interface provides methods to implement special custom logic. The `AuditLogWriter` will read the current audit log record and calls then the `AuditLogProvider` which itself is responsible to set the creation/modification/deletion user and timestamp in the given `AuditLogRecord` (see page 209) object. The `AuditLogProvider` implementation however **must not** save this record to the database in any case! It's the `AuditLogWriter` which is responsible to build the bridge between `Persister` and `AuditLogProvider` and will save the `AuditLogRecord` in the most effective way. Please note that the `AuditLogProvider` interface also provides methods for bulk modifications of entity items. These methods will be called in case of bulk updates which occur in deletion scenarios mostly.

An entity item can have multiple logical audit log records which are distinguished by the classifier field. So you could have an audit log record specific for different export channels like ERP system or Web-Shop etc.

`AuditLogProviders` can be contributed with the `com.heiler.ppm.persistence.auditLog` extension point. If you want to have multiple audit log records, you will need to contribute also an appropriate classifier in this extension point.

HPM Provider

Since in previous HPM releases we already had a root entity audit log feature enabled for certain entities we needed to provide a default implementation which fulfills this known feature. The default implementation writes a "created, changed and deleted" value based on the physical events. Thus in case of any change, the default implementation will write the modification date/user and in case of delete it will write the deleted date/user.

This default implementation is contributed for all entity types and the classifier: "HPM".

Repository Definition

The types area of the repository has been enhanced to have a 1:n sub-entity for every root entity for all entity types. Logical key will be the mentioned classifier. The custom entity uses an enumeration of available classifiers and a default qualification for this key with "HPM". So the fields can be requalified by the user to show the audit log information for a different classifier. The used classifier enum provider uses the classifier elements from the `com.heiler.ppm.persistence.auditLog` extension point.

6.7.6 List Model Persistence

Fragment

6.7.7 Persistence Model

- [Data sources](#) (see page 216)
 - [Extension point](#) (see page 216)
 - [Data source registry](#) (see page 217)
 - [Connection Pool](#) (see page 217)
- [Persistence model](#) (see page 217)
 - [Bundles](#) (see page 218)
 - [Dependencies](#) (see page 218)
 - [Model classes](#) (see page 218)
 - [JPA limitations](#) (see page 218)
 - [Available interfaces/base classes which must or can be used](#) (see page 218)
 - [Persistable](#) (see page 218)
 - [Deletable](#) (see page 219)
 - [Auditable](#) (see page 219)
 - [AuditLogSupport](#) (see page 220)
 - [Example](#) (see page 223)
 - [Extension point](#) (see page 225)
- [Named queries](#) (see page 226)
 - [Conventions](#) (see page 226)
 - [Example](#) (see page 226)
 - [Extension point](#) (see page 227)
- [3rd party libraries](#) (see page 227)

6.7.7.1 Data sources

Since HPM has multiple data sources we provide a corresponding extension point for the application layer to introduce the data sources and to provide needed configuration settings for the JPA persistence-units. Additionally to that the persistence layer gives access to the needed JPA objects so the application layer can also use raw JPQL statements if they absolutely must.

Extension point

The application layer contributes the datasources to the `com.heiler.ppm.persistence.db.datasource` extension point. The extension-point provides elements for the data source (identifier, alias) as well as the jpa persistence unit. The standard HPM application contributes three data sources `MAIN`, `MASTER` and `SUPPLIER`. For every supported database system there must be a separate persistence unit contribution which provides the configuration file for this unit. A persistence unit is a combination of the `DataSource` and the database management system with the corresponding link to the configuration file. All persistence unit configuration files are located in the configuration area's sub-directory called: `database`. Configuration files do have a naming schema like this:

```
[HPM:dataSourceIdentifier].properties.template.[HPM:dbmsIdentifier] (e.g.
main.properties.template.MSSQL2005)
```

Data source registry

The data source registry which can be obtained with

`PersistenceLayer.getContext().getDataSourceRegistry()` provides methods to acquire a JPA `EntityManager`. Please take a look at the corresponding javadoc for [DataSourceRegistry](#) (see page 216) for details on the available methods.

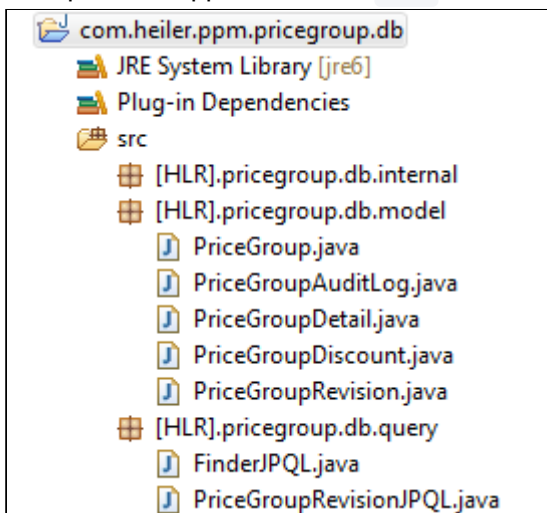
Connection Pool

HPM uses the Tomcat connection pool which can be configured using the persistence-unit configuration files located in the `configurationArea/database`. Please see the connection pool documentation for details.

6.7.7.2 Persistence model

The Heiler Product Manager persistence model is build using standard java persistence technologies (JPA). The HPM platform provides base classes and interfaces which can (or sometimes must) be implemented by the application layer in order to function properly in the HPM context.

Example of an application level `.db` bundle:



The persistence model must follow certain naming conventions as well as implementation rules which assure that the generic HPM functions can use them properly. These conventions are described in further sections. Additionally to that it makes life easier for everyone when similar things are implemented the same way.

Bundles

The persistence layer is strictly separated from other parts of the application and therefore we use own bundles for this layer. All persistence classes as well as JPQL named queries must be located in the plugins which end with `.db`. For example: `com.heiler.ppm.article.db` or `com.heiler.ppm.catalog.db`



This separation has to be maintained since we might need the persistence layer in the database setup again - without the whole HPM application.

Dependencies

The persistence model must not have any dependency to other than `.db` bundles or to third party libraries which are not already reexported by the platform `.db` bundles. It must be guaranteed that the set of all `.db` bundles can work on their own without having any dependency to `.core`, `.server` or `.ui` bundles, either directly or indirectly!

Model classes

Model classes (aka JPA Entities) must be located in the `.model` package and named exactly like the corresponding database table. Note that they must conform to the Java Bean standard, therefore the members **must not** be named like the database columns regarding the case of the first character.

JPA limitations

Generic functionality in HPM makes heavy usage of the JPA annotations as well as the meta models of the JPA provider etc. For arbitrary reasons we had to limit the available JPA functionalities a bit.

- Only bi-directional relations are allowed currently (so every `@OneToMany` annotation should have corresponding `@ManyToOne`)
- Only 1:n or 1:1 relations are allowed, direct n:m mappings are not supported
- The primary key of every jpa entity must be a `bigint` column named `id` which uses either a sequence or is an identity column
- Only use attribute level annotations, no annotation on getters and setters
- Only use `sets` for collections, no lists!
- Only use objects, no primitives (Long instead of long!)
- Every JPA entity must also have an implemented `toString()` method which omits `null` values.
Note: only return the string values of own members, no containment references (not the parent, no children) should be logged in the `toString()` method

Available interfaces/base classes which must or can be used

Persistable

The persistable interface is a marker interface which identifies a class as a HPM persistable object. All persistence model classes must implement this interface, either directly or through a base class or interface.

Deletable

The deletable class provides members, methods and annotations for the SoftDelete feature which has been introduced with HPM 5.3. JPA Entities which must support the soft delete feature should subclass this class, directly or indirectly.

The deletable class also declares all JPA filters for the SoftDelete feature (aliveOnly, deadOnly, deadSince), so they do not have to be declared in the model class itself. For performance reasons, all @OneToMany attributes should have the following JPA filters (for SoftDelete feature) annotated:

```
@Filters( { @Filter( name = Persistable.FILTER_ALIVE_ONLY ), @Filter( name =
Persistable.FILTER_DEAD_ONLY ), @Filter( name =
Persistable.FILTER_DEAD_SINCE ) } )
```

Auditable

The auditable class extends the deletable class and adds members, methods and annotations for the physical audit log functionality which has also been introduced in HPM 5.3.

JPA Entities which must store when and by whom they have been created, changed, deleted should subclass from Auditable.

The database tables must have the following columns in order to be able to use the Auditable interface properly:

Name	Datatype	Nullable
CreationUserID	bigint	yes
CreationTimestamp	datetime	yes
ModificationUserID	bigint	yes
ModificationTimestamp	datetime	yes
DeletionUserID	bigint	yes
DeletionTimestamp	datetime	yes

AuditLogSupport

The AuditLogSupport class reflects the standard business audit log entity which has been introduced for every root entity of HPM 5.3. In contrast to the Auditable this class represents all members for the corresponding AuditLogEntity like ArticleAuditLog or CatalogAuditLog etc. The AuditLogSupport is no physical audit log only (which stores audit log in a simple but effective way for every table) but a more business driven feature. Clients can implement arbitrary AuditLogProviders which have then their own logging algorithms. AuditLogSupport is only available for root entities!

The database table which implements the audit log support must follow the following scheme:

The AuditLog table naming convention is: <Tablename of root entity>AuditLog (e.G. ArticleAuditLog, StructureGroupAuditLog etc.)

The table has a foreign key to it's corresponding root/revision root table and an unique index on the same and the classifier column. Therefore we have a 1:n relationship, where there can be only one record per item and classifier.

Name	Datatype	Nullable
Classifier	nvarchar(50)	no
CreationUserID	bigint	yes
CreationTimestamp	datetime	yes
ModificationUserID	bigint	yes
ModificationTimestamp	datetime	yes
DeletionUserID	bigint	yes
DeletionTimestamp	datetime	yes
Res_Text250_01	nvarchar(250)	yes
Res_Text250_02	nvarchar(250)	yes

Name	Datatype	Nullable
Res_Text250_03	nvarchar(250)	yes
Res_Text250_04	nvarchar(250)	yes
Res_Text2G_01	ntext	yes
Res_Text2G_02	ntext	yes
Res_Text2G_03	ntext	yes
Res_Text2G_04	ntext	yes
Res_Text2G_05	ntext	yes
Res_Text2G_06	ntext	yes
Res_Text2G_07	ntext	yes
Res_Text2G_08	ntext	yes
Res_Bit_01	bit	yes
Res_Bit_02	bit	yes
Res_Bit_03	bit	yes
Res_Bit_04	bit	yes

Name	Datatype	Nullable
Res_Bit_05	bit	yes
Res_Bit_06	bit	yes
Res_Bit_07	bit	yes
Res_Bit_08	bit	yes
Res_Int_01	bigint	yes
Res_Int_02	bigint	yes
Res_Int_03	bigint	yes
Res_Int_04	bigint	yes
Res_Int_05	bigint	yes
Res_Int_06	bigint	yes
Res_Int_07	bigint	yes
Res_Int_08	bigint	yes
Res_BigDecimal12_01	decimal(12,4)	yes
Res_BigDecimal12_02	decimal(12,4)	yes

Name	Datatype	Nullable
Res_BigDecimal12_03	decimal(12,4)	yes
Res_BigDecimal12_04	decimal(12,4)	yes
Res_BigDecimal16_01	decimal(16,6)	yes
Res_BigDecimal16_02	decimal(16,6)	yes
Res_BigDecimal16_03	decimal(16,6)	yes
Res_BigDecimal16_04	decimal(16,6)	yes
Res_DateTime_01	datetime	yes
Res_DateTime_02	datetime	yes
Res_DateTime_03	datetime	yes
Res_DateTime_04	datetime	yes

Example

An abbreviated example of a typical HPM persistence model class:

[com.heiler.ppm.article.db.model.ArticleAttribute](#) (see page 216)

```

@Entity
@DynamicInsert
@DynamicUpdate
@Table( name = "`ArticleAttribute`" )
public class ArticleAttribute extends Auditable implements FieldHandled
{
    @Id

```

```

    @GeneratedValue( strategy = GenerationType.AUTO, generator = "SEQ_ArticleAttribute"
    )
    @SequenceGenerator( name = "SEQ_ArticleAttribute", sequenceName =
    "`SEQ_ArticleAttribute`", allocationSize = 50 )
    @Column( name = "`ID`" )
    private Long id
    = null;

    @ManyToOne( fetch = FetchType.LAZY )
    @JoinColumn( name = "`ArticleRevisionID`", nullable = false )
    private ArticleRevision articleRevision
    = null;

    @Column( name = "`Identifier`" )
    private String identifier
    = null;

    @Column( name = "`NameInKeyLanguage`" )
    private String nameInKeyLanguage
    = null;

    @OneToMany( mappedBy = "articleAttribute", cascade = CascadeType.ALL, fetch =
    FetchType.EAGER )
    @Filters( { @Filter( name = Persistable.FILTER_ALIVE_ONLY ), @Filter( name =
    Persistable.FILTER_DEAD_ONLY ),
        @Filter( name = Persistable.FILTER_DEAD_SINCE ) } )
    @Fetch( FetchType.SUBSELECT )
    private Set< ArticleAttributeLang > articleAttributeLangs
    = new HashSet< ArticleAttributeLang >();

    public Long getId()
    {
        return this.id;
    }

    public void setId( Long newId )
    {
        this.id = newId;
    }

    public Set< ArticleAttributeLang > getArticleAttributeLangs()
    {
        return this.articleAttributeLangs;
    }

    public void setArticleAttributeLangs( Set< ArticleAttributeLang >
    newArticleAttributeLangs )

```

```

{
    this.articleAttributeLangs = newArticleAttributeLangs;
}

//[...] getters and setters for all other fields too [...]

@Override
public String toString()
{
    StringBuilder builder = new StringBuilder();
    builder.append( "ArticleAttribute [" ); //$NON-NLS-1$
    if ( this.id != null )
    {
        builder.append( "id=" ); //$NON-NLS-1$
        builder.append( this.id );
        builder.append( ", " ); //$NON-NLS-1$
    }
    if ( this.identifier != null )
    {
        builder.append( "identifier=" ); //$NON-NLS-1$
        builder.append( this.identifier );
        builder.append( ", " ); //$NON-NLS-1$
    }
    if ( this.nameInKeyLanguage != null )
    {
        builder.append( "nameInKeyLanguage=" ); //$NON-NLS-1$
        builder.append( this.nameInKeyLanguage );
        builder.append( ", " ); //$NON-NLS-1$
    }
    builder.append( "]" ); //$NON-NLS-1$
    return builder.toString();
}
}

```

Extension point

Since JPA is not able to have modular persistence units HPM implemented it's own persistence unit configuration using the `com.heiler.ppm.persistence.db.mapping` extension point. JPA entity classes or packages containing JPA entities must be contributed to this extension point, for each persistence unit they should be available for.



The usual JPA implementations require some sort of `persistence.xml` file which contains the packages or classes which are available as JPA entities. Since HPM is a highly modular application we couldn't provide this kind of file at all - therefore we configured the JPA provider programmatically

6.7.7.3 Named queries

Named queries offer a lot advantages over dynamic string concatenation or just normal HQL strings.

- they will be cached by hibernate and JDBC which improves performance drastically in case they will be used often
- they will be compiled at server startup which makes sure that - in case the server starts at all - all queries are at least syntactically correct
- they are easy to use and to find if you follow our conventions

Conventions

Named queries have to be defined as annotations - usually they will be integrated in the JPA Entities which is, not a very good practice in our opinion. Therefore we use separate classes which only define the named queries and might also contain methods to apply the query to a jpa entity manager or something similar. Classes which contain named queries must be located in the `.query` package and should be named by the application area for which they contain queries of course.

Example

[com.heiler.ppm.article.db.query.ArticleRelationJPQL](#) (see page 216)

```
/** @formatter:off*/
@NamedQueries( {
    @NamedQuery( name = ArticleRelationJPQL.FIND_ARTICLES_BY_CATALOGS,
        query = "SELECT ar.catalogId, ar.article.id, ar.entityId
"
                + " FROM ArticleRevision ar "
                + " WHERE ar.catalogId IN (:catalogIds) ")
    } )
@MappedSuperclass
/** @formatter:on */
public class ArticleRelationJPQL
{
    public static final String FIND_ARTICLES_BY_CATALOGS =
    "hpm.articles.findArticlesByCatalogs"; //NON-NLS-1$
    public static final String PARAM_CATALOG_ID = "catalogIds";
    //NON-NLS-1$

    /**
     * Finds all articles which belong to the given catalogs
     * @return query with articleRevision.catalogId, articleRevision.article.id,
     articleRevision.entityId
     */
    public static Query findArticlesByCatalogs( EntityManager entityManager,
        Collection< Long > catalogIds )
    {
        Query query = entityManager.createNamedQuery( FIND_ARTICLES_BY_CATALOGS );
        query.setParameter( PARAM_CATALOG_ID, catalogIds );
    }
}
```

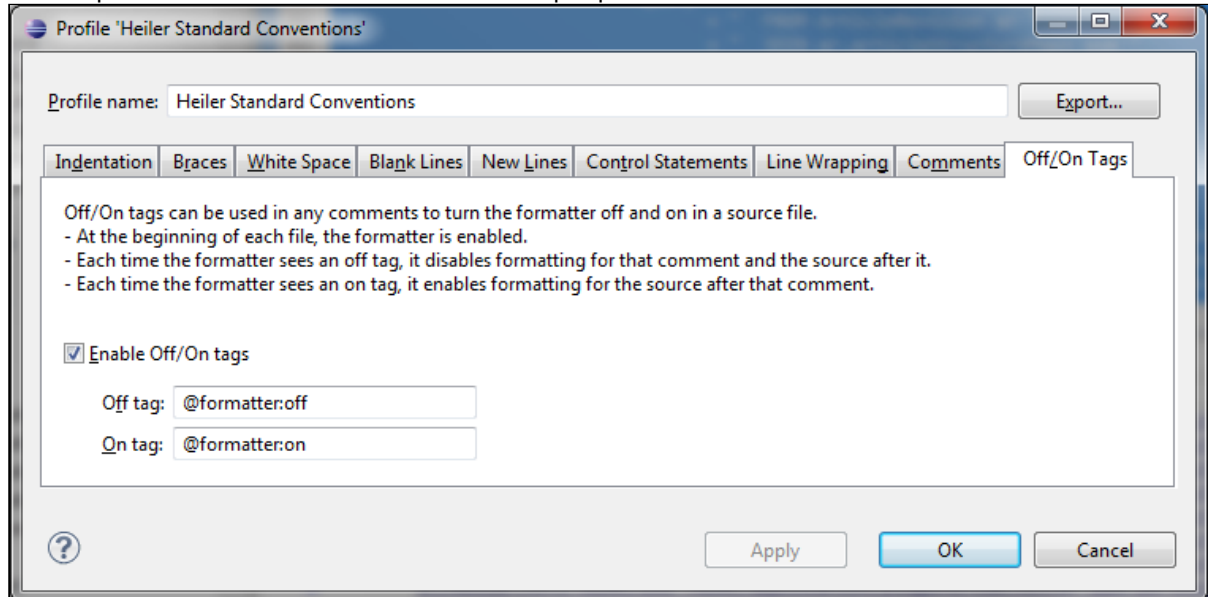
```

    return query;
}
}

```

- Formatter settings

The `@formatter:off` javadoc annotation will prevent the eclipse code formatting from reformatting the named queries. This must be turned on in the Eclipse preferences:



- MappedSuperclass

The `@MappedSuperclass` annotation makes sure that the class will be evaluated by the JPA provider although it is no real JPA entity.

Extension point

Classes containing named queries are also some kind of JPA entity and therefore must also be made visible to the persistence unit configuration by contributing the files or packages to the `com.heiler.ppm.persistence.db.mapping` extension point.

6.7.7.4 3rd party libraries

Since HPM 5.3 we rely on the JPA standard in all areas where possible. Sometime it is still necessary to have access to the underlying JPA provider's classes. HPM uses Hibernate as JPA provider which is located in the `org.hibernate` bundle in the target platform. You can obtain the Hibernate session using:

```

DataSourceRegistry dsr = PersisteLayer.getContext().getDataSourceRegistry();
EntityManager entityManager = dsr.aquireManager(...);
Session session = entityManager.unwrap(Session.class);

```

To obtain the session factory use:

```
DataSourceRegistry dsr = PersisteLayer.getContext().getDataSourceRegistry();
EntityManagerFactory factory = dsr.getFactory(...);
SessionFactory sessionFactory = factory.unwrap(SessionFactory.class);
```

6.7.8 Debug Persistence Layer

6.7.8.1 Log traces

Consider using existing logging facility to debug persistence layer. You can enable trace messages for the whole persistence layer using "com.heiler.ppm.persistence" logging category in log4j.xml:

```
<category name="com.heiler.ppm.persistence">
  <priority value="TRACE"/>
</category>
```

The log output can be too noisy, to make it more clear you can enable logging only for the main components (use TRACE level).

Most usefull:

- **com.heiler.ppm.persistence.server.internal.mediator** - trace all read/write operations (including entity type names and db keys) performed by the default mediators implementations
- **com.heiler.ppm.persistence.server.internal.persister.ORMPersister** - trace business dataGraphs content and derived change summaries which are used to write data into DB

Other components:

- **com.heiler.ppm.persistence.db.internal.DataSourceRegistryImpl** - data source contributions / Hibernate initialization
- **com.heiler.ppm.persistence.db.internal.PersistenceUnitInfoFactory** - persistence entities(classes) initialization / JPA initialization
- **com.heiler.ppm.persistence.db.internal.metamodel.PersistenceMetamodelImpl** - persistence metamodel initialization (can be very noisy)
- **com.heiler.ppm.persistence.server.internal.persister.PersisterExtensionRegistryImpl** - trace persistence layer extensions instantiation
- **com.heiler.ppm.persistence.server.internal.mapping.RepositoryMappingFactory** - repository persistence mappings configuration
- **com.heiler.ppm.persistence.server.internal.filter** - autogenerated hibernate filters
- **com.heiler.ppm.fragment.server.hql** - fragment based on HQL statements

Search:

- **com.heiler.ppm.search.server.internal.service** - to analyse problems/error/exceptions in the entity search engine. In the log messages you can find dump of the search expression tree, HQL, SQL and HQL parameters. Dumped XML can be used to reproduce search queries in unit tests (use HPM xml serilization utils to convert it into search query).

External:

- **org.hibernate** - hibernate logging which can be useful to find configuration problems caused by default settings in hibernate

6.7.8.2 Extension tracing

To trace persistence layer contributed extensions set the following categories to TRACE level:

- **com.heiler.ppm.persistence.server.internal.persister.ORMPersister**
- **com.heiler.ppm.persistence.server.internal.persister.PersisterExtensionUtils**
- **com.heiler.ppm.persistence.server.mediator.utils.DeleterUtils**
- **com.heiler.ppm.persistence.server.mediator.utils.MediatorUtils**
- **com.heiler.ppm.persistence.server.internal.mediator.GenSubMediator**

6.7.8.3 SQL traces

SQL traces are sql commands produced by hibernate and UDA. To active SQL traces for all data sources and all data access methods set **db.default.debug.show_sql** property to *true* (in server.properties).

You can configure SQL traces pro data sources by setting *hibernate.show_sql* in the hibernate config (<dataSource>.properties.template.<DB> file) and <property name="debug" value="true" /> in the UDA config (uda-config.xml.template.<DB> file)

To trace sql parameter values produced by HQL processor set **org.hibernate.type** log4j category to TRACE.

6.7.8.4 JDBC Driver trace

Microsoft JDBC driver logs

Microsoft JDBC driver is using java standard logging facilities. See online docs how to enable and configure logs [http://msdn.microsoft.com/en-us/library/ms378517\(v=sql.90\).aspx](http://msdn.microsoft.com/en-us/library/ms378517(v=sql.90).aspx)

6.8 Infrastructure

Platform components which are used in different layers (configuraiton, serialization, helpers etc)

6.8.1 Identifier Generator

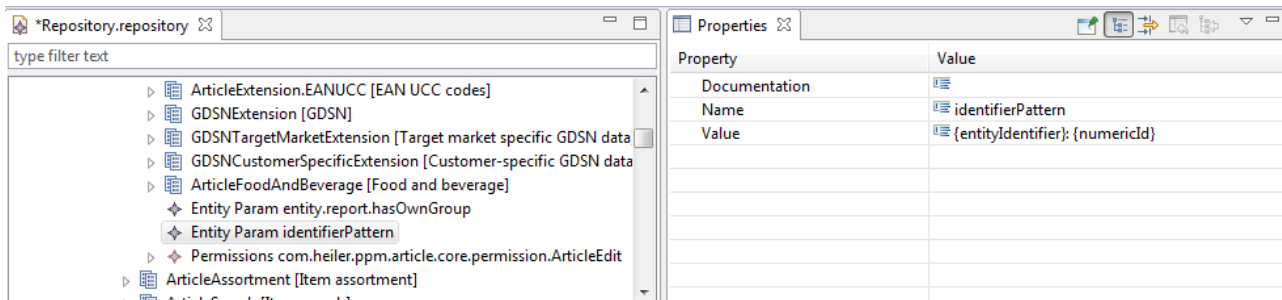
A new object in Product 360 always requires a unique identifier. In a multi server environment it should be guaranteed that a created identifier is always unique.

With Version 8 a new `IdentifierGenerator` service is available which provides the identifiers for root entities (sub entities are currently not supported).

6.8.1.1 Default Implementation

The default implementation to generate identifiers can be configured per entity.

By adding an `entityParam` child element on an entity in the repository you can specify an `identifierPattern` with two possible placeholders.



`{entityIdentifier}` = the identifier of the repository entity

`{numericId}` = the cluster wide unique numeric id with up to 16 digits.

Those placeholders can be combined as you like, for example: `{entityIdentifier}: {numericId}` would result in something like: **Article: 1538732512590398**

Without any entity specific configuration the default pattern which consists of the prefix(entity identifier) and an incremented number separated by an underscore. E.G. **Article_1538732512590398**

6.8.1.2 numericId

Starting with 8.1.1.04 the generic logic to generate the numeric id part of the identifier uses a database sequence.

With the first start of the servers the sequence " `SEQ_IdentifierGenerator` " will be initialized in the **MAIN** database schema.

The length, start and max value of the numeric id can be configured in the server.properties file:

```
# The database sequence 'SEQ_IdentifierGenerator' in the MAIN schema is
responsible for providing unique identifiers.
# If the sequence does not exist yet it will be created and initialized with
the following values.
identifierGenerator.length = 16
identifierGenerator.startWith =
identifierGenerator.maxValue =
```

If no `startWith` is given, the current system time in milliseconds is used as start value (this is due to backwards compatibility reasons with the old id generator logic).
If no `maxValue` is defined, the maximum of the sequence is used.

Since querying a sequence for the next number is always a database roundtrip, we request a batch of ids from it. By default we request 1000 ids.

In the plugin_customization.ini file you can adjust this setting by adding the preference (we do not recommend to adjust it without good reason thou):

```
# Specifies the increment size for the identifier generator i.e. the amount of
numbers the generator may hold in
# memory before it needs to execute a query to the database to get the next
number. If the number is set too small,
# the generator will have to query the database too often, but if the number is
set too high it may result in
# bigger gaps in the generated numbers (the numbers hold by the generator will
be lost after server shutdown).
# Default: 1000
com.heiler.ppm.std.server/identifierGenerator.incrementBy = 1000
```

As every server is requesting his own ids and those ids are shared across root entities and also the ids are requested in packages there are some things to know about those ids.

The only guarantee the system will give is that the id is unique across the whole cluster. There will be gaps in the ids especially if you just look for a single entity.

There will be ids not used at all since a shutdown of the server will not release already loaded ids.

6.8.1.3 Custom Implementation

A custom identifier generator can be contributed to the extension point:

```
com.heiler.ppm.std.core.identifierGenerator
```

identifier: The identifier for the contributed generator. To overwrite a root entity generator, use the root entity identifier e.g. 'Article'

class: The class implementing the `com.heiler.ppm.std.core.identifier.IdentifierGenerator` interface.

You might want to use `IdentifierGeneratorBaseImpl` as base class which already provides a method to retrieve a cluster unique number (unique for all servers in the cluster)

6.8.2 Initializer Framework

The initializer framework is part of the application startup layer of PIM Core with respect to both the application server and the rich client. An initializer basically represents logic for starting up a certain component, e.g. initializing an in-memory cache, so is executed at most once during a node's life-cycle. In order to provide a certain order the startup logic is executed in and to guarantee that certain

components are already existent, each initializer is executed in the context of a so-called start level. For instance, the persistence layer's initializer must be executed before the HPM repository is loaded etc.

- [Extension Point](#) (see page 232)
- [Disposers](#) (see page 232)

6.8.2.1 Extension Point

Initializers are contributed to extension point **com.heiler.ppm.init.core.initializers**, whereas the following information needs to be provided:

- **id** - The unique identifier of the initializer.
- **class** - The class implementing the Initializer interface, containing the actual startup logic.
- **start-level** - The level it is to be executed in, given the following order and levels:
 - STD_BOOT: Marks the initial level for booting the PIM Core platform.
 - STD_READ_ONLY_00 - STD_READ_ONLY_90: Marks standard start up levels for initialization logic which only reads from the DB and does NOT alter data.
 - STD_READ_WRITE_00 - STD_READ_WRITE_90: Marks standard start up levels for initialization logic which can also write to the DB.
 - CUSTOM_00 - CUSTOM_40: Initializers for custom startup logic can use these levels.
 - STD_READY: Marks the initialization to be finished.
- **type** - The type of the initializer, given the following modes:
 - SERVER (default): Initializer is executed every time a server is started. Applicable for components which have to be started and available on all server instances.
 - CLUSTER_SINGLETON: Initializer is executed only once for the entire server cluster. Applicable for components with static initialization logic which is only needed to be executed once in the network life cycle (both single and multi-server environment).
 - CLUSTER_SINGLETON_WITH_FAILOVER: Initializer is executed only once at a time, but is executed on a different server when the previous node failed and is not available anymore. Applicable for components which have a kind of "singleton service" character for the server cluster in a multi-server environment. Component's functionality must be taken over by another instance in case of failure.
- **nodeRole** (optional) - Role which the server instance must have in order to execute the corresponding initializer. This can be set in order to confine initialization logic only on server instances with a dedicated role.



Note that custom initializers are only allowed to use the CUSTOM_XX start-levels! Otherwise the initialization order is going to be jeopardized and correct application startup cannot be guaranteed.

6.8.2.2 Disposers

When orderly shutting down a server instance, some components need to dispose the resources they occupied and might need to do additional cleanup logic. The right means for such tasks is inherently provided by the initializer framework as well. Contributed initializer classes can be extended with cleanup

logic by implementing the *Disposer* interface. When doing so, the platform container keeps such instances in the correct order during startup and reversely executes the disposal logic when shutting down the server.

6.8.3 File Transfer

The File Transfer framework in the Product 360 is used to distribute files between the different nodes, e.g. from client to server or within different components of one server, etc. It is an abstraction of the various protocols (http, shared access, cloud storages like Amazon S3) which can be used to operate with files in order to make this a more transparent API.

- [File Storage](#) (see page 233)
- [Extension Point](#) (see page 234)

6.8.3.1 File Storage

File storages represent an abstract location for storing/accessing files, independent of the actual used protocol. They can be contributed via an [Extension Point](#) (see page 234).

Every contributed file storage needs a provider, which knows how the corresponding storages can be initialized and disposed, and which is able to return the proper storage.

Up from version 8, SMB is the default file storage provider. Nevertheless you can contribute your own provider.

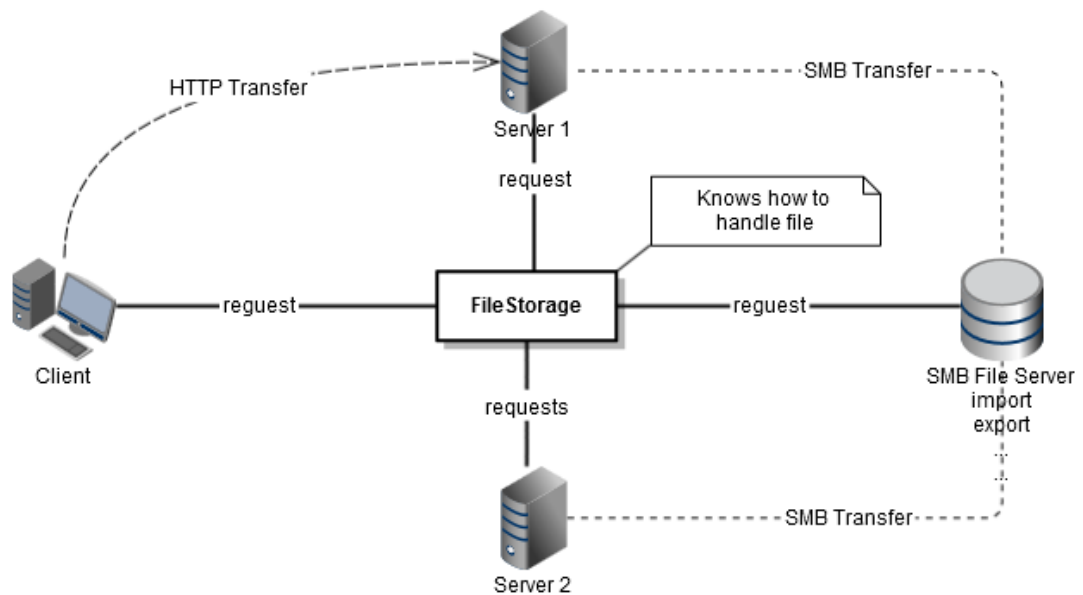
To provide a flexible way to configure a file storage with a provider, you must define **each** contributed file storage with a provider in the `server.properties`. Our default implementation, the SMB storage provider, requires the additional information of the local path. So the declaration for the import file storage could look like this:

filestorage.import = SMB

filestorage.import.path = D:\storage\import

Contributed storages in the default:

- export
- import
- mime
- dataquality
- bpm
- shared



Product 360

6.8.3.2 Extension Point

File storages and its provider can be contributed to the extension point **com.heiler.ppm.filetransfer.core.fileStorage**

Storage:

- **identifier** The identifier for the storage.

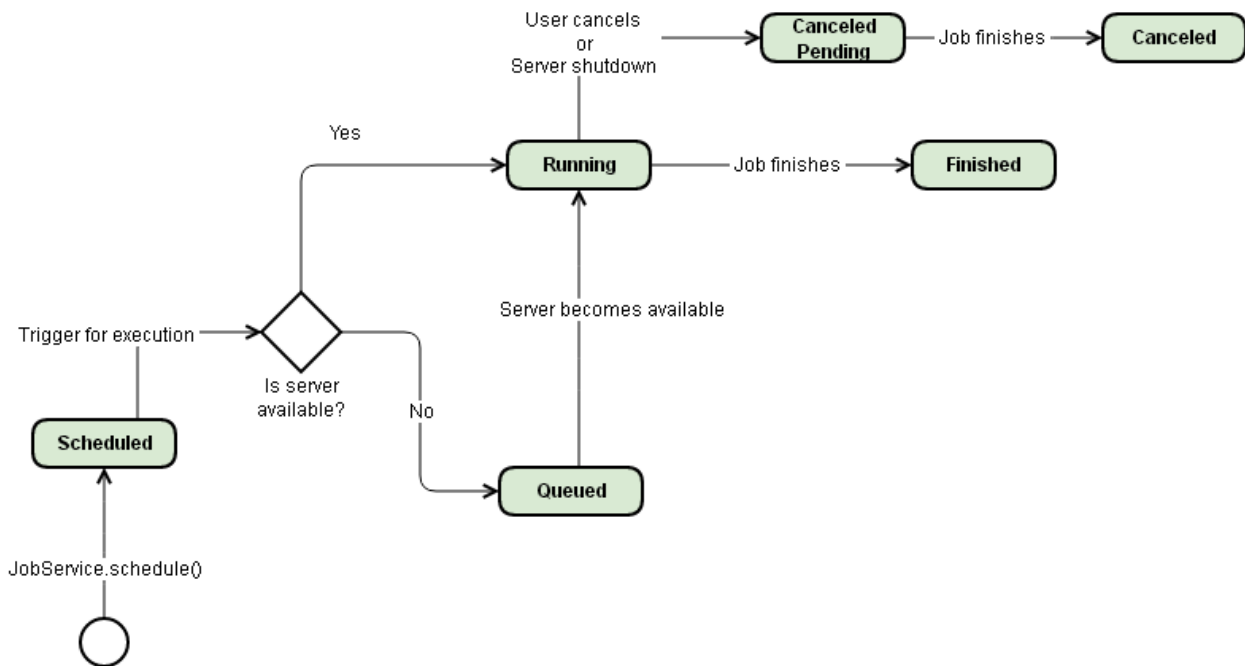
Provider:

- **identifier** The identifier for the storage provider. Contributed provider by default is: SMB
- **class** The class implementing the interface **FileStorageLifecycle**, which is used to initialize and dispose the storage

6.8.4 Server Job Execution

6.8.4.1 Server Job States

Diagram shows the different states of the server jobs



6.8.4.2 Load Balancing

In a multi server environment it is essential to balance the load within all available servers so that they have an equally workload. Since version 8.0.05 there is a server job distributor which calls a selection logic (`JobServerSelector`) to find the next available server to schedule a job. The default implementation `RoundRobinSelector` determines the next available server with the round robin principle. If a server is fully occupied with the maximum of running jobs, the server will be skipped. If all `JOB_SERVER` are fully occupied or only a `CLIENT_SERVER` is available, then the jobs will be queued and executed as soon as the next server is available.

6.8.4.3 Customizing

Implement custom job

Custom jobs can be contributed by using the extension points

`com.heiler.ppm.job.server.ServerJobType` (for the server) and

`com.heiler.ppm.job.ui.ServerJobType` (for the desktop client). For details see the documentation of the extension point.

Custom jobs are not allowed to have the job group `SystemJob`.

Implement custom JobServerSelector

The default implementation `RoundRobinSelector` of the server selector can be customized by the extension point: `com.heiler.ppm.job.server.JobServerSelector`. Therefore the interface `JobServerSelector` has to be implemented. The method of the `JobServerSelector` is called right before the job is going to be executed, so the implementation can decide on which server the job should run.

6.8.4.4 Special handling for Non Scheduled jobs

Jobs like `FindReplace` and `SetValue` use temporary reports which are truncated on server restart. If these jobs are queued and the server is restarted, these jobs are canceled without being rescheduled.

6.8.4.5 Preferences

`plugin_customization.ini`

`com.heiler.ppm.job.server/maxRunningServerJobs`: Defines the maximum of jobs that can run at the same time on a `JOB_SERVER` (excluding system jobs). Default is set to 40 Jobs.

`NetworkConfig.xml`

Server jobs like imports and exports can only be scheduled to run on servers with the role `JOB_SERVER`. This role is defined in the `NetworkConfig.xml`

```
<default-role>CLIENTS_SERVER</default-role>
```

```
<default-role>JOB_SERVER</default-role>
```

6.8.5 Security

6.8.5.1 Encryption of secure information

Product 360 supports the encryption of secure information like passwords in configuration files. The encryption will be executed only if the secure information in the configuration files is enclosed by corresponding markers.

There's a default implementation using AES-256. It is used if encryption is needed and no custom implementation is available.

Custom implementation

Implementation

The interface `EncryptionService` provided by `com.heiler.ppm.encryption` has to be implemented:

```
public interface EncryptionService
{
    /**
     * Encrypts the given {@link CharSequence}.
     * @param charSequenceToEncrypt The {@link CharSequence} to be encrypted. Must not
     * be <code>null</code>.
     * @return The encrypted {@link CharSequence}, never <code>null</code>.
     */
    public CharSequence encrypt( CharSequence charSequenceToEncrypt ) throws
EncryptionServiceException;

    /**
     * Decrypts the given {@link CharSequence}.
     * @param charSequenceToDecrypt The {@link CharSequence} to be decrypted. Must not
     * be <code>null</code>.
     * @return The decrypted {@link CharSequence}, never <code>null</code>.
     */
    public CharSequence decrypt( CharSequence charSequenceToDecrypt ) throws
EncryptionServiceException;
}
```

Note: the implementation has to be provided in all server components like Product 360 application server, Control Center, Audit Trail, ... which you're using.

Installation

The compiled `EncryptionService` implementation has to be provided in the classpath, the start parameter `ppm.encryptionService` contains the name of the class and tells the corresponding application to use that implementation.

Example:

Product 360 server - wrapper.conf

```
wrapper.java.additional.40 = -Dppm.encryptionService =
custom.encryption.encryptionService
```

6.8.6 XML Serialization

6.8.6.1 Overview

Platform provides facilities to save java objects in xml format. Most of the platform objects use this approach (reports, report queries, search queries, import/export configurations and mappings, assortment, value providers, job framework objects). The main purpose of xml serialization is to provide JMV independent, 'human readable' and controllable format to store java objects. If you need to persist service objects and you don't need to perform search or match queries on a set of persisted objects then HPM xml serialization is a correct approach to use. A typical example of xml serialization is a user specific configuration object which should be saved between user login sessions.

What xml serialization is not:

- It is not a domain objects persistence
- It is not the HPM communication framework marshaling

For JEE folks: HPM XML Serialization has the same purposes as Java Architecture for XML Binding (JAXB) technology.

6.8.6.2 Services and Interfaces

All necessary services and interfaces can be found in the `com.heiler.ppm.xml` bundle.

This bundle also contains xml 3rd party libraries which are required in order xml serialization to function properly. Please, don't change or upgrade these libraries because other versions (or spec implementaions) will most likely not work and cause hard-to-diagnose bugs (as of 5.3). These libraries are located in the /endorsed folder of the bundle which is configured as a `java.endorsed.dirs` java setting. This setting effectively redefines system xml-apis implementations.

Creating xml serializable classes

As of HPM 5.3 the correct way to use xml serialization is to implement `DomPersistalbe2` (see page 238) interface (for serialization) and `DomPersistalbe2` (see page 238) interface (for deserializazion).

`DomPersistableFactory` implementations shell be contributed using `com.heiler.ppm.xml.domPersistableFactories`. We recommend to use internal classes for that.

In `DomPersistable2#writeToDom(DomPersistWriteContext context, Element anchorElement)` implementation you can decide which fields will be serialized and in which form: as xml attribute or as a xml sub element. If any of the fields types you want to serialize already implements `DomPersistable2` then the xml serializer will recognize it and will delegate serialization of this type to its `writeToDom(...)` method. Such approach helps to serialize complex object trees where every object is responsible only for serialization of its own data only. This feature is quite helpfull when your java objects contain references to platform objects which already support xml serialization (for example `SearchParameters`). During serialization platform is able to recognize same object instances and

serialize them only once (this will save space and time). Xml representations of the duplicate objects will only contain references to the master object (in xml string this is called sys-id).

To serialize and deserialize java object you can use `DomPersistUtils.toXml(Object,String)` and `DomPersistUtils.fromXml(String)` methods.

Serialization example

```
public class MyConfig implements DomPersistable2
{
    private static final String XML_MY_NAME = "myName";

    private static final String XML_ABOUT_ME = "about";

    private static final String XML_FRIENDS = "myFriendsSearch";

    private String myName;

    private String aboutMe;

    private SearchParameters myFriendsSearchParams;

    public MyConfig( String myName, String aboutMe, SearchParameters
myFriendsSearchParams )
    {
        this.myName = myName;
        this.aboutMe = aboutMe;
        this.myFriendsSearchParams = myFriendsSearchParams;
    }

    @Override
    public String getFactoryId()
    {
        DomPersistUtils.warnUnregistered( Factory.IDENTIFIER, Factory.class );
        return Factory.IDENTIFIER;
    }

    @Override
    public void writeToDom( DomPersistWriteContext context, Element anchorElement )
    throws Exception
    {
        //same name as xml attribute
        anchorElement.addAttribute( XML_MY_NAME, this.myName );
        //save long text as a body of <about>...</about>
        context.writeToDom( this.aboutMe, anchorElement.addElement( XML_ABOUT_ME ) );
        //save search params under <myFriendsSearch>...</myFriendsSearch>
        context.writeToDom( this.myFriendsSearchParams,
anchorElement.addElement( XML_FRIENDS ) );
    }
}
```

```

public static final class Factory implements DomPersistableFactory
{
    public static final String IDENTIFIER = "hlr.me.MyConfig"; //$NON-NLS-1$

    @Override
    public Object createFromDom( DomPersistReadContext context, Element
anchorElement ) throws Exception
    {
        String myName = DomPersistUtils.getAttributeValue( anchorElement, XML_MY_NAME,
true, null );

        Element aboutMeXmlElement = anchorElement.element( XML_ABOUT_ME );
        String aboutMe = context.createFromDom( aboutMeXmlElement );

        Element fredSearchXmlElement = anchorElement.element( XML_FRIENDS );
        SearchParameters searchParams = context.createFromDom( fredSearchXmlElement );

        return new MyConfig( myName, aboutMe, searchParams );
    }
}

//contribution in plugin.xml
<extension
    point="com.heiler.ppm.xml.domPersistableFactories">
    <factory
        class="com.heiler.ppm.me.MyConfig$Factory"
        id="hlr.me.MyConfig">
    </factory>
</extension>

```

Serialize objects without DomPersistable2 interface.

You can serilize objects which do not implement DomPersistable2 interface by implementing `DomPersister` (see page 238) interface in a separate class and contributing it using `persister` element in the `com.heiler.ppm.xml.domPersistableFactories` extension point.

Deprecated interfaces

HPM platform also still contains two deprecated xml searilization machanismis. `DomPersistable` (see page 238) and `PersistableElement` (see page 238) interfaces. As well as corresponding extension point `com.heiler.ppm.xml.elementFactories`. You should not use these mechanisms, but you can still serialize these objects in a context of DomPersistable2 implementaions. For example you can simply call

Serialize DomPersistable

```

public void writeToDom( DomPersistWriteContext context, Element anchorElement )
throws Exception
{
    ...
    Element e = anchorElement.addElement( XML_LEGACY );
    context.writeToDom( this.oldDomPersistable, e );
    ...
}

```

add serialization framework will delegate xml serializaion of this object to the deprecated (but still) working algorithm. So you can mix different xml serilization approaches in the same object tree. However this is highly unrecommended.

The main disadvatage of `DomPersistable` and `PerstableElement` aproaches is that they store full qualified class names. So if the class name has been changed between versions then deserialization of the old xml structures will be impossible because the class name is used by refelection on deserialization.

6.8.6.3 Migration to DomPersistable2

As a general rule you have to migrate your old `DomPersistable` implementaions to `DomPersistable2` approach. Because you also need to support HPM versions migrations you need to follow some rules:

- New `DomPersistable2` implementaion of your class must use the same xml structure as the original `DomPersistable` implementation.
- New `DomPersistable2` implementaion must have the same full qualified name as the original `DomPersistable` implementation.
- New `DomPersistable2` implementaion must have an empty argument public constructor.

Data migration will happen automatically as soon as a cycle of `read_domPersistable_format - save_in_domPersistable2_format` will be sucessfully finished.

It is important to test new `DomPersistable2` with old data from `DomPersistable` serialization. With tests you can be sure that migration with live data will not go wrong in production environment. For that you need to build test cases which take the old XML string and read it using new `DomPersistable2` implementaion. Consider this as a required junit test case when you make refactoring of `DomPersistable` to `DomPersistable2`.

6.8.7 Message Queue Framework

Since Product 360 10.0 there is a generic implementation for interacting with message queues.

Currently only the ActiveMQ Message Queue is supported.

6.8.7.1 Message queue definition in the server.properties file:

Configuring an existing queue or adding a new queue for Product 360 is done by extending the section "Message Queue Settings"

An explanation of each property can be found here: [Server configuration](#)

Example: Add a new queue with the identifier `custom_queue` in the server.properties

server.properties

```
queue.custom_queue.type = ActiveMQ
queue.custom_queue.writer.count = <thread_count_for_writing>
queue.custom_queue.consumer.count = <thread_count_for_consuming>
queue.custom_queue.url = <queue_URL>
queue.custom_queue.username = <username_for_the_queue>
queue.custom_queue.password = <password_for_the_username>
queue.custom_queue.message.format = <message_format_for_writing>
queue.custom_queue.name = P360_CUSTOM_QUEUE
queue.custom_queue.label = Custom queue for new stuff
```

6.8.7.2 Reading from a message queue:

To read from our new custom queue we will need the `MessageQueueService` from the package `com.heiler.ppm.messagequeue.server.v2`

Depending on the object you want to read from the queue you can use the appropriate `getMessageQueue` method

- Case 1: There is only one type of objects to read from the queue

only read the same object type

```
MessageQueueService messageQueueService =
MessageQueueComponent.getMessageQueueService();
MessageQueue< CustomObject> queue = messageQueueService.getMessageQueue(
"custom_queue", CustomObject.class );
MessageQueueReader< CustomObject> customReader = queue.getReader();
```

- Case 2: There can be different types of objects to read from the queue

read different objects

```

MessageQueueService messageQueueService =
MessageQueueComponent.getMessageQueueService();
MessageQueue< Object> queue = messageQueueService.getMessageQueue( "custom_queue",
new MultiCustomObjectConverter() );
MessageQueueReader< Object> customReader = queue.getReader();

```

In all cases it is necessary to specify a function which describes what happens to the message when it gets consumed from the message queue. This function has to be the parameter for the method call `onMessage`

onMessage function

```

customReader.onMessage( this::processMessage );

[...]

private void processMessage( QueueMessage< CustomObject> message )
{
    try
    {
        //custom code
    }
    finally
    {
        message.acknowledge();
    }
}

```

Each session processes the message with the **ActiveMQSession.INDIVIDUAL_ACKNOWLEDGE** mode, it means, that the **acknowledge** method of the message should be always called explicitly to acknowledge message. If this method is not called, the message will be delivered again after the Product 360 server restarts.

Only messages which fulfill the following criteria can be accessed via the function of the `onMessage` method. If one of the criteria is not fulfilled, an error will be logged in the Product 360 server and the message will be acknowledged automatically.

- In the JMS message there must be a JMS property with the key `MessageFormat`
- In the JMS message there must be a JMS property with the key `User`
- In the JMS message there must be a JMS property with the key `Password` or a JMS property with the key `Signature`
- The user which is specified in the JMS property has the action right to use queue communications

- The JMS message body can be successfully converted to the specified object

Authentication methods

Only messages which can successfully validate a user against the Product 360 Server are processed.

1. username / password
The given JMS properties for `User` and `Password` are used to authenticate against the Product 360 Server. Additionally this user has to have the action right for queue communication.
2. username / signature
Alternatively to a password, a signature can be filled as JMS message property which will then be checked by the Product 360 Server automatically and the user will be authenticated.
The signature approach can **only** be used when messages are written by the Product 360 server itself and also consumed by a Product 360 server.
To use this signature for other purposes it is recommended to use the `SignatureService` class.

6.8.7.3 Writing to a message queue:

For writing to a message queue it is also first needed to specify to which queue the messages will be written.

Get a `MessageQueue` object from the `MessageQueueService` of the package `com.heiler.ppm.messagequeue.server.v2`.

From the `MessageQueue` object you can get the `MessageQueueWriter` object.

only write the same object type

```
MessageQueueService messageQueueService =
MessageQueueComponent.getMessageQueueService();
MessageQueue< CustomObject> queue = this.messageQueueService.getMessageQueue(
"custom_queue",CustomObject.class );
MessageQueueWriter< CustomObject> customWriter = queue.getWriter();
```

The `MessageQueueWriter` can write any implementation of `QueueMessage` to the queue. By default we provide the `JMSMessage` implementation of the `QueueMessage` interface.

Example to write a sample Message of type `CustomObject`

write JMS message

```
CustomObject payload = new CustomObject();
```



```
QueueMessage< CustomObject> message = new JMSMessage( payload );
customWriter.write( message );
```

It is also possible to write custom JMS properties to the message by using the `addHeader` method.

add custom header to JMS message

```
QueueMessage< CustomObject> message = new JMSMessage( payload );
message.addHeader( "customMessageProperty", "value" );
customWriter.write( message );
```

For setting the JMS Properties `Username`, `Password`, `JMSCorrelationID` and priority there are predefined methods available.

write predefined properties to message

```
QueueMessage< CustomObject> message = new JMSMessage( payload );
message.setCorrelationId(5);
message.setUser("TinaTen");
message.setPassword("Infa123");
customWriter.write( message );
```

When writing messages from the P360 Server to the queue and also reading from the same queue with a P360 Server the authentication via signing can be used like described in section "Authentication methods".

To use this feature only the username has to be set and additionally the method `setSignMessage` has to be set to `true`.

use signing

```
QueueMessage< CustomObject> message = new JMSMessage( payload );
message.setCorrelationId(5);
message.setUser("TinaTen");
message.setSignMessage(true);
customWriter.write( message );
```

6.8.7.4

Answering to a specific message queue:

When reading from a queue the generic approach will not send a response automatically. If it should be possible to send a response to a specific queue, the message which got read needs to have a JMS header property with the key `ResponseQueueID`. The value has to be the identifier of the message queue where

the response should be sent to. This identifier can be found in the appropriate queue defined in the `server.properties`.

Example:

Send a message which has a `ResponseQueueID` set. Afterwards we read the message from the queue with the identifier `custom_queue` and want to send a response to the queue with the identifier `response_queue`.

server.properties setup

server.properties

```
queue.custom_queue.type = ${queue.default.type}
queue.custom_queue.writer.count = ${queue.default.writer.count}
queue.custom_queue.consumer.count = ${queue.default.consumer.count}
queue.custom_queue.url = ${queue.default.url}
queue.custom_queue.username = ${queue.default.username}
queue.custom_queue.password = ${queue.default.password}
queue.custom_queue.message.format = JSON
queue.custom_queue.name = P360_CUSTOM_QUEUE
queue.custom_queue.label = Custom queue for new stuff

queue.response_queue.type = ${queue.default.type}
queue.response_queue.writer.count = ${queue.default.writer.count}
queue.response_queue.consumer.count = ${queue.default.consumer.count}
queue.response_queue.url = ${queue.default.url}
queue.response_queue.username = ${queue.default.username}
queue.response_queue.password = ${queue.default.password}
queue.response_queue.message.format = JSON
queue.response_queue.name = P360_RESPONSE_QUEUE
queue.response_queue.label = Queue to respond to

queue.dq.type = ${queue.default.type}
queue.dq.writer.count = ${queue.default.writer.count}
queue.dq.consumer.count = ${queue.default.consumer.count}
queue.dq.url = ${queue.default.url}
queue.dq.username = ${queue.default.username}
queue.dq.password = ${queue.default.password}
queue.dq.message.format = XML
queue.dq.name = P360_DATA_QUALITY
queue.dq.label = Data Quality
```

Writing the message with the JMS header `ResponseQueueID` with the value `response_queue`. This value indicates the identifier of the queue where we want to send the response to.

set ResponseQueueID as JMS property

```
QueueMessage< CustomObject> message = new JMSMessage( payload );
message.setCorrelationId(5);
message.setUser("TinaTen");
message.setSignMessage(true);
message.setResponseQueueID("response_queue");
customWriter.write( message );
```

Processing the message we just sent and sending a response message to a queue**⚠ Objects for Response**

Note that all objects which will be send as a response object need the annotation @XmlElement

The method `sendResponseMessage` will first check if the initial message has a JMS header `ResponseQueueID` if it does not have this property, nothing will be send. If it has this property it will resolve the identifier to the correct queue. The following JMS properties will be automatically filtered out when creating the response message:

- User
- Password
- MessageFormat
- ResponseQueueID
- SuccessTargetService
- ErrorTargetService

If the initial message contained the following JMS properties `ResponseUser` , `ResponsePassword` they will then be transferred to the JMS properties `User` and `Password` of the new response message.

Finally the response object will be converted and send to the queue.

Example: Send an error message with status code 500 to the response queue**send error response**

```
customReader.onMessage( this::processMessage );

MessageQueueService messageQueueService =
MessageQueueComponent.getMessageQueueService()
[...]

private void processMessage( QueueMessage< CustomObject> message )
{
```

```

try
{
    //custom code

}
catch(SomeException e)
{
    ErrorObject errorObject = new ErrorObject( "Exception occurred" );
    Response< String > response = new
Response<>( QueueMessageStatus.INTERNAL_SERVER_ERROR, errorObject );
    messageQueueService.sendResponseMessage(message ,response);
}
finally
{
    message.acknowledge();
}
}

```

Example: Send a custom object to the response queue

send response

```

customReader.onMessage( this::processMessage );

MessageQueueService messageQueueService =
MessageQueueComponent.getMessageQueueService()
[...]

private void processMessage( QueueMessage< CustomObject> message )
{
    try
    {
        //custom code
        MyObject myObject = new MyObject( [...] );
        Response< String > response = new Response<>( QueueMessageStatus.OK,
myObject );
        messageQueueService.sendResponseMessage(message ,response);
    }
    finally
    {
        message.acknowledge();
    }
}

```

7 Desktop Client

This is the documentation space for the Heiler Product Manager user interface.

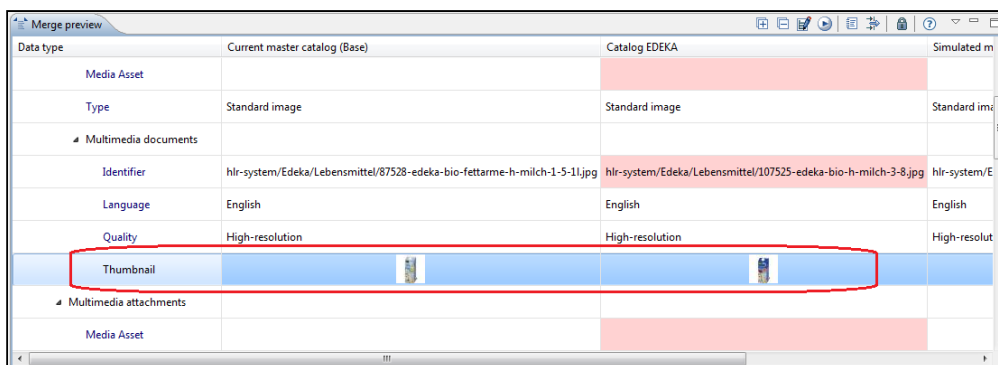
- ["Image preview" - selection handling extensions](#) (see page 249)



- [Automatically set structure group context in the Article Attribute View via customizing](#) (see page 252)
- [Centralized UI settings](#) (see page 252)
- [CKEditor configuration and customization](#) (see page 255)
- [CKEditor troubleshooting](#) (see page 265)
- [Creating custom views](#) (see page 266)
- [Creating custom perspectives](#) (see page 278)
- [Customization of the field handling within views](#) (see page 281)
- [Defining the default perspective](#) (see page 288)
- [Entity specific icons](#) (see page 290)
- [Extension point for SearchView customization](#) (see page 293)
- [HOWTOs](#) (see page 295)
- [ListModelTableViews: selection behavior](#) (see page 302)
- [Long-running processes](#) (see page 305) — This site shows how to handle the long-running processes in the UI.
- [Possibility to disable standard UI elements](#) (see page 307)
- [Possibility to replace the execution class of a view contribution](#) (see page 316)
- [Richtext conversion](#) (see page 318)
- [Richtext validation](#) (see page 323)
- [Table View Layout](#) (see page 327)
- [UI Style Guide](#) (see page 327)
- [XULRunner integration for SWT browser](#)

7.1 "Image preview" - selection handling extensions

7.1.1 Motivation

The "Image preview" is a part of the "Multimedia attachments" perspective and shows the preview of a selected image. There are several potential selection providers for this view. Mainly this view was designed to support selections within the "Multimedia attachments" perspective. Since PIM version 8.0 this view supports also the selection of a thumbnail image in the "Merge preview".



Data type	Current master catalog (Base)	Catalog EDEKA	Simulated m
Media Asset			
Type	Standard image	Standard image	Standard ima
✚ Multimedia documents			
Identifier	hlr-system/Edeka/Lebensmittel/87528-edeka-bio-fettarme-h-milch-1-5-1l.jpg	hlr-system/Edeka/Lebensmittel/107525-edeka-bio-h-milch-3-8.jpg	hlr-system/E
Language	English	English	English
Quality	High-resolution	High-resolution	High-resolut
Thumbnail			
✚ Multimedia attachments			
Media Asset			

The challenge in the selection handling in the "Image preview" is, that there are many several types of selected elements which should be supported by this view. The "Image preview" extracts the image itself from the selected element. The "logic", how to extract the image, can't be located in the "Image preview" since this view doesn't "know" all possible types of selected elements

(e.g. `ArticleMediaAssetDocument` in case of "Merge preview" selections). To prevent a "dependency salad" and to provide a possibility to define custom selections which should be handled by the "Image preview", we added a new extension point: **`com.heiler.ppm.mediaasset.ui.imagePreviewSelectionProviders`**

7.1.2 Extension point definition

The extension point **com.heiler.ppm.mediaasset.ui.imagePreviewSelectionProviders** provides only one contribution element: **selectionProvider**. This element provides the *ID* and the *implementation class*. The contribution implementations should implement the interface **com.heiler.ppm.mediaasset.ui.views.ImagePreviewSelectionProvider** which contains following methods:

```
public interface ImagePreviewSelectionProvider
{
    /**
     * Returns the {@link Predicate} which can be used to determine if a specific
     selection is provided
     * @return a selection predicate (never <code>null</code>)
     */
    Predicate< ISelection > getSelectionPredicate();
    /**
     * Determines if this selection provider can resolve any information for the given
     selection.
     * @param viewPartId the ID of the selection view part (should not be <code>null</
     code>)
     * @param selection the selection to resolve (should not be <code>null</code>)
     * @return <code>true</code> if this provider is able to resolve the given
     selection
     */
    boolean canResolveSelection( String viewPartId, ISelection selection );
    /**
     * Resolves the given selection and returns the image identifier and the quality of
     the selected image
     * @param selection the selection to resolve
     * @return a {@link SelectionInfo} object or <code>null</code> if the given
     selection can't be resolved
     */
    SelectionInfo resolveSelection( Object viewPart, ISelection selection );
}
```

- The method **getSelectionPredicate()** provides a *predicate* which defines whether the "Image preview" should react to a specific selection. Usually the selection predicates are entity type based. For example the predicate for the mediaasset selection looks like this:

```
@Override
public Predicate< ISelection > getSelectionPredicate()
{
    return new

    SelectionPredicateFactory.EntityTypePredicate( MediaAssetCoreConst.ENTITY_TYPE_
    MEDIAASSET ); // <- "MediaAssetType"
```

```
}
```

- The method `canResolveSelection()` determines whether the specific *selection provider* supports the current selection. The implementations of this method can also use the parameter `viewPartId` to decide it. For example if the selection comes from the `MediaAssetsOfItemTableView` the corresponding *selection provider* can resolve it and extract the image identifier from the given selection. The "Image preview" iterates over all registered `ImagePreviewSelectionProvider` and calls this method. The first one which returns `true` will be used to resolve the current selection.
- The last method: `resolveSelection()` cares about the extraction of the image information from the given selection. It receives also the view instance, because some implementations of this method need any specific logic of the causing view. The returned object `SelectionInfo` wraps all required information which is necessary to load the image (image identifier, quality and - optionally - the entity for the coloring of the description area in the "Image preview", e.g. "Item"/"Product"/"Variant").

7.1.3 Standard implementations


In the standard PIM there is already a few implementations of `ImagePreviewSelectionProvider` which are contributed to the new extension point:

- **MediaAssetsOfItemImagePreviewSelectionProvider**
This implementation handles all selections of *media assets* within the "Multimedia attachments" view. The information required for the loading of the image will be extracted from the **first** document of the *media asset*. This implementation was located previously directly in the "Image preview" class and was moved now to `com.heiler.ppm.mediaasset.ui.views.MediaAssetsOfItemImagePreviewSelectionProvider`.
- **MediaAssetsDocumentsImagePreviewSelectionProvider**
This implementation handles all selections of *media asset documents* within the "Multimedia documents" view. This implementation was located previously directly in the "Image preview" class and was moved now to `com.heiler.ppm.mediaasset.ui.views.MediaAssetsDocumentsImagePreviewSelectionProvider`.
- **DocumentsOfCategoryImagePreviewSelectionProvider**
This implementation handles all selections of the *file data* within the "Documents" view. This implementation was located previously directly in the "Image preview" class and was moved now to `com.heiler.ppm.mediaasset.ui.views.DocumentsOfCategoryImagePreviewSelectionProvider`.
- **ArticleMediaAssetDocumentImagePreviewSelectionProvider**
This implementation is view-independent. It handles all selections of type `ArticleMediaAssetDocumentType`. Currently the "Merge preview" notifies such selections if the user clicks on a thumbnail image.
- **MediaAssetFileImagePreviewSelectionProvider**
This implementation handles all selections of the *media asset file* within the "Media Asset File" view.

7.1.4 Custom implementations

To add a custom implementation of `ImagePreviewSelectionProvider` just contribute to the extension point **com.heiler.ppm.mediaasset.ui.imagePreviewSelectionProviders** and implement all methods of the `ImagePreviewSelectionProvider`. Note that the "Image preview" handles all registered *selection providers* in the same way. This means, if you contribute a new *selection provider* for the same type of selection as one existing standard implementation - then this is random which implementation will be used.

7.2 Automatically set structure group context in the Article Attribute View via customizing

 Since PIM 7.0.05

There is a possibility to automatically set the structure group context in the Article Attribute View via customizing. To accomplish this, implement a subclass of **ArticleAttributeTableView** (and contribute it) and override the method **updateStructureContext()** like this example:

```
@Override
protected void updateStructureContext()
{
    StructureGroupProxy structureGroup =
CustomizingHelper.getStructureGroupOutOfArticle( this.currentEntityProxy );
    setStructureContext( structureGroup.getStructureProxy() );
    setStructureGroupContext( structureGroup );
}
```

Now, every time, a new article is selected, the customized article attribute view sets the structure group context depending on the logic, you implemented in the **updateStructureContext()** method.

7.3 Centralized UI settings

 Since 7.1.05

7.3.1 Reason

In rare occasions it's necessary to perform whole UI customization at one place for some reason, e.g. customer is using font which is installed to system, but not used by PIM.

Starting from 7.1.05 PIM supports this functionality through CSS themes.

7.3.2 Client property

Use `com.heiler.ppm.main/path-to-css-customization-file` client preference for pointing to concrete CSS theme.

7.3.3 How it works

Simply speaking: you specify the SWT class name and necessarily properties for it, for example:

Button background

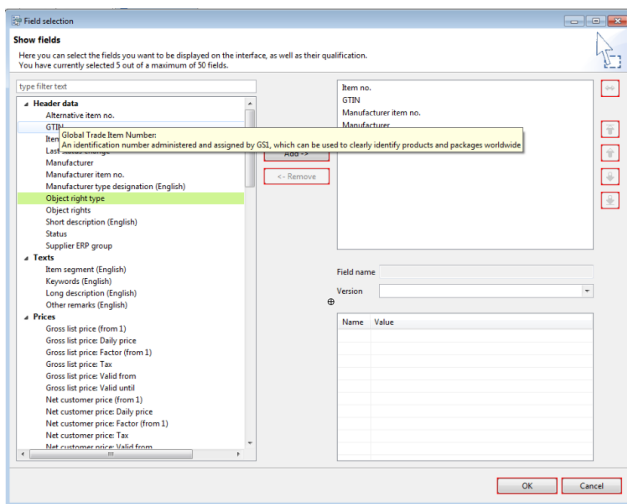
```
Button { background-color: #FF0000 }
```

For sure, CSS allows you to do much more. See [References](#) (see page 255) for more details.

7.3.4 Problems

Sadly, but even CSS customization isn't a golden bullet since operating system can prohibit changing of colors and fonts for some UI elements.

Using above CSS will give this results for buttons:



But this will be a concern only in rare cases, since 90% of requirements and problems could be covered with CSS usage.

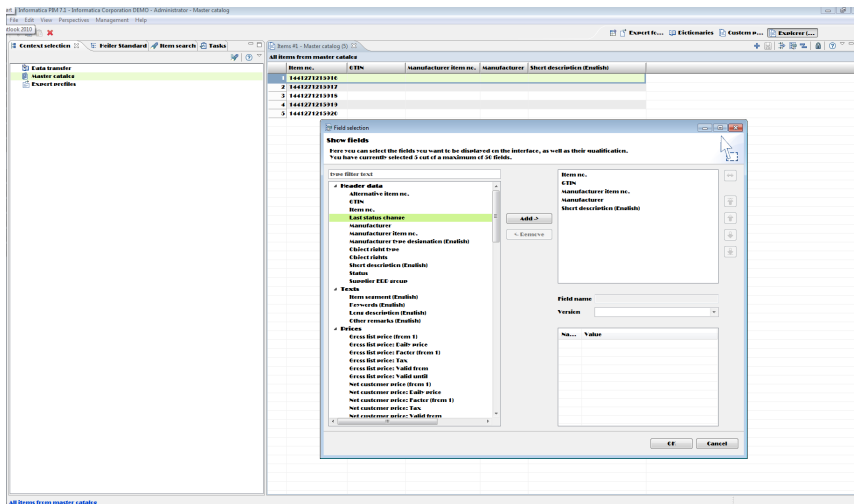
7.3.5 Examples

Changing fonts for whole PIM client:

Global font

```
* { font-family: 'Broadway' }
```

Result:

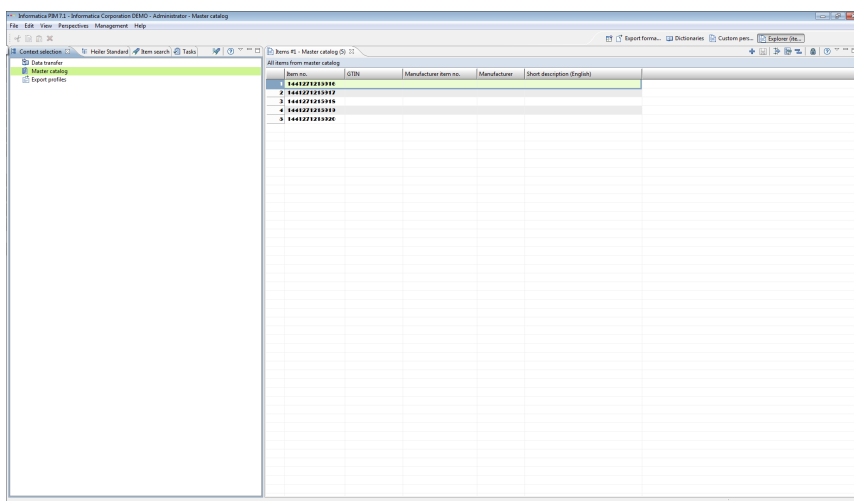


Changing fonts only in tables:

Table font

```
QuasiVirtualTable { font-family: 'Broadway' }
```

Result:



7.3.6 References

- Eclipse CSS support
- Eclipse CSS mapping
- CSS selectors explained

7.3.7 Limitations

Due to a performance issue on loading of a large amount of table items the CSS support was disabled for the SWT class "**TableItem**".

7.4 CKEditor configuration and customization

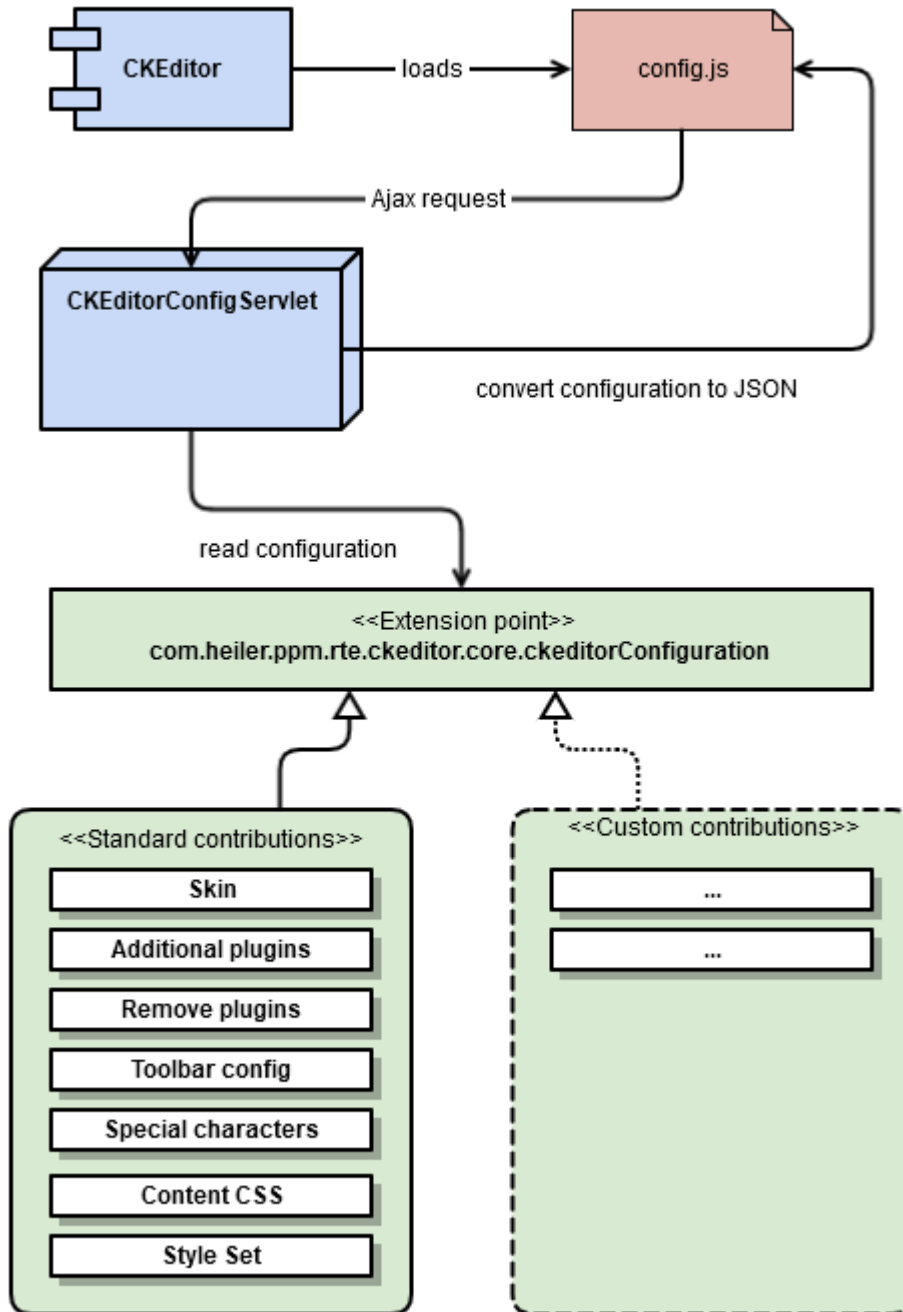


NOTE: all customizations for the *CKEditor* should be located on the **Product 360 Server**, since the configuration of the *CKEditor* is performed on the server!

7.4.1 Motivation

Since the *CKEditor* is embedded into Product 360 as a Rich-Text-Editor, it is used in both clients: Desktop as well as Web. *CKEditor* consists of HTML and JavaScript files and is located on the Server (*com.heiler.ppm.rte.ckeditor.server*). It is available via the HTTP port (e.g. *http://[host]:1501/ckeditor/ckeditor.html*) and can be accessed from the Desktop (called from the SWT *Browser* control) or Web (embedded within the web interface). Since the configuration of the *CKEditor* is quite extensive and had to be done in Desktop as well as in Web, it was simplified and unified in order to configure the *CKEditor* once. Additionally a possibility to customize the *CKEditor* was required.

7.4.2 Big picture



The configuration of the *CKEditor* occurs each time a *CKEditor* instance is instantiated. This is e.g. if the user opens a Rich-Text-Editor in Desktop. The *CKEditor* performs the configuration by using the **config.js** file. This file does some default configurations and then calls an AJAX request to the corresponding servlet. This servlet collects all configuration data and responds it in the form of a JSON string. The **config.js** file puts this JSON string to the *CKEditor* configuration. The servlet (*CKEditorConfigServlet*) uses the extension point

com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration to obtain several parts of the configuration. There is a lot of standard contributions to this extension point, but it can be extended by custom contributions.

7.4.3 Extension point

The extension point **com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration** provides several possibilities to configure/customize the *CKEditor*.



NOTE: all examples used in this article are also available in *SDK Examples* (see also the plugin **com.heiler.ppm.customizing.ckeditor**).

7.4.3.1 ToolbarConfigProvider

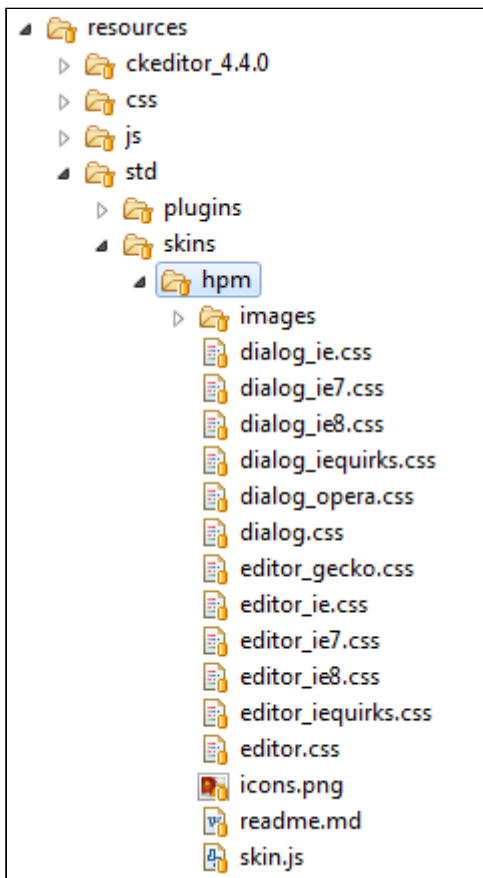
See a separate article how to contribute a custom **ToolbarConfigProvider**: [Toolbar configuration \(see page 264\)](#)

7.4.3.2 SpecialChar

See a separate article how to add additional special characters using the **specialChar** element of the extension point: [Additional special characters \(see page 263\)](#)

7.4.3.3 Skin

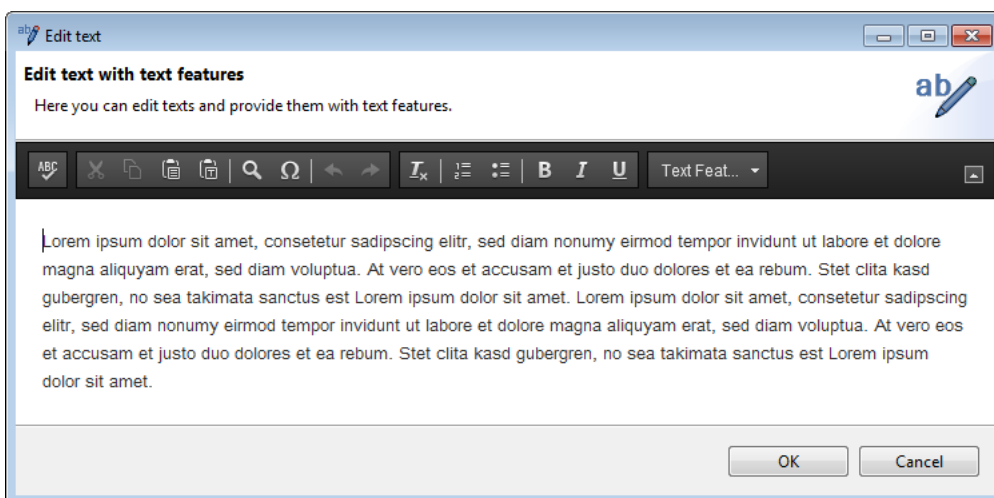
With the "skin" you can define the look&feel of the *CKEditor*. By default we are using the "hpm" skin (based on the "moono" skin which is the "CKEditor 4 Skin Contest Winner"). This skin is located in the folder `/resources/std/skins` of the plugin `com.heiler.ppm.rte.ckeditor.server`.



It is possible to define a custom skin for the *CKEditor*. The skin itself consists of a set of CSS files (see the screenshot above). So you can e.g. copy the "**hpm**" skin, modify the CSS and contribute your custom skin to the extension point.

You can also use any other available skin, which can be downloaded from the official CKEditor site.

Here is an example how to replace the standard skin with "**Moono Dark**":



First you should define a custom *resource* folder for the *CKEditor* (see also [Providing custom resources](#) (see page 262)). Then add to the resource folder a "skin" folder and put there your custom skin (e.g. "moono-dark").

Then use the **skinProvider** element of the extension point to contribute your skin:

```
<extension
  point="com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration">
  <skinProvider
    id="ckeditor.custom.skins.moono-dark"
    order="200"
    skinName="moono-dark"
    skinPath="/ckeditor/custom/skins/moono-dark/">
  </skinProvider>
</extension>
```

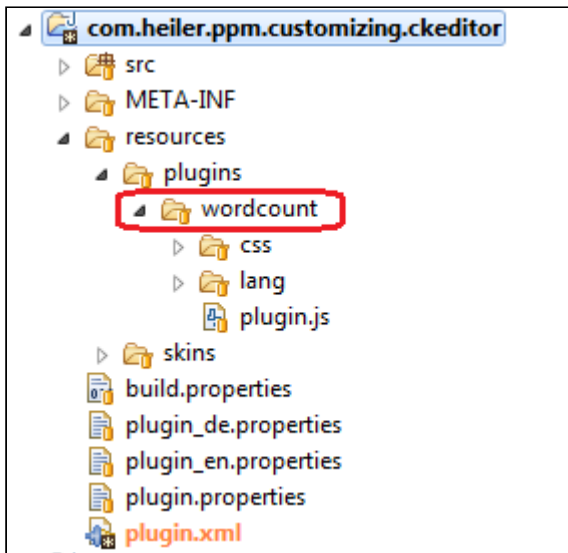
The **order** attribute determines which skin should be used (the "hpm" skin has order 100).

Note that the **skinPath** attribute contains the **alias** of the custom resource folder (see also [Providing custom resources](#) (see page 262)), in this example: `/ckeditor/custom`.

7.4.3.4 Additional plugins

Since the *CKEditor* is plugin-based it is possible to add your custom additional plugins to the editor. Just put your custom plugin to the "plugins" folder within the *resources* folder (see also [Providing custom resources](#) (see page 262)).

As example I added the "wordcount" plugin to the custom plugins folder (see also Word Count & Char Count Plugin):



Now the plugin should be defined as an additional plugin for the *CKEditor*:

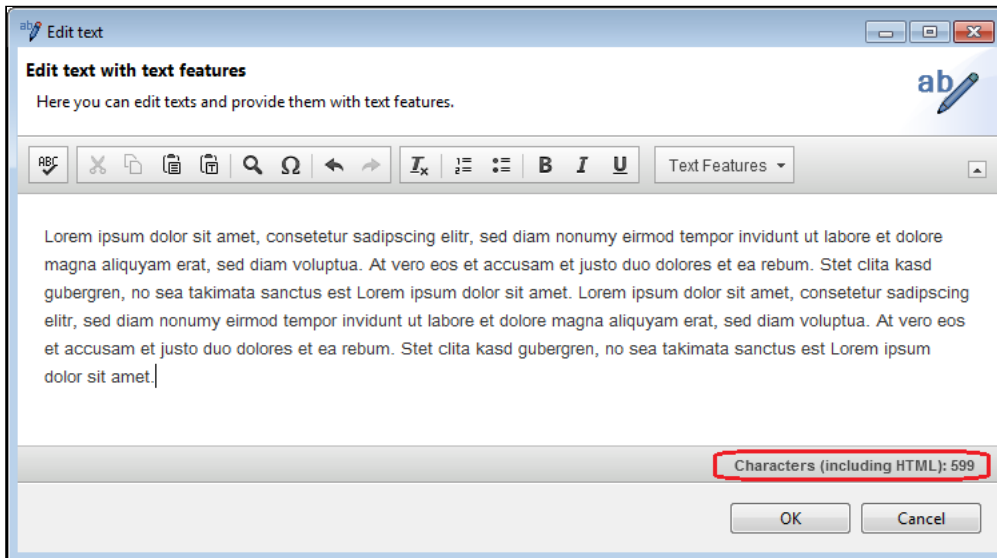
```
<extension
  point="com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration">
  <additionalPlugins>
    <plugin
      name="wordcount">
```

```

        path="/ckeditor/custom/plugins/wordcount/">
    </plugin>
</additionalPlugins>
</extension>

```

As result we have a character counter in the bottom toolbar in the editor:



7.4.3.5 Remove plugins

There is also a possibility to remove installed plugins from the *CKEditor*.

For example to remove the "magicline" plugin, just add following contribution to the extension point:

```

<extension
    point="com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration">
    <removePlugins
        order="200"
        pluginNames="magicline">
    </removePlugins>
</extension>

```

The attribute **pluginNames** contains the names (comma separated) of the plugins which should be removed.

The **order** attribute determines which pluginNames should be used and preferred (the default hpm extension point has order 100).

7.4.3.6 Content CSS

The content.css file is used to set the style of the editor content. For advanced reading see: <http://docs.ckeditor.com/#!/api/CKEDITOR.config-cfg-contentsCss> and have a look to the default content.css file is located in com.heiler.ppm.rte.ckeditor.server/resources/css/contents.css.

This file can be set with the following cssProvider contribution:

```
<extension
  point="com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration">
  <cssProvider
    fileName="contents.css"
    name="com.heiler.ppm.customizing.ckeditor.contents.css"
    path="/ckeditor/custom/"
    order="200">
  </cssProvider>
</extension>
```

The attribute **name** contains the name of this contribution without any further meaning.

The **path** attribute contains the path to the css file relative to the corresponding custom resources described below or to another CKEditor resources.

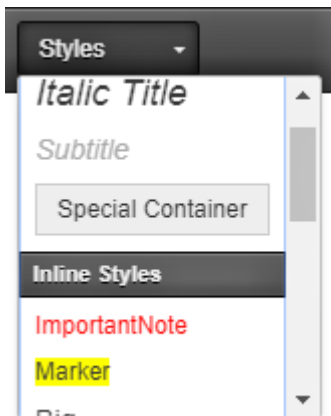
The **fileName** attribute contains the filename of the css file. This attribute is optional and default is 'content.css'.

The **order** attribute of the extension point determines which css file is used as the content css. The standard file has order **100**. So set a higher value to the **order** attribute to make sure, that your custom content css provider will be used.

7.4.3.7 Style Set

Styleset are used for a combobox where you can select special styles which you can apply to a text or a textarea. Stylesets are usual connected with special css classes or styles, so they depends on content.css settings. For further reading have a look to: <http://docs.ckeditor.com/#!/api/CKEDITOR.config-cfg-stylesSet>.

Let's have a look to an example of a styleset combobox:



The styleset file can be set with the following stylesSetFile contribution:

```
<extension
  point="com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration">
```

```
<stylesSetFile
    fileName="styles.js"
    name="customStyleSet"
    path="/ckeditor/custom/"
    order="200">
</stylesSetFile>
</extension>
```

The attribute **name** contains the name of this contribution and is internally used as a key, to determine a particular styleset.

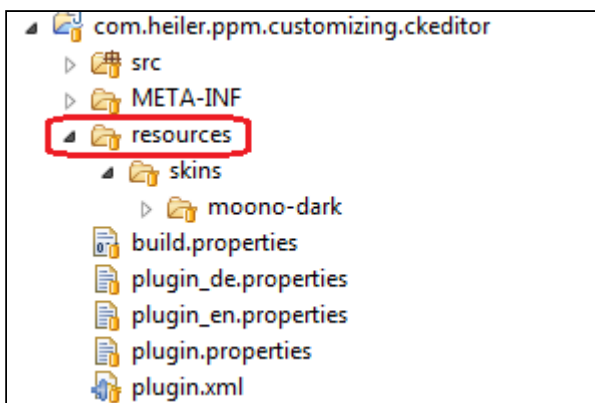
The **path** attribute contains the path to the styleset file relative to the corresponding custom resources described below or to another CKEditor resources.

The **fileName** attribute contains the filename of the styleset file. This attribute is optional and default is 'styles.js'.

The **order** attribute of the extension point determines which styleset file is used. The standard file has order **100**. So set a higher value to the **order** attribute to make sure, that your custom styleset will be used.

7.4.4 Providing custom resources

To provide your custom resources for the *CKEditor* you should define a custom *resource* folder for the *CKEditor*. This folder contains additional resources for the *CKEditor* like custom **skins** or **additional plugins**:



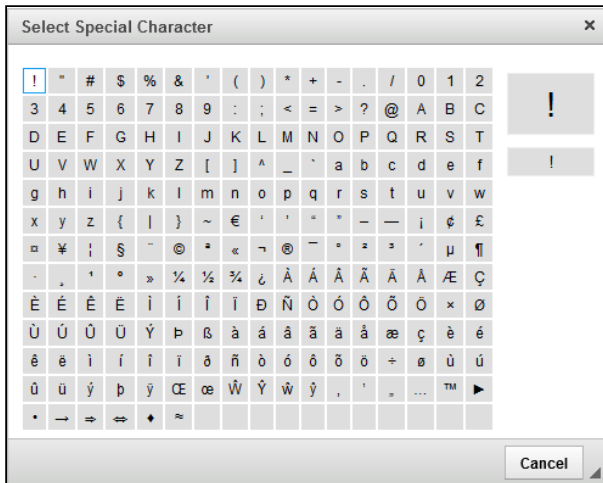
To make this folder available, add following extensions:

```
<extension
    point="org.eclipse.equinox.http.registry.resources">
    <resource
        alias="/ckeditor/custom"
        httpcontextId="ckeditorCustomContext">
    </resource>
</extension>
<extension
    point="org.eclipse.equinox.http.registry.httpcontexts">
    <httpcontext id="ckeditorCustomContext">
        <resource-mapping path="/resources"/>
    </httpcontext>
</extension>
```

```
</httpcontext>
</extension>
```

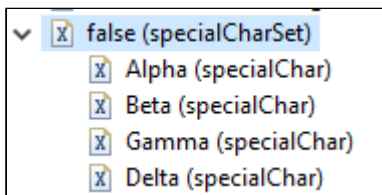
7.4.5 Additional special characters

The CKEditor used in PIM Desktop and PIM Web provides a set of *special characters* which can be pasted to the edited rich-text:



Some customer need additional special characters, which are not contained in this default set. It is possible to add more special characters by using the **specialChar** element of the extension point `"com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration"`. All special characters defined by this extension will be **added at the end** of the default set of the special characters (see the screenshot above) unless the attribute **replace** is set to **true**. To define an additional special character just add a contribution to the extension point `"com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration"`. Each **specialChar** element defines the **order** of the character, the **character** itself and (optional) the **tooltip**. The **order** of the characters is used to sort all additional special characters. But all of them will be added **behind** the default special characters.

Let's look at an example. I will add some *Greek letters* to the special characters (this example is also available in *SDK examples* -> `"com.heiler.ppm.customizing.ckeditor"`):



The XML code for this contribution looks like this:

```
<extension
  point="com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration">
  <specialCharSet replace="false">
    <specialChar
      char="&alpha;"
      order="1"
      tooltip="%specialChar.alpha">
```

```

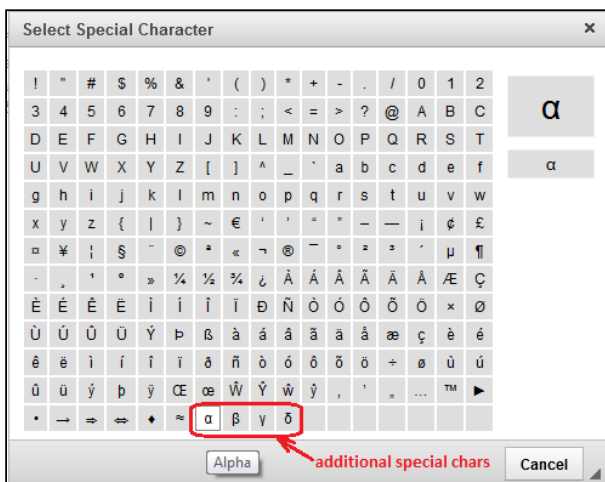
</specialChar>
<specialChar
    char="&beta;"
    order="2"
    tooltip="%specialChar.beta">
</specialChar>
<specialChar
    char="&gamma;"
    order="3"
    tooltip="%specialChar.gamma">
</specialChar>
<specialChar
    char="&delta;"
    order="4"
    tooltip="%specialChar.delta">
</specialChar>
</specialCharSet>
</extension>

```



Note that the character code for e.g. "Alpha" is **α**; but in the *plugin.xml* the sign "&" should be masked. So the value of the attribute "char" in this case will be: **&alpha;**;

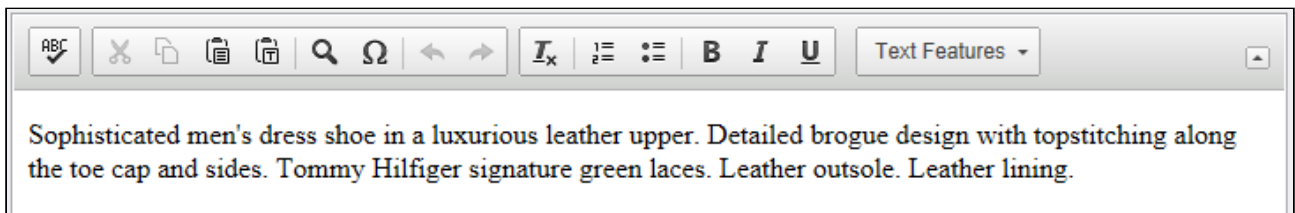
Now if you open the rich-text editor and show the special characters table it should look like this:



For each special character you can define a tooltip which will be shown if the user hovers over the character. If no tooltip is defined, the character string itself will be displayed. The tooltip defining in the contribution is translatable. So set the value of the attribute "tooltip" to the corresponding resource bundle key (e.g. "%specialChar.alpha") by using the "%" sign. Then you can put localized tooltips for different languages in the corresponding "*.properties" files.

7.4.6 Toolbar configuration

The toolbar of the rich text editor in PIM Desktop and PIM Web contains by default a set of several actions:



Since PIM **7.1.01.00** it is possible to customize the rich text editor toolbar using the extension point *com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration*.

To provide your own rich text editor toolbar just implement a *ToolbarConfigProvider* and contribute it using this extension point. The provider class defines which actions and text features are available in the toolbar. There is a base implementation of the provider - *ToolbarConfigProviderBaseImpl* - which should be used for the implementation of a custom toolbar config provider. You can use the standard implementation as a template and as example to make your custom implementation. You find the source code of the standard implementation class in the *SDK examples*:

com.heiler.ppm.customizing.ckeditor.internal.DefaultToolbarConfigProvider.

The **order** attribute of the extension point determines which implementation of the toolbar config provider should be used. The standard implementation has order **100**. So set a higher value to the **order** attribute to make sure, that your custom toolbar config provider will be used:

```
<extension
  point="com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration">
  <toolbarConfigProvider
    class="com.heiler.ppm.customizing.ckeditor.internal.DefaultToolbarConfigP
rovider"
    id="hler.customizing.ckeditor.defaultToolbarConfigProvider"
    order="200">
  </toolbarConfigProvider>
</extension>
```



NOTE: from the PIM version **7.0.05.00** until **7.1.01** the element **toolbarConfigProvider** was located in a separate extension point: *com.heiler.ppm.rte.ckeditor.core.toolbarConfigProviders*. Since the version **7.1.01** it was moved into the more general configuration extension point: *com.heiler.ppm.rte.ckeditor.core.ckeditorConfiguration* (see also [CKEditor configuration and customization](#) (see page 255))

If the customization should affect both PIM Desktop and PIM Web, make the contribution in a *core* plug-in which runs on server and client.

7.5 CKEditor troubleshooting

7.5.1 How to enable the debugging of CKEditor

To see some debug informations from the CKEditor, just add a corresponding Log4J category to the file `log4j.xml` (on the client):

```
<category name="com.heiler.ppm.rte.ckeditor.ui.editor.CKEditor">
  <priority value="DEBUG"/>
</category>
```

After that each time a CKEditor instance is created, the following information will be logged (for example):

```
12:46:51,870 DEBUG [main] [CKEditor] SET BROWSER URL: 'http://DEW179613:1501/
ckeditor/ckeditor.html?
language=en&spellingLang=en_US&loginName=wmichel&editable=true&setReadOnlyWhenReady=f
alse&spellingEnabled=true'
12:46:52,170 DEBUG [main] [CKEditor] CKEDITOR instance is READY
12:46:52,176 DEBUG [main] [CKEditor] [##### CKEDITOR ENVIRONMENT
#####
CKEDITOR.env.ie = 'true'
CKEDITOR.env.isCompatible = 'true'
CKEDITOR.env.quirks = 'false'
CKEDITOR.env.version = '7'
navigator.userAgent: 'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1;
WOW64; Trident/5.0)']
```

The "browser URL" shows which URL with which parameters will be requested to load the CKEditor.

The "CKEditor environment" (available since PIM 7.1.03) contains some useful information about the browser environment, which is embedded into the PIM Desktop. See also <http://docs.ckeditor.com/#!/api/CKEDITOR.env> for more details about specific environment properties.

7.6 Creating custom views

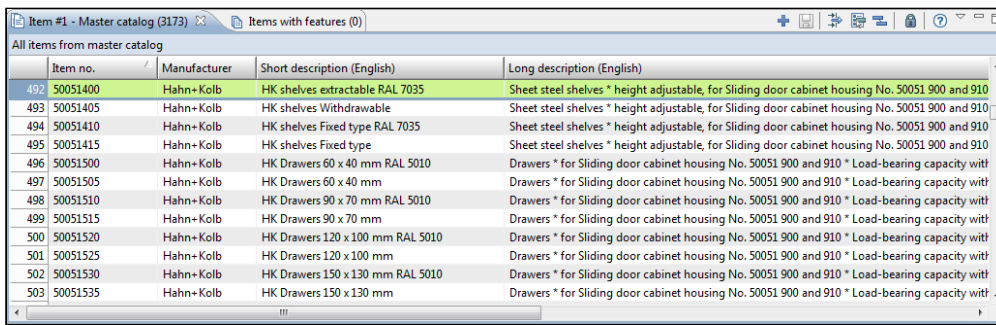
7.6.1 View types

There are two general types of table views: **list model** views and **detail model** views. Read more about **list** and **detail models**: [Data Models](#) (see page 133)

7.6.1.1 List model views

A **list model** view shows the entries of a *root entity* and uses a **list model** to load this entries. The basis class of all **list model** table views is *ListModelTableView*. A **list model** view works only with a repository *root entity*.

The most popular example of a **list model** view is the item view:



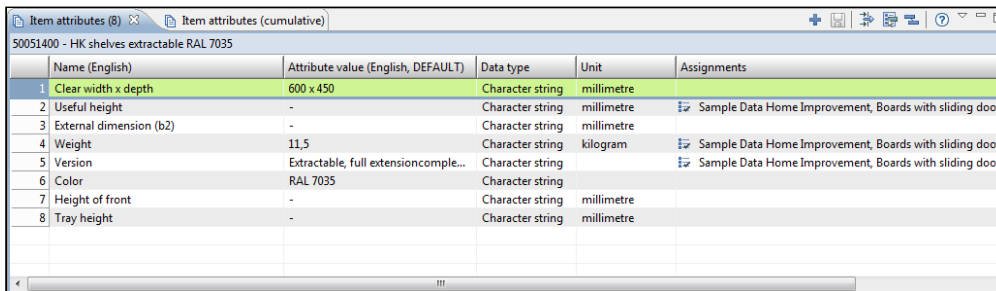
Item no.	Manufacturer	Short description (English)	Long description (English)
492	Hahn+Kolb	HK shelves extractable RAL 7035	Sheet steel shelves * height adjustable, for Sliding door cabinet housing No. 50051 900 and 910
493	Hahn+Kolb	HK shelves Withdrawable	Sheet steel shelves * height adjustable, for Sliding door cabinet housing No. 50051 900 and 910
494	Hahn+Kolb	HK shelves Fixed type RAL 7035	Sheet steel shelves * height adjustable, for Sliding door cabinet housing No. 50051 900 and 910
495	Hahn+Kolb	HK shelves Fixed type	Sheet steel shelves * height adjustable, for Sliding door cabinet housing No. 50051 900 and 910
496	Hahn+Kolb	HK Drawers 60 x 40 mm RAL 5010	Drawers * for Sliding door cabinet housing No. 50051 900 and 910 * Load-bearing capacity with
497	Hahn+Kolb	HK Drawers 60 x 40 mm	Drawers * for Sliding door cabinet housing No. 50051 900 and 910 * Load-bearing capacity with
498	Hahn+Kolb	HK Drawers 90 x 70 mm RAL 5010	Drawers * for Sliding door cabinet housing No. 50051 900 and 910 * Load-bearing capacity with
499	Hahn+Kolb	HK Drawers 90 x 70 mm	Drawers * for Sliding door cabinet housing No. 50051 900 and 910 * Load-bearing capacity with
500	Hahn+Kolb	HK Drawers 120 x 100 mm RAL 5010	Drawers * for Sliding door cabinet housing No. 50051 900 and 910 * Load-bearing capacity with
501	Hahn+Kolb	HK Drawers 120 x 100 mm	Drawers * for Sliding door cabinet housing No. 50051 900 and 910 * Load-bearing capacity with
502	Hahn+Kolb	HK Drawers 150 x 130 mm RAL 5010	Drawers * for Sliding door cabinet housing No. 50051 900 and 910 * Load-bearing capacity with
503	Hahn+Kolb	HK Drawers 150 x 130 mm	Drawers * for Sliding door cabinet housing No. 50051 900 and 910 * Load-bearing capacity with

A **list model** view can load entries for a selection (e.g. *items* of a selected *structure group*) or all available entries of the specified *root entity* (e.g. *structure systems*).

7.6.1.2 Detail model views

A **detail model** view shows the details of an element of the superordinate entity. A detail model view uses an **entity detail model** to load the entries. The basis class of all **detail model** table views is *EntityDetailModelTableView*.

An example of a **detail model** view is the item attribute view:

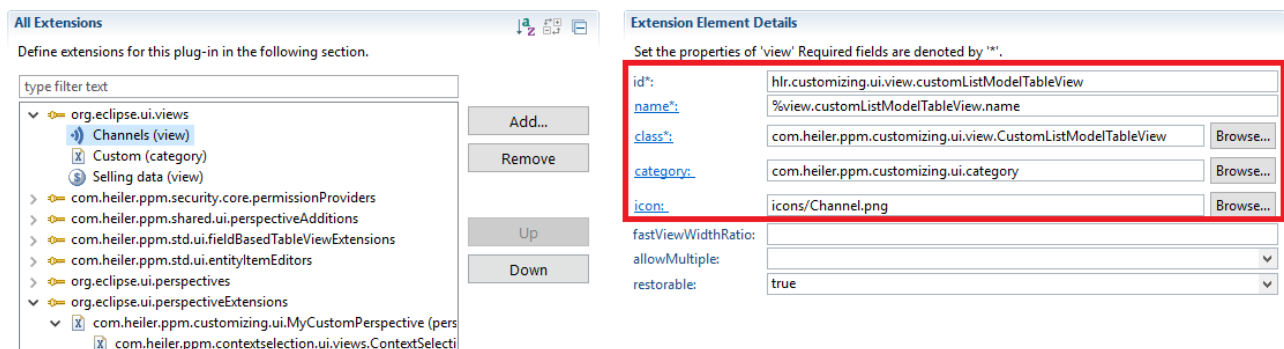


	Name (English)	Attribute value (English, DEFAULT)	Data type	Unit	Assignments
1	Clear width x depth	600 x 450	Character string	millimetre	
2	Useful height	-	Character string	millimetre	Sample Data Home Improvement, Boards with sliding doors,
3	External dimension (b2)	-	Character string	millimetre	
4	Weight	11.5	Character string	kilogram	Sample Data Home Improvement, Boards with sliding doors,
5	Version	Extractable, full extension comple...	Character string		Sample Data Home Improvement, Boards with sliding doors,
6	Color	RAL 7035	Character string		
7	Height of front	-	Character string	millimetre	
8	Tray height	-	Character string	millimetre	

A **detail model** view loads entries for a selected element of the superordinate entity (e.g. *attributes* of a selected *item*).

7.6.2 View contribution

To contribute your custom view use the extension point **"org.eclipse.ui.views"**.



All Extensions

Define extensions for this plug-in in the following section.

type filter text

- org.eclipse.ui.views
 - Channels (view)
 - Custom (category)
 - Selling data (view)
- com.heiler.ppm.security.core.permissionProviders
- com.heiler.ppm.shared.ui.perspectiveAdditions
- com.heiler.ppm.std.ui.fieldBasedTableViewExtensions
- com.heiler.ppm.std.ui.entityItemEditors
- org.eclipse.ui.perspectives
- org.eclipse.ui.perspectiveExtensions
 - com.heiler.ppm.customizing.ui.MyCustomPerspective (pers)
 - com.heiler.ppm.contextselection.ui.views.ContextSelecti

Add...

Remove

Up

Down

Extension Element Details

Set the properties of 'view' Required fields are denoted by '*'.

id*: hlr.customizing.ui.view.customListModelTableView

name*: %view.customListModelTableView.name

class*: com.heiler.ppm.customizing.ui.view.CustomListModelTableView Browse...

category*: com.heiler.ppm.customizing.ui.category Browse...


icon*: icons/Channel.png Browse...

fastViewWidthRatio:

allowMultiple:

restorable: true

In the "Extension Element Details" the properties of your view can be set.

key	value
id	Unique id for the view.
name	<p>Translatable name for the view.</p> <div>  When starting the name with the % sign it is indicating that a variable in a properties file in the plugin directory is used. </div>
class	<p>The class value is build like this:</p> <pre><used factory class>:<used tableview>:<entity></pre> <p><code><used factory class></code> should be "com.heiler.ppm.std.core.contribution.ContributionClassFactory" in every case.</p> <p><code><used tableview></code> depending if the entity is a root entity or a sub entity a different <code>TableView</code> can be used.</p> <p><code><entity></code> has to be the used entity.</p>
category (optional)	Category in which the view can be found in the UI.
icon (optional)	Relative path to the icon which will be shown beside the name of the view.

If you also want to replace an existing standard view with your own custom view - use the extension point "com.heiler.ppm.std.core.contributionClassProviders" (see more details here: [Possibility to replace the execution class of a view contribution](#) (see page 316))

Be attentive defining the "class" attribute of your view contribution. Since PIM version 7.1.08 (resp. 8.0.03.02.00) there is extended "frontend visibility" functionality: Frontend visibility. The generic implementation of the `HiddenElementProvider` hides all views which are contributed for **deactivated** entities. Unfortunately it cannot be differentiated whether an entity for a specific identifier is deactivated or doesn't exist at all. So if your contribution "class" contains any string in place of the entity identifier - the view will not be shown (see the last row in the example below). The entity identifier is expected directly after the view class name following by a colon (":"), e.g.:

```
com.heiler.ppm.std.core.contribution.ContributionClassFactory:com.heiler.ppm.article.
ui.internal.view.ArticleTableView:Article // <- will be hidden if the "Article"
entity is deactivated
com.heiler.ppm.std.core.contribution.ContributionClassFactory:com.heiler.ppm.task.ui.
view.TaskTreeView:Task;task_treeview.html // <- will be hidden if the "Task" entity
is deactivated
com.heiler.ppm.std.core.contribution.ContributionClassFactory:com.heiler.ppm.std.ui.t
able.EntityDetailModelTableView:GDSNTargetMarketExtension // <- will be hidden if the
"GDSNTargetMarketExtension" entity is deactivated
com.heiler.ppm.std.core.contribution.ContributionClassFactory:com.heiler.ppm.contexts
election.ui.views.ContextSelectionTreeView // <-- no entity defined = never hidden
com.heiler.ppm.fulltextsearch.ui.index.views.FulltextSearchIndexViewImpl // <-- no
entity defined = never hidden
com.heiler.ppm.std.core.contribution.ContributionClassFactory:com.heiler.ppm.structur
e.ui.internal.views.mapping.StructureGroupMappingTableView:SourceTree // <-- there is
no repository entity 'SourceTree' - so this view will be hidden
```

7.6.3 PermissionProvider

The visibility of the views is depending on the corresponding action rights (user permissions). To make sure, your custom view can be viewed by the user, add a corresponding entry in the *PermissionProvider*. The permission provider is a class which extends the *com.heiler.ppm.security.core.permission.PermissionProviderBaseImpl*. The permission provider "links" a view or a perspective with a specific action right. Typically the visibility of views and perspectives is depending on the "read" permission of the corresponding entity:

```
/**
 * Adds Permissions for custom views and perspectives
 * @author WMichel
 * @since 7.1
 */
public class CustomUiPermissionProvider extends PermissionProviderBaseImpl
{
    private static final String PERMISSION_BUNDLE_NAME =
"com.heiler.ppm.customizing.ui.permission.permissions"; //$NON-NLS-1$
    public CustomUiPermissionProvider()
    {
        RepositoryService repositoryService = RepositoryComponent.getRepositoryService();
```

```

    Entity channelEntity =
repositoryService.getEntityByIdentifier( ChannelConst.E_CHANNEL );
    String channelReadPermission = channelEntity.getPermissions()
                                                .getRead();

    addViewPermission( "hlr.customizing.ui.view.customListModelTableView",
channelReadPermission ); //$NON-NLS-1$

    Entity articleEntity =
repositoryService.getEntityByIdentifier( ArticleCoreConst.ENTITY_ARTICLE );
    String articleReadPermission = articleEntity.getPermissions()
                                                .getRead();

    addViewPermission( "hlr.customizing.ui.view.customEntityDetailModelTableView",
articleReadPermission ); //$NON-NLS-1$
}
@Override
protected ResourceBundle getBundle( Locale locale )
{
    return ResourceBundle.getBundle( PERMISSION_BUNDLE_NAME, locale );
}
}

```

The permission provider should be also contributed in the corresponding custom plugin:

```

<extension
    point="com.heiler.ppm.security.core.permissionProviders">
    <permissionProvider
        class="com.heiler.ppm.customizing.ui.permission.CustomUiPermissionProvide
r"
        name="Custom UI Permission Provider">
    </permissionProvider>
</extension>

```

7.6.4 Overwriting view methods

The most functionalities of **list model** and **detail model** views are generic and are implemented in the basis view class. Nevertheless it is necessary to override some methods of the view class.

7.6.4.1 initEntityMapping()

Since all **field based** views work entity-based, it is necessary to define for which repository entity the custom view is created. To do this you can override the method *initEntityMapping()* and set the corresponding entity identifier:

```
@Override
protected void initEntityMapping()
{
    setEntityIdentifier( ChannelConst.E_CHANNEL );
}
```

Note: for all **detail model** views you can alternatively define the entity identifier within the view contribution. Just add the entity identifier to the class name after a semicolon (e.g. *com.heiler.ppm.article.ui.internal.view.attribute.ArticleAttributeTableView:ArticleAttribute*)

7.6.4.2 getEntityForPermissions()

The default permissions (*edit/create/delete*) for the views are initialized in a generic way on the basis of the "permission" entity. To determine which entity should be used for the permission initialization the method *getEntityForPermissions()* is called. You can override this method to provide the entity for which the default permissions should be determined:

```
@Override
public Entity getEntityForPermissions()
{
    Entity entity = super.getEntityForPermissions();
    if ( entity == null )
    {
        entity = RepositoryComponent.getRepositoryService()
                                   .getEntityByIdentifier( ChannelConst.E_CHANNEL );
    }
    return entity;
}
```

Note: for all **detail model** views it is not necessary to override this method since the entity which is defined in the view contribution will be used for the permission initialization.

7.6.4.3 initPermissions()

To initialize additional permissions for the view you can override the method *initPermissions()* where the default permissions (*edit/create/delete*) are initialized:

```
@Override
protected void initPermissions()
{
    super.initPermissions(); // <- call 'super' to init default permissions
    LoginToken login = LoginManager.getInstance()
                                   .getLoginToken();

    this.permissionAdd =
login.hasPermission( DictionaryCorePermissionProvider.PERMISSION_WORD_ADD );
    this.permissionSuggest =
login.hasPermission( DictionaryCorePermissionProvider.PERMISSION_WORD_SUGGEST );
}
```

```
}
```

7.6.4.4 createContentProvider()

Especially for **list model** views you should override the method *createContentProvider()* to ensure that the new created entries (e.g. created by other clients) appears in your custom view. Therefore you should create a *ListModelContentProvider* and override the *queryAddNewItem()* method:

```
@Override
protected IStructuredContentProvider createContentProvider()
{
    EntityManager entityManager = EntityManagementComponent.getManagerRegistry()
                                                                .getEntityManager( Channel
Const.ET_CHANNEL );
    return new ListModelContentProvider( entityManager, false, true )
    {
        @Override
        protected boolean queryAddNewItem( EntityOperationOriginator originator,
EntityProxy entityProxy )
            throws Exception
        {
            return true;
        }
    };
}
```

7.6.4.5 createDatasetCreationMechanism()

If you want to customize the *data set new creation* process you can override the method *createDatasetCreationMechanism()*. Here you can provide a custom implementation of the *ICreateMechanism*. For the **list model** views there is a default implementation - *ListModelTableViewCreateMechanism* and for **detail model** views - *EntityDetailModelTableViewCreateMechanism*. If you want, for example, to manipulate the entity properties for the creation of a new entry, you can override the method *getEntityProperties()* in the custom implementation of the *ICreateMechanism*:

```
@Override
protected ICreateMechanism createDatasetCreationMechanism()
{
    return new ListModelTableViewCreateMechanism( this )
    {
        @Override
        protected EntityProperty[] getEntityProperties() throws CoreException
        {
            Entity entity = getEntity();
            EntityProperty[] initValues = new EntityProperty[1];
        }
    };
}
```

```

        initValues[ 0 ] = EntityPropertyFactory.create( entity,
ChannelConst.FT_IDENTIFIER,

String.valueOf( System.currentTimeMillis() ) );
        return initValues;
    }
};
}

```

7.6.4.6 initTableConfigGroup()

If there are several views which work with the same entity they will share the *table layout configuration*, because the *layout configuration* is stored by default under the name of the entity. You can override the method *initTableConfigGroup()* in your custom view to define another *table layout config* identifier:

```

@Override
protected void initTableConfigGroup( String tableConfigGroupIdentifier )
{
    // ignore the tableConfigGroupIdentifier - we need our own!
    tableConfigGroupIdentifier = CONFIG_IDENTIFIER;
    super.initTableConfigGroup( tableConfigGroupIdentifier );
}

```

7.6.5 Customization opportunities

7.6.5.1 Selection handling

If the custom view should react to selection events from the other views, there are several opportunities to manipulate the selection handling in the view.

Selection predicate

The *selection predicate* is used to determine whether a view should react to a specific selection. When a view has been opened by the user it tries to load its content on the basis of the currently selection in other visible views. Therefore the view uses its *selection predicate* and asks the *selection service* to determine which selection is relevant for the view. Read more here about selection predicates in custom views.

selectionChanged()

All **detail model** views handle by default all selection events in the *selectionChanged()* method. The default implementation uses the *selection predicate* to determine if the specific selection is relevant for the view. If true, the detail model of the selected entry will be loaded in the view. For the **list model** views there is no default implementation of the *selectionChanged()* method in the basis class. So if your custom view should handle any selections, you have to implement this method. Here is an example from the *word* view which reacts to the *dictionary* selection and loads all *words* for the selected *dictionary*:

```

@Override
public void selectionChanged( IWorkbenchPart selectionPart, ISelection selection )
{
    if ( selectionPart == this || DataSetUIAdapter.isDirty( getDataSetUIAdapter() ) )
    {
        return;
    }
    // use selection predicate to evaluate the given selection
    Predicate< ISelection > selectionPredicate = getSelectionPredicate();
    if ( selectionPredicate.evaluate( selection ) )
    {
        DictionaryProxy dictionaryProxy =
StdUiUtils.getSingleSelectedElement( selection, DictionaryProxy.class );
        if ( dictionaryProxy != null )
        {
            // Avoid unnecessary requests
            if ( ObjectUtils.equals( dictionaryProxy, this.lastHandledSelection ) )
            {
                return;
            }
            this.lastHandledSelection = dictionaryProxy;
            this.lastReceivedSelectionPart = selectionPart;
            // enable edit mode
            setEditMode( this.permissionEdit ? EDIT_ENABLED : EDIT_DISABLED );
            reloadWordsForPrefix();
        }
    }
    else if ( selectionPart.equals( this.lastReceivedSelectionPart ) )
    {
        handleEmptySelection( selection );
        setEditMode( EDIT_DISABLED );
    }
}

```

7.6.5.2 Data maintenance

Data loading

The most views load the content on the basis of a received selection. But some views have no selection provider. In this case all entries of the corresponding entity should be shown initial in the view. To do that you can override the `postCreate()` method and load the data:

```

@Override
protected void postCreate()
{
    super.postCreate();
    // fill table with all channels
    SearchParameters searchParams = new
SearchParameters( DataSource.MAIN.getIdentifier(), ChannelConst.ET_CHANNEL,

```

```

                                null ); // 'null'
expression means: 'all elements'
    SearchQueryItemList query = new SearchQueryItemList( getEntity(), searchParams );
    queryViewerInput( query );
}

```

Note: if you extend the *GenericDataTableView* this step is already implemented in the basis class.

Data editing

By default the editing behavior of the fields based on the repository settings. The *TableCellAPI* has control about this. It determines whether a specific field is editable, which value is shown in the field and what happens when the value has been changed. For the most views suffices the default implementation in the view basis classes. But if you want to customize the editing behavior in you custom view, you can manipulate the *TableCellAPI*:

```

@Override
protected TableCellAPI createTableCellAPI()
{
    ListModelTableCellAPI cellAPI = new ListModelTableCellAPI( this );
    TableCellValueDecorator valueProvider = new
    TableCellValueDecorator( cellAPI.getValueProvider() )
    {
        @Override
        public boolean canModify( Object rowElement, String colProperty )
        {
            return false;
        }
    };
    cellAPI.setValueProvider( valueProvider );
    return cellAPI;
}

```

In this example all fields of the view will be read-only (independent on the corresponding repository settings).

7.6.6 Example ListModelTableView

In the example below we have a custom **list model** table view for the *Channel* entity. There are following customization in the view:

- we override the method *postCreate()* to load all available channels after the view has been created
- then we override the content provider to ensure that all new created channels (maybe created by other clients) appear in the view automatically
- the *getEntityForPermissions()* method is overridden to provide the entity for the permissions initialization
- in the *initEntityMapping()* method we define the entity for which the view has been created
- and lastly we customize the *ICreateMechanism* to ensure that all new created channels get a default unique identifier (just the current time in millis)

This sample class is also contained in the SDK examples (since 7.1)

```

/**
 * A sample custom implementation of the {@link ListModelTableView} which works with
 the {@link Channel} entity.
 * @author WMichel
 * @since 7.1
 */
public class CustomListModelTableView extends ListModelTableView
{
    public CustomListModelTableView()
    {
        super();
    }
    @Override
    protected void postCreate()
    {
        super.postCreate();
        // fill table with all channels
        SearchParameters searchParams = new
        SearchParameters( DataSource.MAIN.getIdentifier(), ChannelConst.ET_CHANNEL,
                        null ); // 'null'
expression means: 'all elements'
        SearchQueryItemList query = new SearchQueryItemList( getEntity(), searchParams );
        queryViewerInput( query );
    }
    @Override
    protected IStructuredContentProvider createContentProvider()
    {
        EntityManager entityManager = EntityManagementComponent.getManagerRegistry()
                        .getEntityManager( Channel
Const.ET_CHANNEL );
        return new ListModelContentProvider( entityManager, false, true )
        {
            @Override
            protected boolean queryAddNewItem( EntityOperationOriginator originator,
EntityProxy entityProxy )
                throws Exception
            {
                return true;
            }
        };
    }
    @Override
    public Entity getEntityForPermissions()
    {
        Entity entity = super.getEntityForPermissions();
        if ( entity == null )
        {
            entity = RepositoryComponent.getRepositoryService()
                    .getEntityByIdentifier( ChannelConst.E_CHANNEL );
        }
        return entity;
    }
}

```



```

@Override
protected void initEntityMapping()
{
    setEntityIdentifier( ChannelConst.E_CHANNEL );
}
@Override
protected ICreateMechanism createDatasetCreationMechanism()
{
    return new ListModelTableViewCreateMechanism( this )
    {
        @Override
        protected EntityProperty[] getEntityProperties() throws CoreException
        {
            Entity entity = getEntity();
            EntityProperty[] initValues = new EntityProperty[1];
            initValues[ 0 ] = EntityPropertyFactory.create( entity,
ChannelConst.FT_IDENTIFIER,
String.valueOf( System.currentTimeMillis() ) );
            return initValues;
        }
    };
}
}

```

7.6.7 Example EntityDetailModelTableView

In the example below we have a custom **detail model** table view for the *ArticleSales* sub-entity. There are following customization in the view:

- we customize the *ICreateMechanism* to ensure that all new created *ArticleSales* entries get the "Sales" classifier and the "Ean" field applies the value of the parent item "Ean" by default
- the *createSelectionPredicate()* is overridden to let the view only react to *item* selections (including multi-selections)
- we overridden the *handleEmptySelection()* method to set a custom message to the view content description area if no *item* is selected

This sample class is also contained in the SDK examples (since 7.1)

```

/**
 * A sample custom implementation of the {@link EntityDetailModelTableView} which
 * works with the 'ArticleSales'
 * sub-entity.
 * @author WMichel
 * @since 7.1
 */
public class CustomEntityDetailModelTableView extends EntityDetailModelTableView
{
    @Override
    protected ICreateMechanism createDatasetCreationMechanism()
    {

```

```

return new EntityDetailModelTableViewCreateMechanism( this )
{
    @Override
    protected EntityProperty[] getEntityProperties()
    {
        Entity entity = getEntity();
        LogicalKey logicalKey = entity.getLogicalKey(
"$ArticleTradingType.LK.Classifier" ); //$NON-NLS-1$
        EntityProperty[] initValues = new EntityProperty[2];
        initValues[ 0 ] = EntityPropertyFactory.create( logicalKey, "Sales" ); //$
$NON-NLS-1$
        try
        {
            EDataObject articleObject = getParentObject();
            Object articleEan = articleObject.get( "ean" ); //$NON-NLS-1$
            initValues[ 1 ] = EntityPropertyFactory.create( entity,
"$ArticleTradingType.Ean", articleEan ); //$NON-NLS-1$
        }
        catch ( CoreException e )
        {
            ErrorDialog.openError( getSite().getShell(), getPartName(),
e.getLocalizedMessage(), e.getStatus() );
        }
        return initValues;
    }
};
}
@Override
protected Predicate< ISelection > createSelectionPredicate()
{
    return new SelectionPredicateFactory.EntityPredicate( "Article", true ); //$NON-
NLS-1$
}
@Override
protected void handleEmptySelection()
{
    super.handleEmptySelection();
    StdViewDescriptionArea descriptionArea = ( StdViewDescriptionArea )
getTableContainer().getDescriptionArea();
    descriptionArea.setContentDescription( "No item selected" ); //$NON-NLS-1$
}

```

7.7 Creating custom perspectives

7.7.1 Creating perspectives

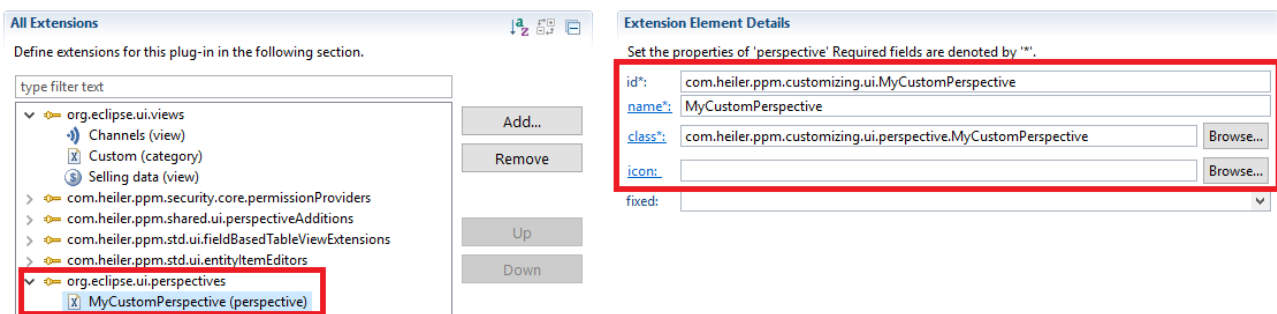
To create a custom perspective it is necessary to add a new java class to a custom plugin containing the new perspective with a perspective identifier.

Custom perspective class

```
public class MyCustomPerspective implements IPerspectiveFactory
{
    public static String PERSPECTIVE_ID =
    "hlr.custom.ui.perspective.MyCustomPerspective"; //$NON-NLS-1$

    @Override
    public void createInitialLayout( IPageLayout layout )
    {
        layout.setEditorAreaVisible( false );
    }
}
```

Now add a contribution to the `org.eclipse.ui.perspectives` extension point and fill the mandatory properties marked with a *.



Properties of "Extension Element Details"

key	value
id	PERSPECTIVE_ID of the created perspective class
name	Translatable name for the perspective. <div> <i>i</i> When starting the name with the % sign it is indicating that a variable in a properties file in the plugin directory is used. </div>
class	Java class reference which implements the <code>IPerspectiveFactory</code> .

key	value
icon (optional)	Relative path to the icon which will be used besides the name of the perspective.

An example custom perspective is provided in the SDK package "com.heiler.ppm.customizing.ui".

7.7.2 Adding views to a perspective

The screenshot shows the Eclipse IDE's 'All Extensions' and 'Extension Element Details' panels. In the 'All Extensions' panel, the 'com.heiler.ppm.customizing.ui.MyCustomPerspective (perspectiveExtension)' is selected. In the 'Extension Element Details' panel, the 'targetID' property is set to 'com.heiler.ppm.customizing.ui.MyCustomPerspective'.


To add a view to a perspective it is necessary to first add a contribution to the `org.eclipse.ui.perspectiveExtensions` extension point.

The `targetID` has to be the ID from the perspective where the views should be added.

After creating the `perspectiveExtension` a view can be added to it.

The screenshot shows the Eclipse IDE's 'All Extensions' and 'Extension Element Details' panels. In the 'All Extensions' panel, the 'com.heiler.ppm.customizing.ui.MyCustomPerspective (perspectiveExtension)' is selected. In the 'Extension Element Details' panel, the 'id' property is set to 'hrl.customizing.ui.view.customEntityDetailModelTableView'.

Properties of "Extension Element Details" of a view in a `perspectiveExtension`.

key	value
id	The ID of the view you want to add.
relationship	Defines how the current view is displayed to the view given at "relative" (stated below), for example "left", "right", "bottom", "stack"
relative	The view which is the reference for the relationship above
ratio	<p>The percentage ratio [0.01 -1] of space between the relative and the current view</p> <div>  When choosing relationship "stack", no ratio can be given </div>

An example custom perspective is provided in the SDK package " `com.heiler.ppm.customizing.ui` ".

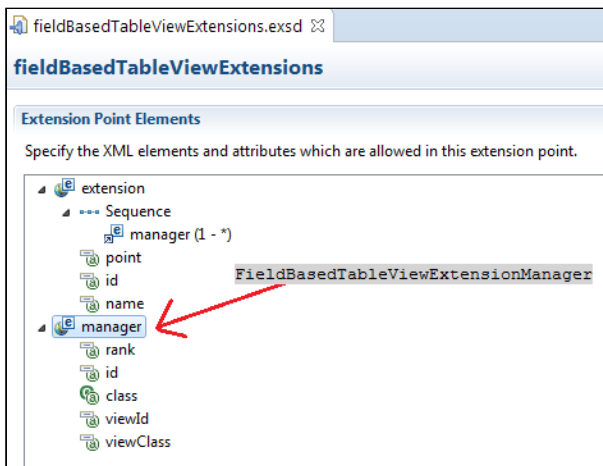
7.8 Customization of the field handling within views

7.8.1 Motivation

One of the long-term goals of the PIM development team is to eliminate the necessity to override the standard views in order to put some custom "business logic" to the existing views. Instead of this it should be possible to affect the standard views "from outside". As first step for this approach we created a new extension point which can be used to customize some *field based* logic of the standard views.

7.8.2 Extension point

The new extension point "*fieldBasedTableViewExtensions*" allows to influence the field handling of the standard views. For that reason the extension point provides the element **manager**:



You can contribute a *FieldBasedTableViewExtensionManager* for a view class or for a view ID. On the view creation the corresponding *FieldBasedTableViewExtensionManager* will be searched. First we search for a manager contribution for the view ID, if nothing found - for the view class. So if your *FieldBasedTableViewExtensionManager* is more specific for only one particular view, then define the view ID in the manager contribution. If the manager should be used for all derivations of a view - use the view class attribute instead. Additionally you can define the **rank** of your *FieldBasedTableViewExtensionManager*. If there are several *manager* contributed for the same view ID or for the same view class - the one with the highest **rank** will be used. All standard implementations of *FieldBasedTableViewExtensionManager* have the **rank** "1". So use a higher rank if you want, that your *manager* is used instead. Each manager contribution should define a unique id, so that it can be disabled by using the activities (see also [Possibility to disable standard UI elements](#) (see page 307)).

7.8.2.1 Availability

This extension point is available since PIM version 7.1.01.00 (see also the corresponding JIRA issue:

■ HPM-176

94 -

Projectreq
uest:
extension
point to

)

7.8.3 View extension manager implementation

In order to provide a possibility to change the field handling of the standard views, the *FieldBasedTableViewExtensionManager* provides a lot of methods which can be implemented in the custom *manager*. There are already some standard implementations of the *FieldBasedTableViewExtensionManager* for a number of views:



For all *FieldBasedTableViews* the default implementation of the manager is used: *FieldBasedTableViewExtensionManagerBaseImpl*. You can also extend the standard *manager* implementations and put your custom code. All the standard *FieldBasedTableViewExtensionManager* implementations are located in public packages, so you can use them in your customizings. An example of a custom implementation of *FieldBasedTableViewExtensionManager* you can find in SDK example plugins: `sdk\examples\customizing\com.heiler.ppm.customizing.ui` -->

`CustomAccumulatedAttributeTableViewExtensionManager`

7.8.3.1 TableCellValueProvider

By defining your own *TableCellValueProvider* you can affect the getting resp. setting of the field values in the table view as well as define whether a specific field value can be modified by the user. To provide your custom *TableCellValueProvider* implement the method `createTableCellValueProvider(TableCellValueProvider decoree)` in your *FieldBasedTableViewExtensionManager*. The default implementation of the just returns the `decoree` parameter without any modifications:

```
@Override
```

```

public TableCellValueProvider createTableCellValueProvider( TableCellValueProvider
decreee )
{
    // just return the given value provider
    return decreee;
}

```

7.8.3.2 CellEditor

Sometimes it is necessary to provide an other `CellEditor` for a specific field in the view then a standard `CellEditor`. Usually the standard `CellEditors` are created in a generic way by using the `FieldBasedCellEditorFactory` dependent on the field datatype. So if the corresponding *manager* returns `null` the standard `CellEditor` will be used :

```

@Override
public CellEditor getCellEditorForField( Composite parent, Field field )
{
    // return 'null' by default
    return null;
}

```

7.8.3.3 ICreateMechanism

If you want to change the *dataset creation* process of a view, then the method `getDatasetCreationMechanism()` of the *FieldBasedTableViewExtensionManager* is the right place for it. The default implementation of the manager returns `null`. If no other manager implementation for the view is registered, then the default creation mechanisms will be used: so the `ListModelTableViewCreateMechanism` for all list-model views and the `EntityDetailModelTableViewCreateMechanism` for all detail model views.

7.8.3.4 IDoubleClickListener

If the user clicks into an editable cell, the `CellEditor` will be activated, but - if the cell is not editable - we are able to add a double-click listener to the table to define, wath should happen. As default for all `FieldBasedTableViews` we provide a double-click listener which shows a simply dialog with the content of the clicked cell. This is usefull for all not-editable fields, since this allows the user to select and copy the field value (which is usually not possible within the table):

```

@Override
public IDoubleClickListener getDoubleClickHandler()
{
    if ( this.tableView instanceof FieldBasedTableView )
    {

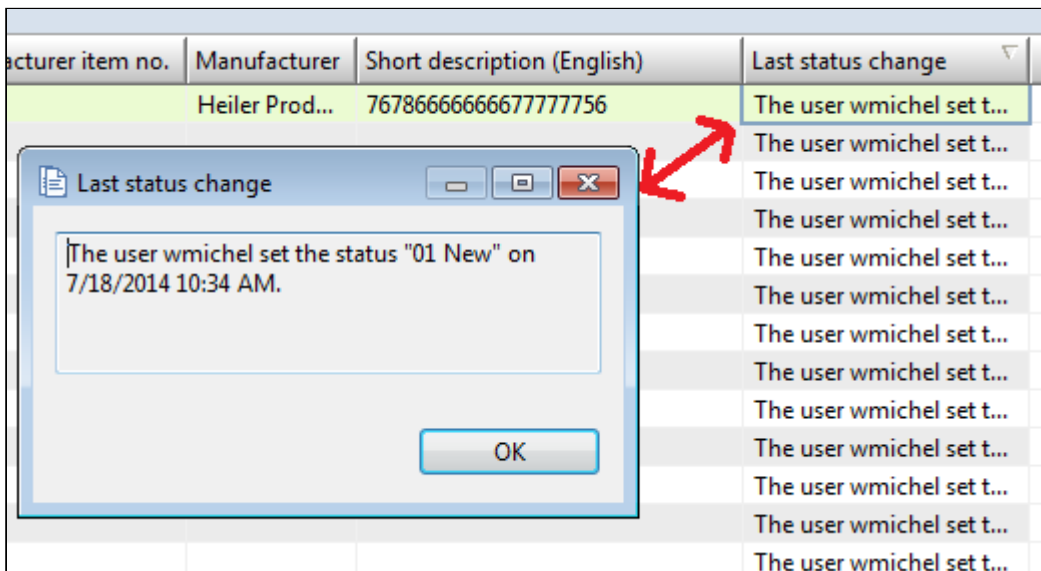
```

```

return new
DefaultFieldBasedTableViewDoubleClickListener( ( FieldBasedTableView ) this.tableVie
w );
}
return null;
}

```

Another advantage of this behavior: if the cell content is very large it is hard for the user to read it within the table. Now with a double-click the user can open a read-only dialog with a multi-line text field and can read the whole text:



7.9 Defining the default perspective

7.9.1 Summary

Before PIM 7.1.01 the default perspective of the PIM workbench was defined in the *plugin_customization.ini*. Now there is another possibility to determine which perspective is the *default* perspective.

The extension point *"com.heiler.ppm.shared.ui.perspectiveAdditions"* got a new element - *"defaultPerspectiveResolver"*. By contributing of this element you can affect which perspective will be shown in the workbench by default. In the standard PIM there is a default contribution of the resolver which provides the ID of the "Explorer (items)" perspective:

Extension Element Details

Set the properties of "defaultPerspectiveResolver". Required fields are denoted by "*".

rank*:

class*:

The "rank" attribute of the *defaultPerspectiveResolver* contribution determines which resolver is used. You can contribute many resolver, but only one with the **highest** rank will be used. The standard resolver has the rank "1". So if you contribute another resolver - use a higher rank for your contribution.

IMPORTANT: do not set the *defaultPerspectiveId* preference into the *plugin_customization.ini* if you want to use this extension point. The framework asks the preference first and, if it's set, this one will be used.

7.9.2 Example

A possible custom implementation of the *DefaultPerspectiveResolver* were a resolver which provides the default perspective ID dependent on the user group of the logged user. Here is an example how to implement such a one resolver.



The source code of the example implementation you can also find in the SDK example (*com.heiler.ppm.customizing.ui*).

This example implementation uses a simple mapping of the user group identifiers to the corresponding perspective IDs:

```
/**
 * The {@link DefaultPerspectiveResolver} implementation which resolves the ID of the
 * default perspective dependent on
 * the user group of the current user.
 * @author WMichel
 * @since 7.1.01.00
 */
public class UserGroupDefaultPerspectiveResolver implements
DefaultPerspectiveResolver
{
    public UserGroupDefaultPerspectiveResolver()
    {
    }
    @Override
    public String getDefaultPerspectiveId()
```

```

{
    LoginToken loginToken = LoginManager.getInstance()
        .getLoginToken();
    Group[] groups = loginToken.getGroups();
    // what about users who are in several user groups?
    for ( Group group : groups )
    {
        if ( group instanceof UserGroupPrincipal )
        {
            UserGroupPrincipal groupPrincipal = ( UserGroupPrincipal ) group;
            String identifier = groupPrincipal.getIdentifier();
            // in this sample implementation we have just a mapping of
            "UserGroupIdentifier" to the corresponding "perspectiveId" in the resource bundle
            // but this is not really a smart solution ;)
            // in you customizing you can certainly implement another kind of mapping
            return Messages.getString( identifier );
        }
    }
    return null;
}
}

```

Then the resolver should be contributed with a higher rank as the standard one:

```

<extension
    point="com.heiler.ppm.shared.ui.perspectiveAdditions">
    <defaultPerspectiveResolver
        class="com.heiler.ppm.customizing.ui.perspective.UserGroupDefaultPerspectiveResolver"
        rank="2">
    </defaultPerspectiveResolver>
</extension>

```

7.10 Entity specific icons

For a couple of repository entities there is a set of icons which can be obtained dynamically in PIM code.

Here is an example how to get an icon for a specific entity:

```

// this code provides a small image (16x16px) for the entity "Article"
String entityIdentifier = ArticleCoreConst.ENTITY_ARTICLE;
Image image = RepositoryUiUtils.getImage( entityIdentifier,
    ImageType.ICON_SMALL );

```

There are three several sizes of the entity images:



















- ImageTypes#ICON_SMALL - 16x16px
- ImageTypes#ICON_MEDIUM - 32x32px

























- `ImageTypes#ICON_LARGE` - 48x48px
















An additional image type - `ImageTypes#DIALOG` - can be used to get an appropriate image for the title area of an `TitleAreaDialog`. In this case the large variant of the corresponding image will be used and scaled to the size of the title area.

If there is no icon for the requested entity - the `default` icon will be returned (see also the table below).

All entity icon files are located in the plugin `com.heiler.ppm.configuration.server` -> folder `conf/icons`. At the moment there are following entity icons available:

Entity	Small	Medium	Large
Article			
ArticleAssortment			
Assortment			
EntityReportQuery			
MasterCatalog			
MediaAsset			

Entity	Small	Medium	Large
PriceGroup			
Product			
Product2G			
ProductAssortment			
Revision			
Structure			
StructureGroup			
SupplierCatalog			

Entity	Small	Medium	Large
Task			
Unit			
User			
Variant			
default (as fallback)			

7.11 Extension point for SearchView customization

Using the new extension point **com.heiler.ppm.search.ui.searchViewAdditions** you can customize certain *SearchViews* without subclassing them.

7.11.1 Predefined search parameters

Sometimes it is useful to add additionally search parameters to each search which is triggered in a *SearchView*. This "predefined" search parameters are invisible for the user and will be ("and-") conjuncted with other search parameters defined by the user.

There are two ways to define *predefined* search parameters:

1. **via extension:**


```

<extension
    point="com.heiler.ppm.search.ui.searchViewAdditions">
    <searchView
        searchViewId="com.heiler.ppm.article.ui.views.ArticleSearchView">
        <searchParameter
            fieldIdentifier="ArticleLang.DescriptionShort">
            <searchValue
                caseSensitive="false"
                operator="contains"
                value="hammer">
            </searchValue>
            <logicalKey
                identifier="ArticleLangType.LK.Language"
                value="7">
            </logicalKey>
        </searchParameter>
    </searchView>
</extension>

```

2. via class:

```

public class ArticlePredefinedSearchParameters implements
PredefinedSearchParameters
{
    @Override
    public Expression getPredefinedSearchExpression()
    {
        RepositoryService repositoryService =
RepositoryComponent.getRepositoryService();
        Field field = repositoryService.getFieldByIdentifier(
"$NON-NLS-1$"ArticleLang.DescriptionShort" ); //$NON-NLS-1$
        FieldPath fieldPath = RepositoryUtils.getFieldPath( field );
        fieldPath.setData( SearchView.KEY_INITIAL_IDENTIFIER,
field.getInitialIdentifier() );
        fieldPath.setData( SearchView.KEY_IDENTIFIER, field.getIdentifer() );
        fieldPath.setData( SearchView.KEY_NAME, field.getName() );
        fieldPath.setData( SearchView.KEY_NAME_FROM_TOP, field.getNameFromTop() );
        fieldPath.getEntityPath()
            .setLogicalKeyValue( "$NON-NLS-1$"ArticleLangType.LK.Language", 9 ); //$NON-NLS-1$
        FieldPathExpression fieldPathExpression = new
FieldPathExpression( fieldPath );
        ValueExpression valueExpression = new ValueExpression( "hammer" ); //$NON-NLS-1$
        Expression operation = fieldPathExpression.contains( valueExpression );
        return operation;
    }
}

```

```

<extension
    point="com.heiler.ppm.search.ui.searchViewAdditions">
    <searchView

predefinedSearchParameters="com.heiler.ppm.custom.article.ui.internal.ArticlePr
edefinedSearchParameters"
        searchViewId="com.heiler.ppm.article.ui.views.ArticleSearchView">
    </searchView>
</extension>

```

In both cases, each search which is performed in the *ArticleSearchView* will be "erniched" with the additional search parameter for the field "Short description".

You can also use both ways together. In this case all predefined search parameters will be and-conjoined.

7.11.2 Different result view

Generally the result of the search will be shown in the *default* view for the root search entity. The root search entity is defined in the search view contribution as an initialization parameter (e.g. *com.heiler.ppm.articlesearch.ui.internal.view.ArticleSearchView:Article*). The *default* view for the root entity will be obtained from the repository. Additionally you can define a different result view which should be used to show the search result. Just use the same extension point and specify the result view ID:

```

<extension
    point="com.heiler.ppm.search.ui.searchViewAdditions">
    <searchView
        resultViewId="com.heiler.ppm.custom.article.ui.CustomArticleView"
        searchViewId="com.heiler.ppm.article.ui.views.ArticleSearchView">
    </searchView>
</extension>

```

In this case the result of the search performed in the *ArticleSearchView* will be always shown in the *CustomArticleView*.

7.11.3 Download

You can find a full source code example for this here.

7.12 HOWTOs

- [How to center an image in a table cell \(see page 296\)](#)
- [How to contribute your own status line in the table view \(see page 296\)](#)
- [How to customize the status text in the table view \(see page 297\)](#)
- [How to add a custom menu item to the text celleditor context menu \(see page 297\)](#)
- [How to contribute a context menu item for a specific Entity \(see page 299\)](#)
- [How to create a dynamic menu contribution item \(see page 299\)](#)

- [How to create/override a key binding for a command contribution \(see page 301\)](#)
- [How to add a permission to popupmenu action \(see page 302\)](#)

7.12.1 How to center an image in a table cell

If you have a table column with images, it is sometimes makes sense to center the images. Assuming you work with a subclass of *StdTableView*. Just override the method *centerImageInCell* and return *true* if the specified image is an image you want to center or if the given *colIndex* is in the column with images you want center. Here is an example:

```
@Override
protected boolean centerImageInCell( Image image, int colIndex )
{
    if ( getTableContainer() instanceof FieldBasedTableContainer )
    {
        // if you use a field-based view, just compare the given colIndex with the
        // index of the image column
        FieldBasedTableContainer container = ( FieldBasedTableContainer )
getTableContainer();
        int imageColumnIndex = container.getColumnIndex( ANONYM_LOGGEDIN_IDENTIFIER );
        return ObjectUtils.equals( colIndex, imageColumnIndex +
StdTableContainer.RESERVED_COLUMNS );
    }
    // otherwise, you can compare the given image with the images which should be
    // centered
    StdAbstractUIPlugin plugin = UserManagementUiPlugin.getDefault();
    Image active = plugin.getImage( ResourceConst.IMG_ACTIVE );
    return ObjectUtils.equals( active, image );
}
```

As result your image will be centered in the table cell:

	User name	Password	Active	logged in	Given name	Name	Alias
1	Administrator	*****	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		Administrator	admin
2	wmichel	*****	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Willy	Michel	wmichel
3	Guest	*****	<input checked="" type="checkbox"/>				

3 users_logged_in_small.png

7.12.2 How to contribute your own status line in the table view

If you want to implement your own status line component in the table view, just contribute the extension point *com.heiler.ppm.std.ui.tableViewStatusLines*. The default contribution of the status line component should be commented then. Here is the default contribution as example:

```
<extension point="com.heiler.ppm.std.ui.tableViewStatusLines">
    <statusLine
        id="com.heiler.ppm.std.ui.table.statusline.defaultTableViewStatusLine"
        class="com.heiler.ppm.std.ui.table.statusline.DefaultTableViewStatusLine">
```

```
</statusLine>
</extension>
```

The status line component should contain a status text control (usually a label), to show the status text in the table view. Additionally you can add other controls to the status line component. The status line component will be added to the footer area of the table view.

7.12.3 How to customize the status text in the table view

If you want to show your own text in the table view's status line, just contribute the extension point *com.heiler.ppm.std.ui.statusTextResolvers*. The default contribution of the status text resolver should be commented then. Here is the default contribution as example:

```
<extension point="com.heiler.ppm.std.ui.statusTextResolvers">
  <resolver
    id="com.heiler.ppm.std.ui.table.statusline.defaultStatusTextResolver"
    class="com.heiler.ppm.std.ui.table.statusline.DefaultStatusTextResolver">
  </resolver>
</extension>
```

Your own *StatusTextResolver* will be called each time the selection of the table view has been changed. As parameter your resolver receives a *tableViewer* and should return a suitable status text for the current table state. The default resolver just shows the number of selected items in the table:

	Item no.	EAN	Manufacturer item num...	Manufacturer	Short description (German)
1	2115412	4010314015276	00033712	GEYER AG	Montageplatte Zb
2	2056868	4010314016013	00047433	GEYER AG	Montageplatte 600X276mm
3	2056874	4010314016020	00047434	GEYER AG	Montageplatte 800X276mm
4	2056878	4010314016037	00047435	GEYER AG	Montageplatte 850X497,5mm
5	2056882	4010314016044	00047436	GEYER AG	Montageplatte 1000X500mm
6	2691513	4015080007678	000767	Moeller GmbH	Stahlblech-Gehäuse HxBxT=2
7	2654824	4010314145034	00079268	GEYER AG	WAGO Draht-Kettenbrücke 31
8	2100000	40150800010160	001016	Moeller GmbH	Umhaute f. K05/ + K150/

5 items selected

4 table_view_status_line_text.png

7.12.4 How to add a custom menu item to the text celleditor context menu

Since we have our own context menu for the text celleditor, it is possible to append a custom menu item there. Let us assume, we would add a new menu item "To Upper Case" on the context menu. When the user clicks this menu item, the selected text in the celleditor will be converted to uppercase.

To do this we need the following:

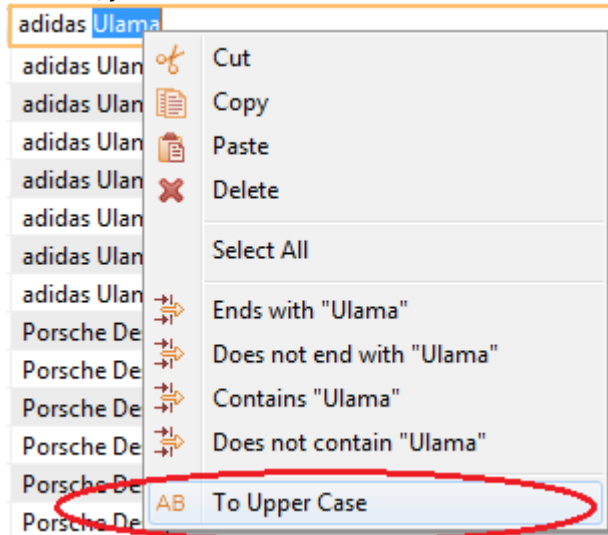
1. Define a command, which will be executed, when the menu item is clicked:

```
<command
    defaultHandler="com.heiler.ppm.std.ui.example.ToUpperCaseHandler"
    id="com.heiler.ppm.std.ui.example.toUpperCase"
    name="To Upper Case">
</command>
```

2. Implement the default handler for this command, which converts the selected text in the celleditor to uppecase (see *com.heiler.ppm.std.ui.example.ToUpperCaseHandler* for implementation details).
3. Now you should add a contribution for the extension point *org.eclipse.ui.menus*:

```
<extension
    point="org.eclipse.ui.menus">
    <menuContribution
        allPopups="false"
        locationURI="popup:com.heiler.ppm.std.ui.celleditors">
        <separator
            visible="true"
            name="edit">
        </separator>
        <command
            commandId="com.heiler.ppm.std.ui.example.toUpperCase"
            style="push"
            label="%menu.toUpperCase.label"
            icon="icons/UpperCase.png">
        </command>
    </menuContribution>
</extension>
```

As result, your custom menu item is added to the celleditor:



Note: if you enable the native context menu for the text celleditors (in client *plugin_customization.ini*), your custom menu item will be not appear.

7.12.5 How to contribute a context menu item for a specific Entity

If you want to add a context menu item for several views, which show entries of a specific entity, you can use the corresponding *PropertyTester*.

- Create a menu contribution with the *locationURI* "**popup:org.eclipse.ui.popup.any**"
- Add a *visibleWhen* condition to the menu contribution with a property tester which tests, if the selected entries are of a specific entity. This should prevent that the menu item appears in all views regardless of the showing entity.

```
<extension
  point="org.eclipse.ui.menus">
  <menuContribution
    allPopups="false"
    locationURI="popup:org.eclipse.ui.popup.any?after=edit-ext">
    <command
      commandId="com.heiler.ppm.assortment.ui.command.cloneAssortment"
      id="com.heiler.ppm.product2g.assortment.ui.cloneProductAssortment"
      style="push"
      icon="icons/16x16/Copy.gif"
      label="%command.CloneProductAssortment.label"
      tooltip="%command.CloneProductAssortment.tooltip">
      <visibleWhen
        checkEnabled="false">
        <with
          variable="selection">
          <test
            property="com.heiler.ppm.std.ui.adapter.entity"
            value="Product2GAssortment">
          </test>
        </with>
      </visibleWhen>
    </command>
  </menuContribution>
</extension>
```

In this example the contributed menu item will be only added to views which show the product assortments.

7.12.6 How to create a dynamic menu contribution item

Sometimes it is necessary to create a menu item with dynamic sub-menus, which are determined and built at runtime in a Java class. For this you need first your own implementation of class *CompoundContributionItem*:

VERY IMPORTANT: don't override the constructor with the parameter 'id', otherwise the contribution item will be NOT appear in the menu.

```

public class ExampleDynamicContributionItem extends CompoundContributionItem
{
    @Override
    protected IContributionItem[] getContributionItems()
    {
        MenuManager menu = new MenuManager( "Example menu", getId() ); //$NON-NLS-1$

        IContributionItem subMenu1 = createExampleContributionItem( "Submenu 1" ); //$NON-NLS-1$
        menu.add( subMenu1 );
        IContributionItem subMenu2 = createExampleContributionItem( "Submenu 2" ); //$NON-NLS-1$
        menu.add( subMenu2 );
        IContributionItem subMenu3 = createExampleContributionItem( "Submenu 3" ); //$NON-NLS-1$
        menu.add( subMenu3 );

        return new IContributionItem[] { menu };
    }

    /** Creates a new example command contribution item with the given label. */
    private IContributionItem createExampleContributionItem( String label )
    {
        String itemId = hashCode() + "." + label; //$NON-NLS-1$
        String commandId = ExampleHandler.COMMAND_ID;
        CommandContributionItemParameter parameter = new
        CommandContributionItemParameter(

StdUiUtils.getActiveSite(),

itemId,

commandId,

CommandContributionItem.STYLE_PUSH );
        parameter.label = label;
        CommandContributionItem item = new CommandContributionItem( parameter );
        return item;
    }
}

```

Now you only need to contribute your dynamic menu item:

```

<menuContribution
    locationURI="menu:com.heiler.ppm.example.ui.views.ExampleView">
    <dynamic
        id="com.heiler.ppm.example.ui.handler.example"
        class="com.heiler.ppm.example.ui.handler.ExampleDynamicContributionItem">
    </dynamic>
</menuContribution>

```

7.12.7 How to create/override a key binding for a command contribution

If you want to bind a command contribution on a key sequence, you just add a contribution for the extension point "org.eclipse.ui.bindings". But if the key sequence is already bound on an other command?

You can override the key binding for the current context using "contextId" parameter of the binding contribution.

Here is an example for the "delete" key. The key sequence "DEL" is bound in HPM main plugin on the command "org.eclipse.ui.edit.delete". So if you want to call a specific command in a specific view, when the user press the "delete" key, you should "override" the key binding for the view context.

Therefore you need to define a context for the view:

```
<extension point="org.eclipse.ui.contexts">
  <context
    parentId="org.eclipse.ui.contexts.window"
    name="StructureFeatureTableView context"

    id="com.heiler.ppm.structurefeature.ui.internal.view.StructureFeatureTableView">
</extension>
```

Then in the view class we activate this context in the context service:

```
public class StructureFeatureTableView extends ListModelTableView implements
  IEntityChangeListener
{
  public static final String VIEW_ID =
  "com.heiler.ppm.structurefeature.ui.internal.view.StructureFeatureTableView"; //$NON-
  NLS-1$
  ...

  @Override
  public void createPartControl( final Composite parent )
  {
    ...

    // set the view context to support the key binding for 'delete' command
    contribution
    IContextService contextService = ( IContextService )
    getSite().getService( IContextService.class );
    contextService.activateContext( VIEW_ID );
  }
}
```

Now you can bind the "delete" key on your command with the specified context:

```
<extension
  point="org.eclipse.ui.bindings">
```



```

    <key
      commandId="com.heiler.ppm.structurefeature.ui.handler.deleteStructureFeature"
    contextId="com.heiler.ppm.structurefeature.ui.internal.view.StructureFeatureTableView
    "
      schemeId="com.heiler.ppm.main.defaultAcceleratorConfiguration"
      sequence="DEL">
    </key>
  </extension>

```

7.12.8 How to add a permission to popupmenu action

Note: Actions were used before commands exist. Please use commands instead of actions.

If you need to add a permission for the visibility of an action you need to add following at the plugin.xml

```

<extension
  point="org.eclipse.ui.popupMenus">
  <viewerContribution
    id="ArticleTableView.PopupMenu"
    targetID="com.heiler.ppm.article.ui.views.ArticleTableView">
    <visibility>
      <objectState
        name="permissions" value="com.heiler.ppm.structure.core.permission.StructureGroupRead">
      </objectState>
    </visibility>
    <action .... />
  </viewerContribution>
</extension>

```

7.13 ListModelTableViews: selection behavior

7.13.1 Problem definition

The input for a `ListModelTableView` is always a `ListModel` which contains `ListEntries`. So each table row encapsulates a `ListEntry` as data object. If the user selects one or many rows in a table a selection will be notified. This selection contains all data objects of the selected rows. So in case of `ListModelTableViews` this data objects are `ListEntries`. The problem of this behavior is, that there are several selection listeners which receive selection events and extract the containing elements from the selection. Some of these listeners may hold this elements for internal use. If a `ListEntry` is referenced in this way by any listener, the whole `ListModel` is also referenced there (because any `ListEntry` has a reference to the parent `ListModel`). This behavior may cause performance issues when the user e.g. loads very huge catalogs (with ~ 1 million items). The reason is that a loaded `ListModel` allocates a lot of memory, especially if the grouping tree is enabled - in this case all grouping columns are loaded completely.

So if the user loads another catalog or the same catalog again the "old" `ListModel` can't be removed from the memory so long as any references to the `ListModel` (or its `ListEntries`) are existing. The new loaded `ListModel` in turn allocates a lot of memory too. So at this point it can lead to an "OutOfMemory" error. To solve such issues the providing of the selection in all `ListModelTableViews` has been changed.

7.13.2 Solution

Briefly speaking: a `ListModelTableView` doesn't fire `ListEntries` as a selection anymore. Instead of that the corresponding `EntityProxies` will be extracted and notified as a selection. To ensure that, the selection provider of the `ListModelTableView` was customized. Each time a selection of the view is requested - the `EntityProxies` of the currently selected entries will be extracted and provided. In this way no `ListEntries` anymore will be referenced outside of the view. In most cases the listeners of the selection event need anyway only the proxies of the selected entries. So now they get these proxies directly and should not extract them itself from the `ListEntries`.

7.13.3 Adverse impacts

Unfortunately many receivers of a `ListModelTableView` selection assume that the selection contains `ListEntries` only and try to cast the selection elements to `ListEntries`, which can lead to a "ClassCastException" now. In the standard PIM code all such locations were adjusted to be able to handle also selections which contain `EntityProxies` instead of `ListEntries`. Luckily both classes - `ListEntry` and `EntityProxy` - have a common interface - `EntityItem`. So to be sure that no "ClassCastException" will be thrown, it is **recommended** to cast the selection elements to `EntityItem` which can be both: a `ListEntry` or an `EntityProxy`. To get the `EntityProxy` from an `EntityItem` just call the method `getEntityProxy()`.

7.13.4 Examples

7.13.4.1 Command handler

```
public class CustomCommandHandler extends AbstractHandler
{
    @Override
    public Object execute( ExecutionEvent event ) throws ExecutionException
    {
        ISelection selection = HandlerUtil.getCurrentSelection( event );
        // if you need only the first selected entry:
```

```

    EntityProxy selectedElement = StdUiUtils.getFirstSelectedElement( selection,
EntityProxy.class );
    if ( selectedElement != null )
    {
        // handle selected proxy
        ...
    }

    // if you need all selected entries:
    EntityProxy[] selectedElements = StdUiUtils.getSelectedElements( selection,
EntityProxy.class );
    if ( selectedElements.length > 0 )
    {
        // handle selected proxies
        ...
    }
}
}

```

7.13.4.2 Custom views

```

public class CustomTableView extends StdTableView
{
    @Override
    public void selectionChanged( IWorkbenchPart selectionPart, ISelection selection )
    {
        Predicate< ISelection > selectionPredicate = getSelectionPredicate();
        if ( selectionPredicate.evaluate( selection ) )
        {
            EntityProxy selectedElement = StdUiUtils.getFirstSelectedElement( selection,
EntityProxy.class );
            if ( selectedElement != null )
            {
                // set/update viewer input for the current selected entry
                ...
            }
            this.lastReceivedSelectionPart = selectionPart;
        }
        else if ( selectionPart.equals( this.lastReceivedSelectionPart ) )
        {
            if ( selection.isEmpty() )
            {
                handleEmptySelection();
            }
            else
            {
                handleMultiSelection( selection );
            }
        }
    }
}

```

```
}
```

7.13.5 Don'ts

Never cast the elements of a selection to `ListEntries` without checking if they are really

`ListEntries`:

```
public void anyMethodWhichGetsAnySelection ( ISelection selection )
{
    IStructuredSelection structuredSelection = ( IStructuredSelection ) selection;
    ListEntry listEntry = ( ListEntry ) structuredSelection.getFirstElement(); // <-
    this is a potential cause of a "ClassCastException"
}
```

Alternatively use `StdUiUtils` to extract any objects you expect from the given selection (like in examples above). The corresponding methods of `StdUiUtils` are proof against `NullPointerException` - or `ClassCastException`.

7.14 Long-running processes

This site shows how to handle the long-running processes in the UI.

Basically, it should be avoided to execute long-running processes in the UI. If it is still necessary, you have to ensure that these processes do not run on the UI-thread. This article illustrates the possibilities one has to deal with it.

7.14.1 Rule number one: Do not block the UI thread!

If you write code which will be used by any UI components. You must always take care that the code does not include calls that can take a long time

```
1  protected void someMethodInUI()
2  {
3      final String label = someBusinessObject.invokeVeryLongOperation();
4      someUiObject.setText( label );
5  }
```

The problem in the line 3 is: you don't know how much time takes this call. And so long as the call lasts, the UI thread is waiting. In other words: the UI hangs.

7.14.2 Best Practice

The best practice to avoid this issue is to run the long-running processes within an eclipse job:

```

RunnableWithProgress runnable = new RunnableWithProgressBaseImpl()
{
    @Override
    protected void doRunWithCoreException( IProgressMonitor progressMonitor ) throws
    CoreException, InterruptedException
    {
        invokeVeryLongOperation();
    }
};
IProgressService progressService = PlatformUI.getWorkbench()
                                           .getProgressService();
Job job = new RunnableWithProgressJob( jobName, runnable );
job.schedule();
progressService.showInDialog( shell, job );

```

Important is to schedule the job **before** showing it in a dialog via the progress service. In this case the progress service waits for a short duration before opening the progress dialog. Otherwise the dialog appears **always**, also if the operation lasts only few miliseconds.

The advantage of using jobs is the possibility to add a *JobListener* to the job to be notified when the job is done:

```

job.addJobChangeListener( new JobChangeAdapter()
{
    @Override
    public void done( IJobChangeEvent event )
    {
        // IMPORTANT: access the UI stuff only within the UI thread!!!
        Display.getDefault()
            .asyncExec( new Runnable()
            {
                @Override
                public void run()
                {
                    updateUI();
                }
            } );
    }
} );

```

If you don't want to show the progress dialog, you can also use an other function of the progress service: showing a busy cursor while an operation is running:

```

RunnableWithProgress runnable = new RunnableWithProgressBaseImpl()
{
    @Override
    protected void doRunWithCoreException( IProgressMonitor progressMonitor ) throws
    CoreException, InterruptedException
    {

```

```

        invokeVeryLongOperation();
    }
};
IProgressService progressService = PlatformUI.getWorkbench()
                                           .getProgressService();
progressService.busyCursorWhile( new RunnableWithProgressWrapper( runnable ) );

```

Disadvantages of this method:

- the user is forced to wait until the operation is finished
- no listener registration is possible (e.g. to be notified when the operation is done)

7.15 Possibility to disable standard UI elements

- [Activities](#) (see page 307)
- [General information](#) (see page 307)
- [Views](#) (see page 307)
- [Perspectives](#) (see page 307)
- [Menu items](#) (see page 309)
 - [menuContribution -> command](#) (see page 309)
 - [menuContribution -> dynamic](#) (see page 310)
 - [menuContribution -> menu](#) (see page 310)
 - [viewContribution -> action](#) (see page 311)
 - [objectContribution -> action](#) (see page 312)
- [Preference pages](#) (see page 312)
- [Drop listeners](#) (see page 313)
- [Status lines](#) (see page 314)
- [Status text resolvers](#) (see page 314)
- [FieldBasedTableViewExtensionManager](#) (see page 315)
- [Complete deletion of action sets](#) (see page 315)
- [Download](#) (see page 316)

Using Eclipse Activities it is possible to disable certain UI elements such views, menu items etc. in the standard Product Manager. The following explains how the various UI elements can be disabled.

7.15.1 Activities

Eclipse Activities can be used to hide/display certain UI elements. We are already using activities in the standard Product Manager for example to disable unnecessary views, menu items and preference pages of the eclipse platform.

To learn more about Eclipse Activities, read following sources:

- [Help - Eclipse Platform](#)
- [FAQ How do I add activities to my plug-in?](#)
- [Eclipse Activities – Hide / Display certain UI elements](#)

7.15.2 General information

You can use **activities** to disable the most of the UI elements, assumed the corresponding UI contribution has the *ID* attribute. The element to disable should be clearly recognizable. This means that the *ID* of the UI contribution should be unique in the whole plugin. If there are more than one UI element with the same *ID*, all elements with this *ID* in the plugin will be disabled. The *pattern* attribute of the **activityPatternBinding** consists of the *pluginID* + *contributionID*, where the *contributionID* is the *ID* of the UI contribution to disable and the *pluginID* is the *ID* of the plugin where the UI element is contributed.

7.15.3 Views

To disable (hide) a specific standard view just add a new **activities** extension to your custom plugin and bind this activity with the view:

```
<extension
  point="org.eclipse.ui.activities">
  <activity
    id="com.heiler.ppm.custom.activities.hide.views"
    name="Deactivate views">
  </activity>
  <activityPatternBinding
    activityId="com.heiler.ppm.custom.activities.hide.views"
    pattern="com.heiler.ppm.article.ui/
com.heiler.ppm.article.ui.views.ArticleTableView">
  </activityPatternBinding>
</extension>
```

NOTE:

- the disabled view does not appear in the "Open view..." dialog, but if you have any actions which open this view programmatically, you have to disable these actions too. See below how to disable menu items.
- the disabled view is not removed from the client workspace, so if the view was already shown in the client before disabling, it will be still shown. In this case the old workspace should be deleted to ensure that the disabled view does not appear anymore.



Since PIM version 7.1.08 there is an alternative possibility to hide views and perspectives. See also the corresponding article: Frontend visibility

7.15.4 Perspectives

For disabling a whole perspective we have since the version 5.2 an extension point **com.heiler.ppm.shared.ui.perspectiveAdditions**. Please use this extension point for perspectives instead of **activities**.

```
<extension
    point="com.heiler.ppm.shared.ui.perspectiveAdditions">
    <disablement
        perspective-
id="com.heiler.ppm.structure.ui.perspectives.StructureEditorPerspective">
    </disablement>
</extension>
```

7.15.5 Menu items

With **activities** you can't disable *commands* or *actions* itself, but the menu contributions for this *commands* and *actions*:



In the newer Eclipse versions there are known issues hiding menu items. See <https://wiki.eclipse.org/Eclipse4/KnownIssues/4.1>
So there can be issue with "item is visible or enabled when it should not be". Please take that into account implementing your customization.

7.15.5.1 menuContribution -> command

```
<extension
    point="org.eclipse.ui.menus">
    <menuContribution
        allPopups="false"

locationURI="popup:com.heiler.ppm.article.ui.views.attribute.AccumulatedAttributeTabl
eView?after=additions">
        <command

commandId="com.heiler.ppm.article.ui.command.attribute.deleteAcumulatedArticleAttribu
te"
            icon="icons/Delete.gif"

id="com.heiler.ppm.article.ui.menu.attribute.deleteAcumulatedArticleAttribute"
            label="%action.deleteAttributes.name">
            ...
        </command>
        ...
    </menuContribution>
</extension>
...
<extension
    point="org.eclipse.ui.activities">
    <activity
```



```

        id="com.heiler.ppm.custom.activities.hide.commands"
        name="Deactivate commands">
    </activity>
    <activityPatternBinding
        activityId="com.heiler.ppm.custom.activities.hide.commands"
        pattern="com.heiler.ppm.article.ui/
com.heiler.ppm.article.ui.menu.attribute.deleteAcumulatedArticleAttribute">
    </activityPatternBinding>
</extension>

```

7.15.5.2 menuContribution -> dynamic

```

<extension
    point="org.eclipse.ui.menus">
    <menuContribution
        allPopups="false"
        locationURI="popup:com.heiler.ppm.revision.ui.versioning.Submenu?
before=additions">
        <dynamic
            class="com.heiler.ppm.article.ui.internal.command.VersionArticlesOfStruct
ureGroupContributionItem"

            id="com.heiler.ppm.article.ui.internal.command.VersionArticlesOfStructureGroupContrib
utionItem">
                ...
            </dynamic>
        </menuContribution>
    </extension>
    ...
<extension
    point="org.eclipse.ui.activities">
    <activity
        id="com.heiler.ppm.custom.activities.hide.commands"
        name="Deactivate commands">
    </activity>
    <activityPatternBinding
        activityId="com.heiler.ppm.custom.activities.hide.commands"
        pattern="com.heiler.ppm.article.ui/
com.heiler.ppm.article.ui.internal.command.VersionArticlesOfStructureGroupContributio
nItem">
    </activityPatternBinding>
</extension>

```

7.15.5.3 menuContribution -> menu

```

<extension
    point="org.eclipse.ui.menus">

```

```

        <menuContribution
            allPopups="false"

locationURI="popup:com.heiler.ppm.article.ui.views.attribute.AccumulatedAttributeTable
eView?after=additions">
    <menu
        icon="icons/Delete.gif"

id="com.heiler.ppm.article.ui.views.attribute.AccumulatedAttributeTableView.deleteAtt
ributeMaps"
        label="%action.deleteAttributeMaps.name">
            ...
    </menu>
</menuContribution>
</extension>
...
<extension
    point="org.eclipse.ui.activities">
    <activity
        id="com.heiler.ppm.custom.activities.hide.commands"
        name="Deactivate commands">
    </activity>
    <activityPatternBinding
        activityId="com.heiler.ppm.custom.activities.hide.commands"
        pattern="com.heiler.ppm.article.ui/
com.heiler.ppm.article.ui.views.attribute.AccumulatedAttributeTableView.deleteAttribu
teMaps">
    </activityPatternBinding>
</extension>

```

7.15.5.4 viewContribution -> action

```

<extension
    point="org.eclipse.ui.viewActions">
    <viewContribution

targetID="com.heiler.ppm.structure.ui.internal.views.structure.StructureView"
        id="com.heiler.ppm.article.ui.structureContribution">
        <action
            class="com.heiler.ppm.article.ui.internal.contribution.NotEmptyGroupsFilt
erAction"
            icon="icons/StructureGroupFilter.gif"

id="com.heiler.ppm.article.ui.structurecontribution.NotEmptyGroupsFilterAction"
            ...
        </action>
    </viewContribution>
</extension>
...
<extension

```

```

        point="org.eclipse.ui.activities">
<activity
    id="com.heiler.ppm.custom.activities.hide.actions"
    name="Deactivate actions">
</activity>
<activityPatternBinding
    activityId="com.heiler.ppm.custom.activities.hide.actions"
    pattern="com.heiler.ppm.article.ui/
com.heiler.ppm.article.ui.structurecontribution.NotEmptyGroupsFilterAction">
</activityPatternBinding>
</extension>

```

7.15.5.5 objectContribution -> action

```

<extension
    point="org.eclipse.ui.popupMenus">
    <objectContribution
        adaptable="true"
        id="com.heiler.ppm.article.ui.actions.StructurePopupMenu"

objectClass="com.heiler.ppm.structure.core.internal.wrapper.StructureGroupWrapper">
        <action
            class="com.heiler.ppm.article.ui.internal.popup.StructurePopupMenuShowNotClassified"
            icon="icons/ShowItemsNotClassified.gif"
            id="com.heiler.ppm.article.ui.actions.StructurePopupMenu.submenu"
            ...
        </action>
    </viewContribution>
</extension>
...
<extension
    point="org.eclipse.ui.activities">
    <activity
        id="com.heiler.ppm.custom.activities.hide.actions"
        name="Deactivate actions">
    </activity>
    <activityPatternBinding
        activityId="com.heiler.ppm.custom.activities.hide.actions"
        pattern="com.heiler.ppm.article.ui/
com.heiler.ppm.article.ui.actions.StructurePopupMenu.submenu">
    </activityPatternBinding>
</extension>

```

7.15.6 Preference pages

To disable (hide) a specific preference page just add a new **activities** extension to you custom plugin and bind this activity with the preference page:

```

<extension
  point="org.eclipse.ui.activities">
  <activity
    id="com.heiler.ppm.custom.activities.hide.preferencePages"
    name="Deactivate preferencePages">
  </activity>
  <activityPatternBinding
    activityId="com.heiler.ppm.custom.activities.hide.preferencePages"
    pattern="com.heiler.ppm.article.ui/
com.heiler.ppm.article.ui.preference.PreferencePageAttributes">
  </activityPatternBinding>
</extension>

```

7.15.7 Drop listeners

Using **activities** you can also disable UI elements which are contributed for our own extension points, for example **dropListeners** or **drop-choices**:

```

<extension
  point="com.heiler.ppm.std.ui.dropListeners">
  <dropListener
    targetClass="org.eclipse.emf.ecore.sdo.EDataObject"

    id="com.heiler.ppm.article.ui.internal.transfer.attribute.ArticleAttributeStructureGroupAttributeDropListener"
    ...
  </extension>
  ...
<extension
  point="org.eclipse.ui.activities">
  <activity
    id="com.heiler.ppm.custom.activities.hide.dropListeners"
    name="Deactivate dropListeners">
  </activity>
  <activityPatternBinding
    activityId="com.heiler.ppm.custom.activities.hide.dropListeners"
    pattern="com.heiler.ppm.article.ui/
com.heiler.ppm.article.ui.internal.transfer.attribute.ArticleAttributeStructureGroupAttributeDropListener">
  </activityPatternBinding>
</extension>

```

NOTE: the *id* attribute of the **dropListeners** element is now mandatory, so you can use this *id* to disable a specific drop listener.

To deactivate a **drop-choice** element use its *identifier* attribute:

```

<extension

```

```

        point="com.heiler.ppm.std.ui.entityItemEditors">
    <drop-choice
        class="com.heiler.ppm.article.ui.internal.transfer.ArticleStructureGroupDropC
hoice"
        identifier="com.heiler.ppm.article.ui.articleStructureGroupReference"
        ...
    </drop-choice>
</extension>
...
<extension
    point="org.eclipse.ui.activities">
    <activity
        id="com.heiler.ppm.custom.activities.hide.dropChoice"
        name="Deactivate drop choices">
    </activity>
    <activityPatternBinding
        activityId="com.heiler.ppm.custom.activities.hide.dropChoice"
        pattern="com.heiler.ppm.article.ui/
com.heiler.ppm.article.ui.articleStructureGroupReference">
    </activityPatternBinding>
</extension>

```

7.15.8 Status lines

You can also disable the default *status line* in the table views using **activities**:

```

<extension
    point="org.eclipse.ui.activities">
    <activity
        id="com.heiler.ppm.custom.activities.hide.tableViewStatusLines"
        name="Deactivate tableViewStatusLines">
    </activity>
    <activityPatternBinding
        activityId="com.heiler.ppm.custom.activities.hide.tableViewStatusLines"
        pattern="com.heiler.ppm.std.ui/
com.heiler.ppm.std.ui.table.statusline.defaultTableViewStatusLine">
    </activityPatternBinding>
</extension>

```

7.15.9 Status text resolvers

If you disable the default *status text resolver*, make sure that any other *resolver* is contributed, otherwise the *status line* is always empty.

```

<extension
    point="org.eclipse.ui.activities">
    <activity

```

```

        id="com.heiler.ppm.custom.activities.hide.statusTextResolvers"
        name="Deactivate statusTextResolvers">
    </activity>
    <activityPatternBinding
        activityId="com.heiler.ppm.custom.activities.hide.statusTextResolvers"
        pattern="com.heiler.ppm.std.ui/
com.heiler.ppm.std.ui.table.statusline.defaultStatusTextResolver">
    </activityPatternBinding>
</extension>

```

7.15.10 FieldBasedTableViewExtensionManager

Although a *FieldBasedTableViewExtensionManager* contribution can be overridden by another one with a higher **rank**, it is also possible to disable a *manager* contribution by using *activities*:

```

<extension
    point="org.eclipse.ui.activities">
    <!-- disable view extension manager -->
    <activity
        id="com.heiler.ppm.customizing.activities.disable.viewExtensionManager"
        name="Disable ViewExtensionManager">
    </activity>
    <activityPatternBinding

activityId="com.heiler.ppm.customizing.activities.disable.viewExtensionManager"
        pattern="com.heiler.ppm.dictionary.ui/
hlr.dictionary.ui.wordTableViewExtensionManager">
    </activityPatternBinding>
</extension>

```

7.15.11 Complete deletion of action sets

An action set is a logical group of menus and actions that should appear together at the same time. See this FAQ for more detailed explanation.

When **activities** are used to disable standard UI elements, such elements are not removed, they are just hid. This implies that disabled elements are still present in the internals of the Eclipse RCP ecosystem.

Sometimes this is unacceptable because of the additional overhead, e.g. disabled elements were registered as listeners for selection and if selection count increases - processing time also increases.

Specify action set id's as comma-separated values in the `com.heiler.ppm.main/completely-deleted-actionsets` preference of `plugin_customization.ini` file. This will completely delete corresponding action sets from the internals of the Eclipse RCP ecosystem.

For example: `com.heiler.ppm.main/completely-deleted-actionsets`
`= org.eclipse.ui.actionSet.openFiles, org.eclipse.ui.actionSet.keyBindings`

7.15.12 Download

You can find the full source code as SDK examples.

7.16 Possibility to replace the execution class of a view contribution

7.16.1 Motivation

In order to eliminate the necessity of modifying the standard code, there is a new possibility to replace the execution class of a view contribution. Assuming that you have to override the standard item view to add a customer-specific functionality. So you extend the class *ArticleTableView* and create you own custom article view class with the additional logic. Now if you contribute you custom class in a *view* contribution, you have both views in the PIM RichClient: the standard item view and your custom item view. So you have to modify the *plugin.xml* of the plugin *com.heiler.ppm.article.ui* to hide the standard item view.

7.16.2 Solution

Now from PIM 7.0.01.00 there is a new possibility to contribute you own custom class without modifying the standard code. There is a new extension point - **com.heiler.ppm.std.core.contributionClassProviders** - which allows you to replace the execution class of a standard contribution with your own custom class. In order to allow the using of this extension point, all standard view contributions were modified. Now the *class* attribute of a view contribution contains instead of the name of the view class the name of the specific *class factory*. This factory is called when the execution class of a view contribution is loaded. At this point we can check, if there are extensions of the **com.heiler.ppm.std.core.contributionClassProviders** for the corresponding view ID.

Now a standard view contribution looks like this:

```
<extension
    point="org.eclipse.ui.views">
    <view
        allowMultiple="true"
        icon="icons/ArticleList.gif"
        class="com.heiler.ppm.std.core.contribution.ContributionClassFactory:com.heiler.ppm.article.ui.internal.view.ArticleTableView:Article"
        category="com.heiler.ppm.views.categories.Article"
        name="%view.articles.name"
        id="com.heiler.ppm.article.ui.views.ArticleTableView"/>
    </extension>
```

So if you can see, the *class* attribute contains following parts: the name of the factory class + the name of the base class + (optional) initialization data. All parts are separated by the ':'

The *ContributionClassFactory* tries to find an extension of the point **com.heiler.ppm.std.core.contributionClassProviders** which is contributed for the *id* defined in the standard view contribution.

```
<extension
  point="com.heiler.ppm.std.core.contributionClassProviders">
  <classProvider
    class="com.heiler.ppm.custom.article.ui.internal.CustomArticleView:Article"
    contributionId="com.heiler.ppm.article.ui.views.ArticleTableView">
  </classProvider>
</extension>
```

If a corresponding extension was found - the custom class defining in the custom contribution will be loaded for the view contribution. If there is no corresponding extension - the base class defining in the view contribution will be loaded.

7.16.2.1 Base class

The original view contribution contains in the *class* attribute the name of the base class. This class will be instantiated when no *classProvider* contribution for the view ID was found. **Please note** that the corresponding custom view class should extend the base class defined in the original view contribution. If the custom view class is not an instance of the base class, a *CoreException* will be thrown.

7.16.2.2 Priority

By using of the *priority* attribute you can determine which custom contribution for a particular contribution ID. If there are several *classProvider* contributions for the same *contributionId*, the one with the highest priority will be used. If the *priority* attribute is omitted - the priority of **100** is used. So set the value of the *priority* attribute to a number higher than 100 to ensure, that a specific *classProvider* contribution is used.

```
<extension
  point="com.heiler.ppm.std.core.contributionClassProviders">
  <classProvider
    class="com.heiler.ppm.custom.article.ui.internal.CustomArticleView:Article"
    contributionId="com.heiler.ppm.article.ui.views.ArticleTableView"
    priority="200">
  </classProvider>
</extension>
```

7.16.2.3 Additional attributes

You can also add additional attributes to the *classProvider* contribution such as *name* or *icon*. In this case the custom view name and icon will be shown in the view title.

```
<extension
  point="com.heiler.ppm.std.core.contributionClassProviders">
  <classProvider
    class="com.heiler.ppm.custom.article.ui.internal.CustomArticleView:Article"
    contributionId="com.heiler.ppm.article.ui.views.ArticleTableView"
    priority="200"
```



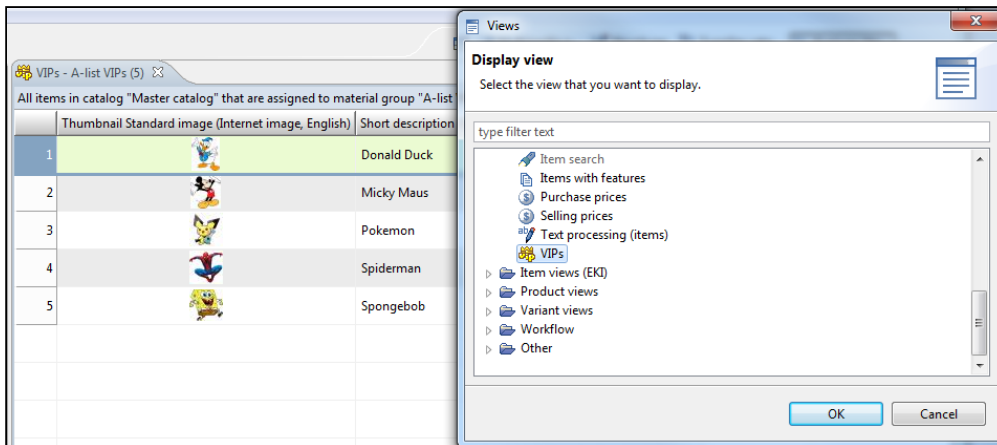
```

name="%view.vips.name"
icon="icons/vip.gif">
</classProvider>
</extension>

```

Note: since the additional attributes are not defined in the extension point **com.heiler.ppm.std.core.contributionClassProviders** - a warning will be shown in the *plugin.xml* editor. You can just ignore this warnings.

The custom name and icon are shown in the view title and in the "Open view..." dialog:



7.16.3 Outlook

Basically the possibility to replace the execution class is not view specific. Theoretically it would work in any other contribution which have a *class* attribute. The new extension point **com.heiler.ppm.std.core.contributionClassProviders** and the factory class *ContributionClassFactory* are quite generic and are not view specific. However, currently only *view* contributions are enabled for the replacing of the execution class. If you need this possibility for other kind of contributions in the standard code, please contact the PIM development team and explain your request.

7.16.4 Download

You can find a full source code example for this here.

7.17 Richtext conversion



NOTE: this article relates to the PIM version 7.1.03

7.17.1 Media-neutral format

To support the "media-neutral" storage of the rich-texts, the text edited within the RichText-Editor will be converted before saving.

7.17.2 Internal markup

On the conversion some of HTML tags (outputted by the RichText-Editor) will be replaced by our own tags, which are more "media-neutral". Following internal tags are available:

HTML tag	Internal markup	Description	Remarks
 	<CRLF>	Line break	see also Rich Text editing
<p>	<PAR>	Paragraph	see also Rich Text editing

On the other side on the loading of a saved richt-text into the RichText-Editor the internal tags will be replaced by the HTML tags again to provide a proper rendition.

7.17.3 Overview of single conversion steps

While conversion there are following steps (executed in the order as below):

- From **XHTML** to **NEUTRAL**:
 - a. **Carriage returns**
 - i. all " \n \n " are replaced with " <CRLF> " (related to the CKEditor bug <http://dev.ckeditor.com/ticket/12879>)
 - ii. all " \r\n " are replaced with ""
 - iii. all " \n\t " are replaced with ""
 - iv. all " \n " are replaced with ""
 - b. **TABs**
 - i. all 4-times non-breaking spaces () are replaced with " \t "
 - c. **Non-breaking spaces**
 - i. all remaining are removed (if the corresponding setting `com.heiler.ppm.richtext.server/remove-nbsp` is set to **true**)
 - d. **Paragraphs**
 - i. if the paragraph support is **enabled** (the server-side setting `com.heiler.ppm.richtext.server/enable-paragraphs` is **true**):

1. all "<p>" tags are replaced with "<PAR>"
2. all "</p>" are replaced with "</PAR>"
- ii. if the paragraph support is **disabled**:
 1. the "<p>" tag at the beginning is removed
 2. the "</p>" at the end is removed
 3. if a paragraph is followed by another paragraph - it is replaced by "<CRLF>"
 4. finally all remaining "<p>" and "</p>" tags are removed
- e. **BRs**
 - i. all "
" tags are replaced with "<CRLF>" tags
- f. **Ampersands**
 - i. nothing happens by default (is only a "hook" for customizings)
- From **NEUTRAL** to **XHTML**
 - a. **Ampersands**
 - i. nothing happens by default (is only a "hook" for customizings)
 - b. **BRs**
 - i. all "<CRLF>" tags are replaced with "
" tags
 - c. **Paragraphs**
 - i. all "<PAR>" tags are replaced with "<p>"
 - ii. all "</PAR>" are replaced with "</p>"
 - iii. all **empty** paragraphs are replaced with "<p> </p>"
 - d. **Non-breaking spaces**
 - i. nothing happens by default (is only a "hook" for customizings)
 - e. **TABS**
 - i. all "\t" are replaced with 4-times non-breaking spaces ()
 - f. **Carriage returns**
 - i. nothing happens by default (is only a "hook" for customizings)

7.17.4 Customization of individual conversion steps

Each of the conversion steps listed above may be customized. To be able to customize the rich-text conversion, it is necessary to implement and contribute a `RichTextMarkupConversionHandler`. The conversion handler will be used each time a rich-text should be converted from "XHTML" to "NEUTRAL" and vice versa. There is a default implementation of the conversion handler -

`DefaultRichTextMarkupConversionHandler` - which just contains the standard conversion logic (as described above). It is recommended to use the `DefaultRichTextMarkupConversionHandler` as a basis-class for the custom implementation. In this way the custom conversion handler has only to override those conversion steps which should be modified. The default conversion handler is located in a public package - **com.heiler.ppm.richtext.core.conversion** - and can be used as a template for customizings.

To contribute a custom conversion handler just use the extension point

`com.heiler.ppm.richtext.core.richtextMarkupConversionExtensions`:

```
<extension
  point="com.heiler.ppm.richtext.core.richtextMarkupConversionExtensions">
```

```

    <conversionHandler
      class="com.heiler.ppm.customizing.ckeditor.internal.CustomRichTextMarkupC
onversionHandler"
      id="hlr.customizing.ckeditor.internal.customRichTextMarkupConversionHandl
er"
      rank="100">
    </conversionHandler>
  </extension>

```

If more than one conversion handler is registered - the one with the higher **rank** will be used. The `DefaultRichTextMarkupConversionHandler` is contributed with the **rank** "1". So use a higher **rank** for your custom conversion handler to force the usage of this one.

IMPORTANT: put the custom plugin to the **client** (to make it work in PIM Desktop) or/and to the **server** (to make it work in PIM Web).

7.17.5 Sample implementation of a custom conversion handler

As a sample implementation of a custom conversion handler and as a potential solution for customers which reported the issue

■ HPM-204

57 -

Problem
with non-
breaking
spaces in

the `CustomRichTextMarkupConversionHandler` was implemented and is located in SDK examples (`com.heiler.ppm.customizing.ckeditor`).

This sample conversion handler implements a kind of special treatment for **non-breaking spaces** (NBSPs). At first it differentiates between "wanted" and "unwanted" NBSPs. Wanted NBSPs are real "non-breaking spaces" which were put between two words to prevent a line-break. This NBSPs might be *copy&pasted* from the other software (like PDF reader) or added somehow else. Important is: the user wants to keep them! Another kind of NBSPs are NBSPs which were added by the `CKEditor` for example to escape leading/trailing spaces. So the implementation of the sample custom conversion handler tries to find such "unwanted" NBSPs and either remove them or replace them with spaces. Therefore there are two modes: "REMOVE" or "REPLACE". Please set the desired mode for the customer. See the mentioned class in SDK examples for more implementation details.

7.18 Richtext validation

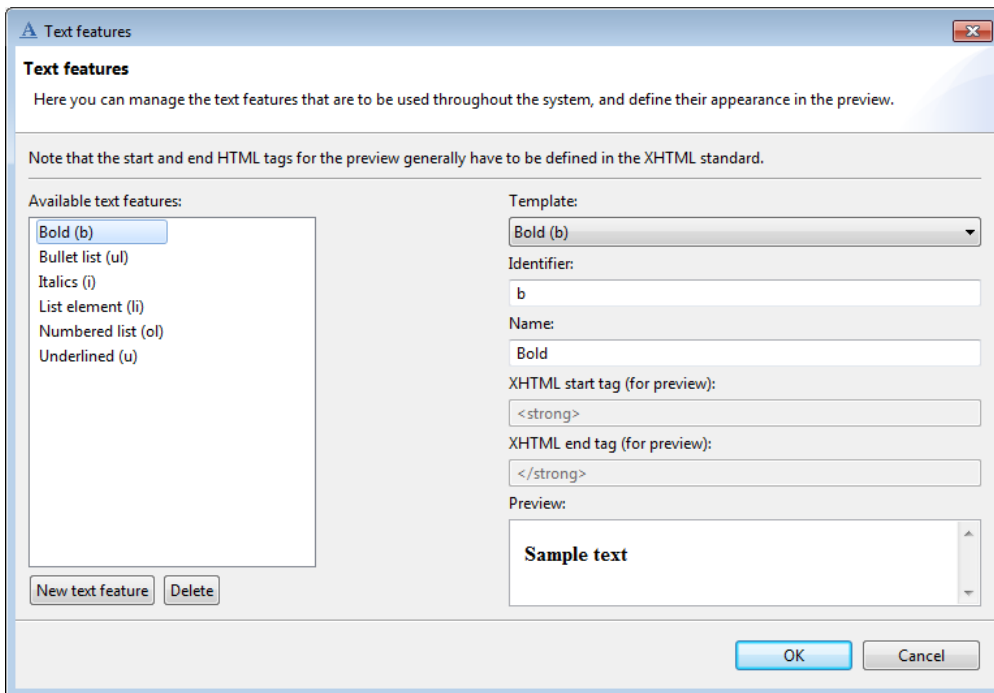
Before the rich-text is stored in the database it will be validated relating to text formattings (in other words "tags"). By default all tags are supported which are defined as so called "text features".

7.18.1 Text features

The text features are predefined format templates which can be applied to the rich-text while editing. There is a set of standard text features which are created first time the text features are used.

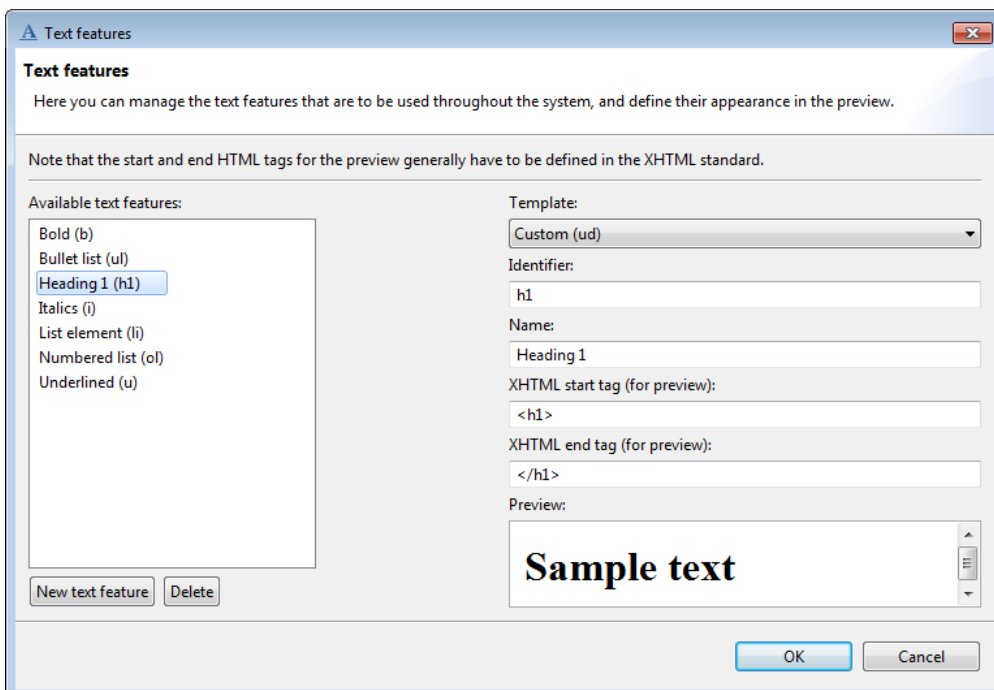


NOTE: the initial creation of the standard text features is performed for the language of the client which requests the first time the text features. After the initial creation is done, the standard text features are only available in this one language. There is already an "idea" for this issue: HPM-20684

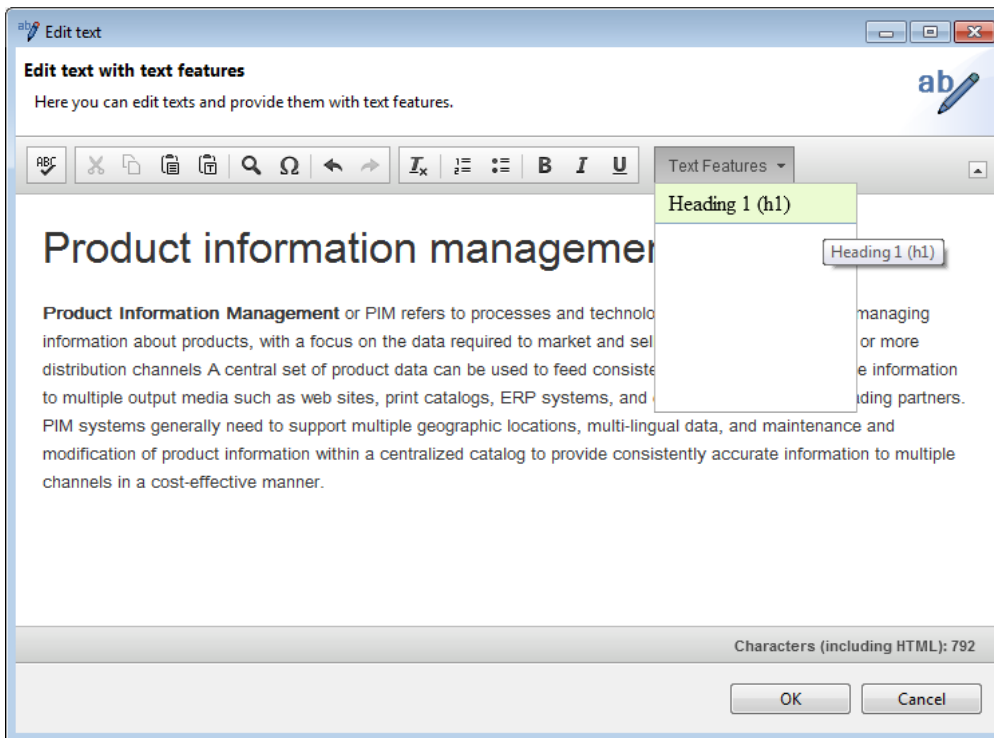


Additional to the standard text features there is a possibility to define custom text features.

To create a custom text feature just click on the button "New text feature" within the "Text features" dialog. The select the custom template from the drop-down list and put the required tag information.



After the custom text feature is created it can be used e.g. while rich-text editing. All custom text features appear in the corresponding drop-down list within the RichText-Editor:

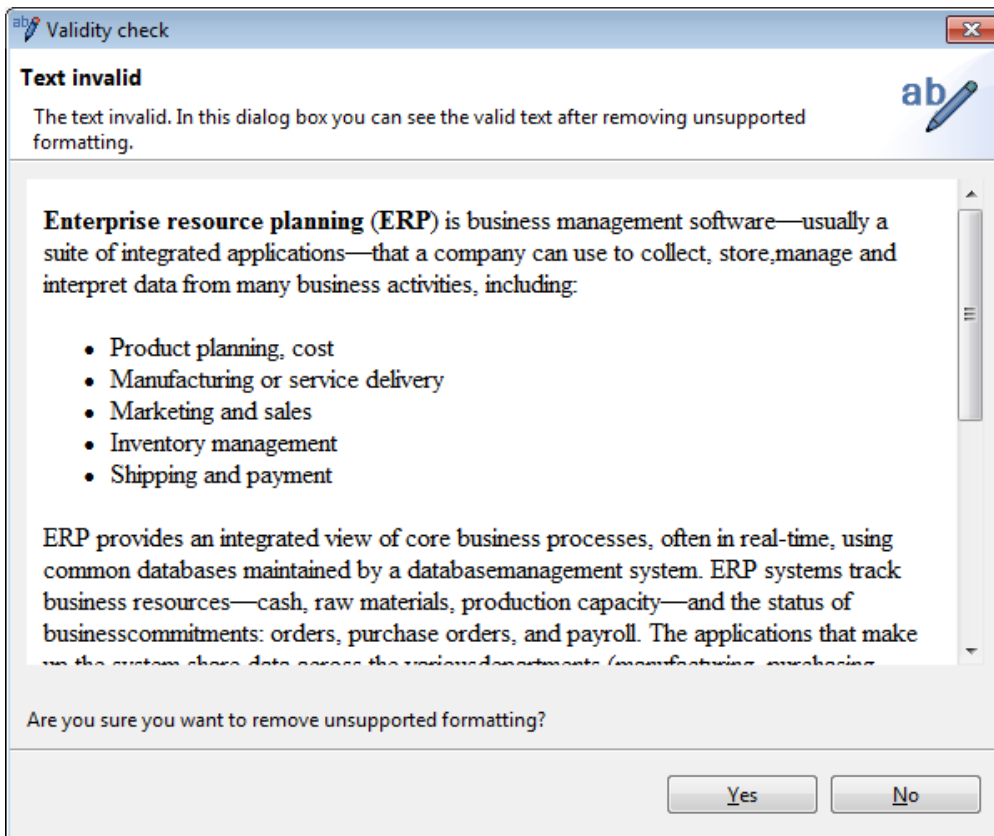


7.18.2 Validation of text formattings

While editing the rich-text within the RichText-Editor it can happen that unsupported text formattings are applied (maybe by Copy&Paste some text from a web page or from a "Word" document). On saving of the rich-text all text formattings are checked whether they are "valid". Following text formattins are valid:

- Standard text features (like "Bold", "Italic" etc.) which are created automatically on first use
- Custom text features created by the user within the "Text features" dialog
- Those tags which are determined as "supported" tags by any contributed `RichTextValidator` (see the corresponding section about *RichTextValidators*)

If any text fromattings are found which are not supported, they will be removed from the rich-text. In this case a confirmation dialog will be shown. The user can decide whether this "unsopported" formatting should be removed from the text:



If the user confirms with "Yes" the corresponding formattings will be removed from the rich-text before saving.

7.18.3 RichTextValidators extension point

Sometimes it is necessary to support specific text formattings although they are not defined as a text feature. For example all text formattings which contain any attributes within the start-tag, are not supported by default. In this case you can contribute a custom `RichTextValidator` which will be requested every time the rich-text validation is performed. The `RichTextValidator` implements only one method: `isSupportedTag()`. This method will be called for both tags: start-tag and end-tag. So if you implement your custom `RichTextValidator` ensure that it returns `true` for both tags.

Here is an example of a custom `RichTextValidator` which supports all **** tags:

```

public class CustomRichTextValidator implements RichTextValidator
{
    @Override
    public boolean isSupportedTag( String tag )
    {
        return ( tag.startsWith( "<font" ) || tag.startsWith( "</font>" ) ); //$NON-
NLS-1$ //$NON-NLS-2$
    }
}

```

To register this custom RichTextValidator just use the new extension point com.heiler.ppm.richtext.core.richtextValidators:

```

<extension
    point="com.heiler.ppm.richtext.core.richtextValidators">
    <richTextValidator
        class="com.heiler.ppm.customizing.ckeditor.internal.CustomRichTextValidat
or"
        id="hlr.customizing.ckeditor.customRichTextValidator">
    </richTextValidator>
</extension>

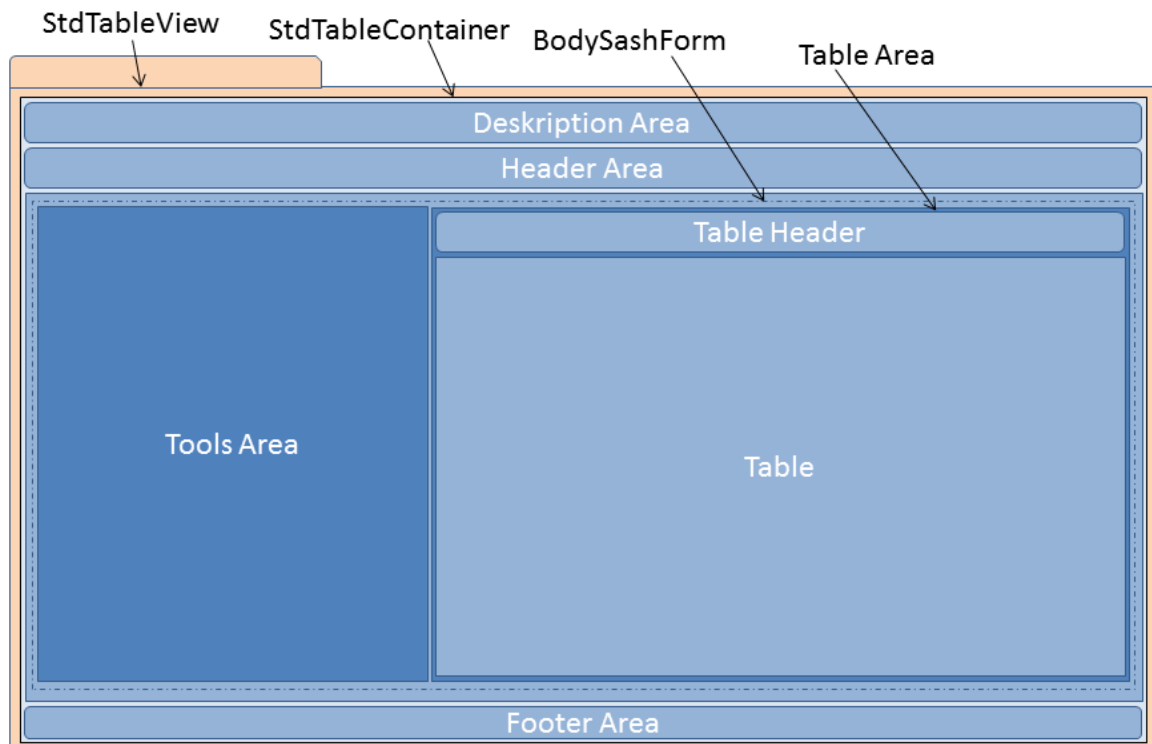
```



NOTE:

- the extension point "com.heiler.ppm.richtext.core.richtextValidators" is available since PIM 7.1.02
- the custom RichTextValidator should be registered on the **PIM Client** since the rich-text validation is performed on the client-side

7.19 Table View Layout



7.20 UI Style Guide

This style guide contains specific instructions on how to design and develop the HPM application UI.

- [Style Guide Is Not a UI Specification](#) (see page 329)
- [Layout](#) (see page 329)
 - [Workbench](#) (see page 330)
 - [Perspectives](#) (see page 330)
 - [Views](#) (see page 330)
 - [Editors](#) (see page 330)
- [Text](#) (see page 330)
- [Components](#) (see page 331)
 - [Commands](#) (see page 331)
 - [Menus](#) (see page 331)
 - [Toolbars](#) (see page 331)

- [Status Bar](#) (see page 331)
- [Dialogs](#) (see page 331)
 - [Validations](#) (see page 331)
- [Wizards](#) (see page 331)
- [Interaction style](#) (see page 331)
 - [Drag & Drop](#) (see page 331)
 - [Context menu](#) (see page 331)
 - [Keyboard navigation](#) (see page 332)
- [Error Handling](#) (see page 332)
- [Language](#) (see page 332)
- [Icons](#) (see page 332)
- [Preferences](#) (see page 332)
- [Help](#) (see page 332)

7.20.1 Style Guide Is Not a UI Specification

Style guides are different from user interface specifications:

- A specification document details the functionality of a UI design for developers building an application. It is usually more descriptive and is often accompanied by wireframes that act as blueprints for the design. In contrast, a style guide is often a general outline of the elements of a UI design.
- Style guides have a longer shelf-life than specifications documents that are often tied to a project life-cycle. When an application is first created, some elements of the initial specification document might turn into the application style guide for long-term reference.
- Elements of a style guide may be referred to from a specification. For example, the functionality of a web application enhancement would be captured in a specifications document; but the operation of standard UI controls found throughout the website would be outlined in the website style guide and referred to by the specifications document.

7.20.2 Layout

Size controls and panes within a window to match their typical content. Avoid truncated text and their associated ellipses. Users should never have to interact with a window to view its typical content reserve resizing and scrolling for unusually large content. Specifically check:

- **Control sizes.** Size controls to their typical content, making controls wider, taller, or multi-line if necessary. Size controls to eliminate or reduce scrolling in windows that have plenty of available space. Also, there should never be truncated labels, or truncated text within windows that have plenty of available space. However, to make text easier to read, consider limiting line widths to 65 characters.
- **Column widths.** Make sure table view columns have suitable default, minimum, and maximum sizing. Choose table views default column widths that don't result in truncated text, especially if there is space available within the table view.
- **Layout balance.** The layout of a window should feel roughly balanced. If the layout feels left-heavy, consider making controls wider and moving some controls more to the right.
- **Layout resize.** When a window is resizable and data is truncated, make sure larger window sizes show more data. When data is truncated, users expect resizing windows to show more information.

Set a minimum window size if there is a size below which the content is no longer usable. For resizable controls, set minimum resizable element sizes to their smallest functional sizes, such as minimum functional column widths in list views.

7.20.2.1 Workbench

7.20.2.2 Perspectives

7.20.2.3 Views

7.20.2.4 Editors

7.20.3 Text

- **Use ordinary, conversational terms when you can.** Focus on the user goals, not technology. This is especially effective if you are explaining a complex technical concept or action. Imagine yourself looking over the user's shoulder and explaining how to accomplish the task.
- **Be polite, supportive, and encouraging.** The user should never feel condescended to, blamed, or intimidated.
- **Remove redundant text.** Look for redundant text in window titles, main instructions, supplemental instructions, content areas, command links, and commit buttons. Generally, leave full text in main instructions and interactive controls, and remove any redundancy from the other places.
- **Use Headline style capitalization** for menus, tooltip and all titles, including those used for windows, dialogs, tabs, column headings and push buttons. Capitalize the first and last words, and all nouns, pronouns, adjectives, verbs and adverbs. Do not include ending punctuation.
- **Use Sentence style capitalization** for all control labels in a dialog or window, including those for check boxes, radio buttons, group labels, and simple text fields. Capitalize the first letter of the first word, and any proper names such as the word Java.
- **For feature and technology names, be conservative in capitalizing.** Typically, only major components should be capitalized (using title-style capitalization).
- **For feature and technology names, be consistent in capitalizing.** If the name appears more than once on a UI screen, it should always appear the same way. Likewise, across all UI screens in the program, the name should be consistently presented.
- **Don't capitalize the names of generic user interface elements**, such as toolbar, menu, scroll bar, button, and icon.
 - **Exceptions:** Address bar, Links bar, ribbon.
- **Don't use all capital letters for keyboard keys.** Instead, follow the capitalization used by standard keyboards, or lowercase if the key is not labeled on the keyboard.
- **Ellipses mean incompleteness.** Use ellipses in UI text as follows:
 - **Commands.** Indicate that a command needs additional information. Don't use an ellipsis whenever an action displays another window---only when additional information is required. Commands whose implicit verb is to show another window don't take an ellipsis, such as Advanced, Help, Options, Properties, or Settings.
 - **Data.** Indicate that text is truncated.
 - **Labels.** Indicate that a task is in progress (for example, "Searching...").

Tip: Truncated text in a window or page with unused space indicates poor layout or a default window size that is too small. Strive for layouts and default window sizes that eliminate or reduce the amount of truncated text. For more information, see [#Layout \(see page 329\)](#).
- **Don't use blue text that isn't a link, because users may assume that it is a link.** Use bold or a shade of gray where you'd otherwise use colored text.
- **Use bold sparingly to draw attention to text users must read.**

- **Use the main instruction to explain concisely what users should do in a given window or page.** Good main instructions communicate the user's objective rather than focusing just on manipulating the UI.
- **Express the main instruction in the form of an imperative direction or specific question.**
- **Don't place periods at the end of control labels or main instructions.**
- **Use one space between sentences.** Not two.

7.20.4 Components

7.20.4.1 Commands

7.20.4.2 Menus

7.20.4.3 Toolbars

7.20.4.4 Status Bar

7.20.4.5 Dialogs

Validations

7.20.4.6 Wizards

7.20.5 Interaction style

7.20.5.1 Drag & Drop

If for a particular data type the drag and drop behavior is supported, it should generally be possible for all users, to drag one or multiple records of this data type. On drop of the records should be checked whether the user has needed action rights and, if not, the drop operation is aborted and a corresponding information message is displayed.

7.20.5.2 Context menu

Design a context menu to display only those options that are actually available in the current context for a particular user. That means, if the current user has no permissions to perform an action, the corresponding menu entry should not be displayed. Privileged users might be shown more options than regular users. Offer a CHANGE option only to users who are authorized to make changes to the particular data being displayed.

If an action can not be executed, due to the current context, the corresponding menu item should be disabled (grayed out).

When menus are provided in different views, design them so that option lists are consistent in wording and ordering.

7.20.5.3 Keyboard navigation

7.20.6 Error Handling

7.20.7 Language

7.20.8 Icons

7.20.9 Preferences

7.20.10 Help

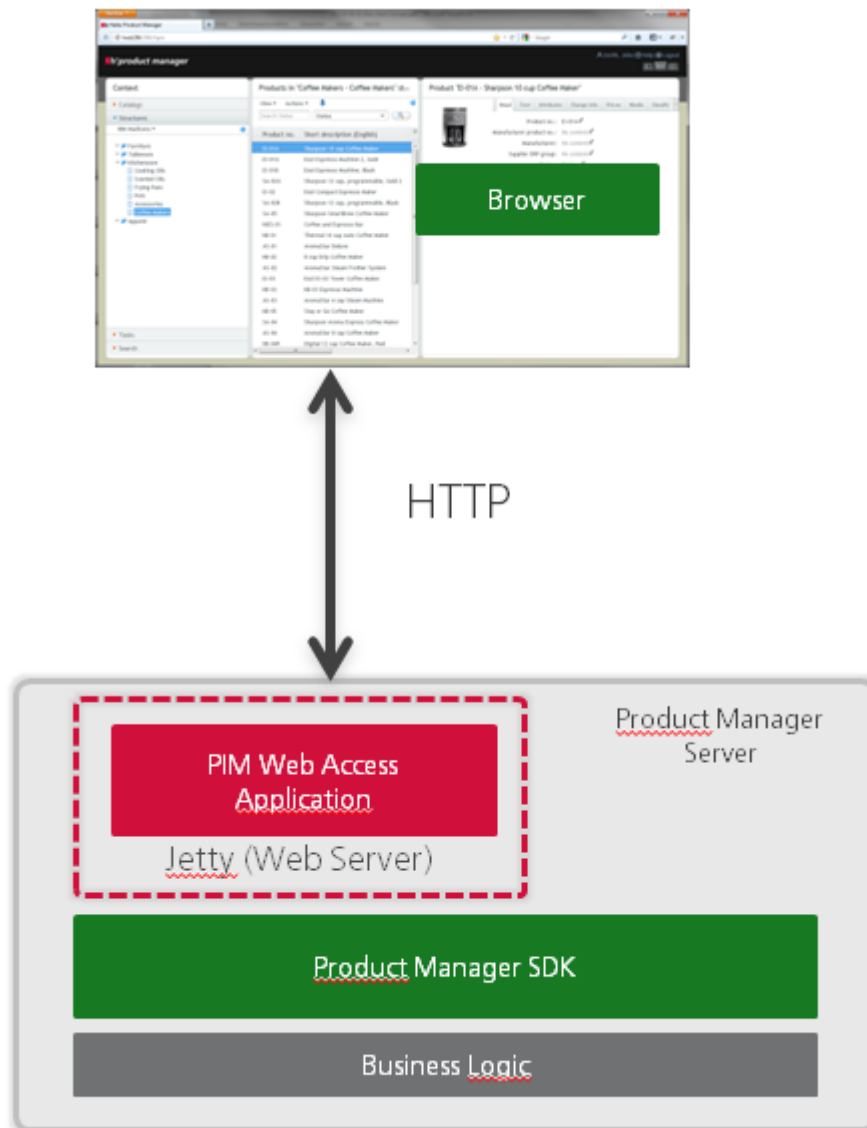
8 Web Client

8.1 Technical Overview

Technical Training Slides

8.2 Product 360 Web UI and Application Server

The Web UI modules run directly within the Product 360 Server. That means that during deployment, no additional Web/Application Server is needed. Instead, all functionality is implemented within a set of OSGi modules on top of the Product 360 SDK.



For more information on installation and deployment scenarios refer to the [installation guide](#) (see page 332).

8.3 Technology Stack

8.3.1 OSGi

Since the Product 360 Server is an Eclipse RCP based application that uses OSGi as modularization container, all Web UI modules are OSGi bundles, too.

8.3.2 Guice

Google Guice is used as dependency injection framework. Each OSGi bundle typically provides a Guice module. Guice modules are registered using an Extension Point and are picked up during the application bootstrap to resolve all dependencies.

An important aspect when binding classes to a guice context are scopes. The following scopes are used in the Web UI:

- Application scope, i.e. a single instance for the application. These classes must be implemented thread safe as they are shared by multiple sessions / requests.
- Session scope, i.e. a single instance per http session. Within a session, multiple application windows can exist. Implementation don't need to be thread safe, as Vaadin synchronizes on the Application object.
- Window scope, i.e. a single instance of browser tab (window).

The definition of scopes must be done carefully, to avoid unintended sharing of resource, concurrency issues or dead locks.

8.3.3 Vaadin

The Web UI Widget Toolkit is provided by Vaadin. Vaadin provides a server-side programming model for web applications. It uses GWT on the client for rendering and a proprietary protocol for syncing changes between the server's session state and the client's ui in the browser. You can check the exact Vaadin version used by searching for the Vaadin plugins in the Servers plugin folder of your deployment.

8.4 Asynchronous loading of custom dashboard components

8.4.1 Introduction

A business dashboard is composed of a number of components, like a pie chart, a bar chart or a task list. If one of the used components doesn't work asynchronously it will not only block itself, but the entire user interface, until it has processed everything that is required to display the desired data. Because of that it isn't even possible to switch to another area of the web application while it is loading the information that has to be displayed.

This is why we introduced the possibility to load data for components asynchronously. In order to achieve an asynchronous behavior we introduced a service providing the possibility to execute an operation on the server in a separate thread. After the execution has terminated a callback is invoked which updates the user interface accordingly. To let a user know that data is still being processed and will be displayed afterwards we also provide a layout class which is capable of displaying a loading indicator.

8.4.2 Technical limitations

Asynchronous loading needs the following prerequisites to be used on a dashboard:

- The web application needs to support push mode. If `webfrontend.properties` has the property `web.vaadin.pushMode` set to `DISABLED`, then synchronous loading will be used for all dashboards.
- All components on a given dashboard must support asynchronous loading. If one of them does not, the whole dashboard will load synchronously. The following sections contain a guide on how to implement asynchronous loading in a dashboard component.

8.4.2.1 Asynchronous loading and Vaadin push mode detection

To determine if a component supports asynchronous loading use the interface `AsyncLoadingFlexComponent`. General support is indicated by following method:

Asynchronous loading support

1	<code>AsyncLoadingFlexComponent.supportsAsynchronousLoading();</code>
---	---

The method automatically returns `true` for all components of the type `AbstractAsyncLoadingFlexComponent`. The method may be overwritten to determine per component whether asynchronous loading is supported. So a component is considered to support asynchronous loading only if it implements the `AsyncLoadingFlexComponent` interface and the method `supportsAsynchronousLoading()` returns `true`.

For determining whether asynchronous loading is finally to be used per component, use the flag `useAsynchronousLoading` of the interface `AsyncLoadingFlexComponent`.

Determine whether to use asynchronous loading

1	<code>AsyncLoadingFlexComponent.isUseAsynchronousLoading();</code>
---	--

The flag is set to `false` for all components during creation of a certain dashboard via `AsyncLoadingFlexComponent.setUseAsynchronousLoading(boolean useAsynchronousLoading)` if one of these following conditions is true:

- One of the components on the dashboard does not support asynchronous loading.
- Push mode is set to `DISABLED`.

Use the following code snippet to check if Vaadin's push mode is generally available.

Check Vaadin's push mode

1	<code>UI.getCurrent()</code>
---	------------------------------

```

2    .getPushConfiguration()
3    .getPushMode()
4    .isEnabled();

```

Useful code snippets and guidelines

The classes you will be working with are:

- `AsyncServerOperationExecutorFactory` : Provides an interface to pass a `Callable< ResultType >` and an implementation of `ServerOperationCallback< ResultType >` for creating a `AsyncServerOperationExecutor`.
- `AsyncServerOperationExecutor` : The actual executor.
- `ServerOperationCallback< ResultType >` : Interface that has to be implemented for handling a callback from `AsyncServerOperationService`.
- `AbstractAsyncLoadingFlexComponent` : An abstract component you should extend to build your own components. It implements the interface `AsyncLoadingFlexComponent`, which provides the necessary interface for controlling the overall dashboard behavior.

8.4.2.2 Getting `AsyncServerOperationExecutorFactory`

This is done by a simple injection.

Getting `AsyncServerOperationExecutor`

```

1    @Inject
2    private AsyncServerOperationExecutorFactoryService asyncExecutorFactory;

```

8.4.2.3 Calling `AsyncServerOperationExecutorFactory`

The interface provides only one method with four parameters of the following types:

1. `String`
2. `String`
3. `Callable< ResultType >`
4. `ServerOperationCallback< ResultType >`

This results in a nice call like this.

Calling `AsyncServerOperationExecutor`

```

1    public void doSomething()

```

```

2    {
3        AsyncServerOperationExecutor executor = this.asyncExecutorFactory.create
        ( REQUEST_TYPE, this.componentId, asyncServerOperationCallable, new
          ServerOperationCallbackImplementation< ResultType > );
4        executor.execute();
5    }

```

You'll be given an example on how to build each of the three required parameters below.

8.4.2.4 Implementing `Callable< ResultType >`

You have to implement the method `call()` with a fitting return type. Usually this would call another method which does the actual work for a better readability.

An implementation may look like the following example.

Extending `AbstractServerOperationRequest`

```

1    Callable< ResultType > asyncServerOperationCallable = new Callable<
2        ResultType >()
3    {
4        @Override
5        public ResultType call() throws Exception
6        {
7            return getSomeResultType();
8        }
9    };

```

8.4.2.5 Implementing `ServerOperationCallback< ResultType >`

For handling the callback after a `Callable< ResultsType >` has been executed you need to implement the interface `ServerOperationCallback< ResultType >`. To do so you have to override two methods:

1. `void finished(ServerOperationResult< ResultType> serverOperationResult)`: This method is called when the execution was successful.
2. `void error(Throwable t)`: This method is called when the execution failed.

Usually your implementation will look like the code below.

Implementing `ServerOperationCallback< ResultType >`

```

1    private class MyCallback implements ServerOperationCallback< ResultType >
2    {

```

```

3      @Override
4      public void finished( ServerOperationResult< ResultType >
      serverOperationResult )
5      {
6
7          handleServerOperationResult( serverOperationResult.getOperationResult() );
8      }
9
10     @Override
11     public void error( Throwable t )
12     {
13         handleServerError( t );
14     }

```

We recommend implementing this as a private inner class of your dashboard component class, because it is unlikely that anything else needs access to it. Of course there might exceptions from this recommendation, depending on the dashboard components you're implementing.

8.4.2.6 Purpose of using a generic `< ResultType >`

The API is implemented using generics, so it is easy to guarantee that it always returns the expected type and thereby avoids casting classes of type `Object`.

8.4.2.7 Extending `AbstractAsyncLoadingFlexComponent`

`AbstractAsyncLoadingFlexComponent` is an abstract component that implements the interface `AsyncLoadingFlexComponent`.

A component supporting asynchronous loading by extending `AbstractAsyncLoadingFlexComponent` must still implement synchronous loading. This is because if there is any component on a dashboard that doesn't support asynchronous loading all other components on that dashboard have to behave like they're loading their data synchronously.

By implementing the interface `AsyncLoadingFlexComponent` there are three methods controlling the component's overall loading behavior.

Method	Return type	Default value
<code>supportsAsynchronousLoading()</code>	<code>boolean</code>	<code>true</code>
<code>isUseAsynchronousLoading()</code>	<code>boolean</code>	<code>true</code>

Method	Return type	Default value
<code>setUseAsynchronousLoading(boolean useAsynchronousLoading)</code>	<code>void</code>	-

Method `supportsAsynchronousLoading()`

The method specifies whether the component supports loading data asynchronously or not. It is necessary so that the underlying dashboard can decide if it is possible to use asynchronous loading at all.

Method `isUseAsynchronousLoading()`

The method specifies if the component actually uses asynchronous loading. After all, there might be circumstances in which a developer decides to actively disable that capability. The method is necessary so that the underlying dashboard can decide if it is possible to use asynchronous loading at all.

Method `setUseAsynchronousLoading(boolean useAsynchronousLoading)`

This method allows to enable or disable asynchronous loading. It can be used by a developer to actively control the component's behavior. It is necessary for the underlying dashboard to have that method implemented so that asynchronous loading can be deactivated in case another component on that dashboard doesn't allow for asynchronous loading to be used at all.

8.4.3 User interface

For implementation convenience a dashboard component should use `LoadingIndicatorVerticalLayout` as main layout for displaying the loading indicator.

An example for switching it on and off is in the code snippet below.

Switching the loading indicator on and off

```

1  private void setLoadingIndicatorVisible( boolean visible )
2  {
3      Component loadingIndicator = ( ( LoadingIndicatorVerticalLayout ) this.m
ainLayout ).getLoadingIndicator();
4      if ( loadingIndicator != null )
5      {
6          loadingIndicator.setVisible( visible );
7      }
8  }
```

8.4.4 Piecing it all together

With all the examples above your resulting class will probably look similar to the one below. We highly recommend sticking to the practice of using private inner classes and dealing with results and errors in the actual class. This way the requests and callbacks in each of your dashboard components should look very similar.

Example implementation

```

1  public class AsyncLoadingExample extends AbstractAsyncLoadingFlexComponent
2  {
3      private static final String
ASYNC_LOADING_EXAMPLE_REQUEST = "AsyncLoadingExample.Request"; //$NON-
NLS-1$
4      @Inject
5      private AsyncServerOperationExecutorFactory asyncExecutorFactory;
6      private final String          componentId
= UUID.randomUUID().toString();
7      private Integer               currentValue;
8
9      @Override
10     public void refresh( boolean reload ) // refresh method from interface
FlexComponent
11     {
12         if( reload ) {
13             loadData();
14         } else {
15             showData( this.currentValue );
16         }
17     }
18
19     public void loadData()
20     {
21         if( isUseAsynchronousLoading() ) {
22             setLoadingIndicatorVisible( true );
23
24             Callable< Integer > asyncServerOperationCallable = new Callable<
Integer >()
25             {
26                 @Override
27                 public Integer call() throws Exception
28                 {
29                     return loadingOperation();
30                 }
31             }
32
33             AsyncServerOperationExecutor executor = this.asyncExeutorFactory.cre
ate( ASYNC_LOADING_EXAMPLE_REQUEST, this.componenteId,
asyncServerOperationCallable, new MyServerOperationCallback() );
34             executor.execute();

```

```

35     } else {
36         // load data synchronously, must always be supported as well
37         this.currentValue = loadingOperation();
38         showData( this.currentValue );
39     }
40 }
41
42 public Integer loadingOperation()
43 {
44     // use only server operations, no UI relevant code
45     return new Integer(1);
46 }
47
48 public handleAsyncServerOperationResult( Integer value )
49 {
50     this.currentValue = value;
51     // visualize the result - in case the result is empty display a text
    like 'No data available'
52     setLoadingIndicatorVisible( false );
53     showData( value );
54 }
55
56 public void handleAsyncServerError( Throwable t )
57 {
58     LOG.error( "Error loading data asynchronously for AsyncLoadingExample",
    t );
59     setLoadingIndicatorVisible( false );
60     // display error as text inside dashboard component
61     showError( t );
62 }
63
64 private void showData( Integer value ) {
65     // visualize data in component
66 }
67
68 private void showError( Throwable t ) {
69     // visualize error as text, do not show stack traces, use short
    description like 'Error on loading'
70 }
71
72 private class MyServerOperationCallback implements
    ServerOperationCallback< Integer >
73 {
74     @Override
75     public void finished( ServerOperationResult< Integer >
    serverOperationResult )
76     {
77         handleAsyncServerOperationResult( serverOperationResult.getOperationResult
    ( ) );
78     }
79
80     @Override

```



```

81     public void error( Throwable t )
82     {
83         handleAsyncServerError( t );
84     }
85 }
86 }

```

8.4.5 Thread pool for background execution

The techniques described above utilize a thread pool for background execution. Details on the thread pool's configuration are described in the chapter 'Asynchronous loading thread pool configuration' of the Configuration Manual.

Memory consumption

Please keep in mind that dashboard components usually aggregate data for a lot of items, products or variants and thus consume a considerable amount of memory.

8.5 Customizing the Application Theme

- [Introduction](#) (see page 342)
- [Creating a new theme fragment](#) (see page 342)
- [Enabling theme](#) (see page 345)
- [Example of new theme](#) (see page 345)
- [Support of multiple themes](#) (see page 345)
- [Customize Login/SAML/Supplier Portal integration favicon and html page title](#) (see page 346)

8.5.1 Introduction

Vaadin separates the appearance of the user interface from its logic using *themes*. Themes can include CSS style sheets, custom HTML layouts, and any necessary graphics.

To customize the appearance of user interface one should create new theme with all necessary changes. More in-depth information is provided at Vaadin book.

Product 360 Web comes with a predefined theme named '**crystal**'. It is located in *com.heiler.ppm.web.theme* bundle.

To customize the application layout, the following steps are necessary:

1. Create a new fragment or use the example provided with the SDK
2. Put the custom CSS content into the styles.css file (only deltas)
3. Deploy the fragment into the Product Manager plugins folder

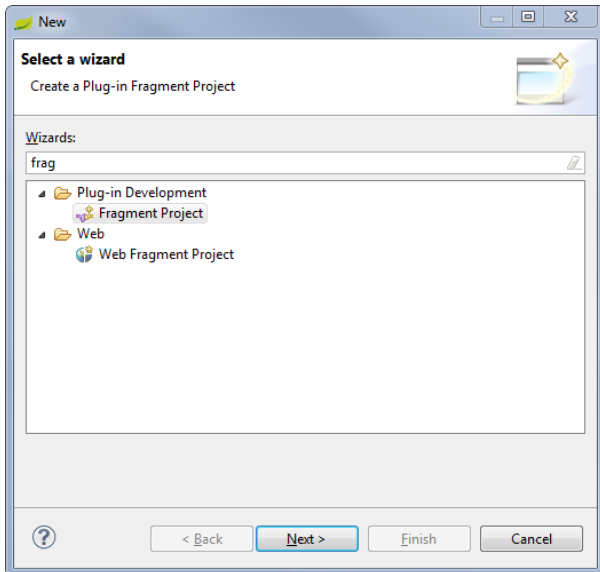
8.5.2 Creating a new theme fragment

This section describes how to create a new theme fragment from scratch. Alternatively, the SDK comes with a ready-to-use fragment *com.heiler.ppm.web.custom.theme* in the examples folder which can be used as template.

No SDK?

If you don't want to setup the SDK you can also use this plug-in project and unzip it into the Product 360 server \plugins folder. Changes to the included css affect a running application after a Browser refresh.

Use the New Project wizard of Eclipse to create a new fragment.



Custom themes are placed under /VAADIN/themes/<theme_name>/ folder. Create such directory structure under src folder, this location is fixed by Vaadin. The name of a <theme_name> folder defines the name of the theme. Assume, that <theme_name> is equal to heilerlime. A theme must contain the styles.css stylesheet, but other contents have free naming.

To inherit build-in theme one should use @import statement. For instance, to inherit 'heiler' theme one must add such import to styles.css file:

styles.css contents

```
@import "../symphony/styles.css";
```

To change background color of all panels one should add following properties to styles.css file:

styles.css contents

```
@import "../symphony/styles.css";

.v-panel-content {
    background-color: #D5F184;
```

```
}
```

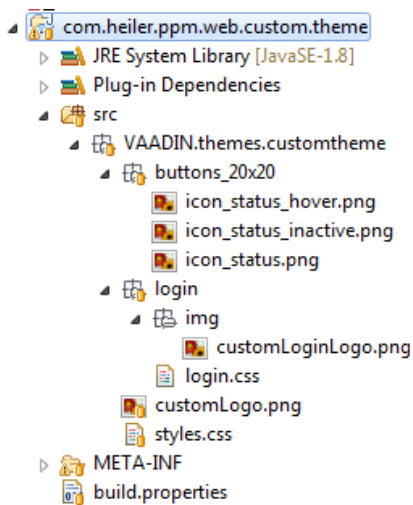
Theme bundle manifest example:

MANIFEST.MF

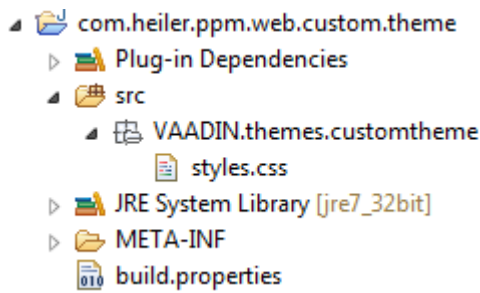
```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Theme
Bundle-SymbolicName: com.heiler.ppm.web.custom.theme
Bundle-Version: 1.0.0.qualifier
Bundle-Vendor: Heiler Software AG
Fragment-Host: com.vaadin.server;bundle-version="7.1.0"
Bundle-RequiredExecutionEnvironment: JavaSE-1.7
Export-Package: VAADIN.themes.customtheme
```

When customizing the login page it is required to define a base custom login.css inside a login subpackage of the custom theme. That login.css must refer to the base symphony style as well to have any base styling:

```
@import "../../symphony/login/login.css";
```



Example of contents of custom theme bundle:



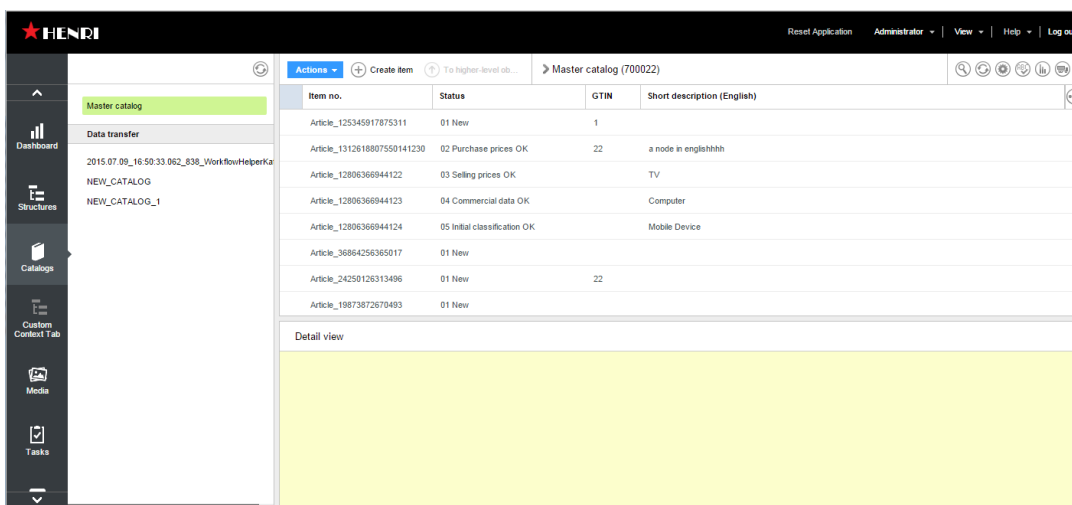
8.5.3 Enabling theme

To change theme one should modify property web.client.theme in **webfrontend.properties** file. For instance:


```
web.client.theme = customtheme
```

8.5.4 Example of new theme

After performing previous steps you will get following appearance of the PIM Web application:



8.5.5 Support of multiple themes

 This feature is available since PIM 7.1.

PIM Web allows to use different themes simultaneously in one application.

To achieve this, one should create any number of custom themes like shown in the section "Creating a new theme fragment". Each theme should have unique name. Let's suppose you created theme named **"heilerlime"**

To switch the theme to **"heilerlime"**, you should only add parameter to the current url:

Before:

`http://localhost:1501/pim/webaccess`

After:

`http://localhost:1501/pim/webaccess?theme=heilerlime`

You can change the theme at every place of the application, not only at the login page.

Before:

`http://localhost:1501/pim/webaccess#!catalog/catalogId=MASTER`

After:

`http://localhost:1501/pim/webaccess?theme=heilerlime#!catalog/catalogId=MASTER`

After you specified theme in the url parameter, theme name is stored in cookies and afterwards will be used for subsequent requests.

To go back to default theme, you should specify its name in the parameter. Currently default theme is *heiler*, so url similar to

`http://localhost:1501/pim/webaccess?theme=heiler#!catalog/catalogId=MASTER`

will bring you to default theme. Alternative is to clear your browser cookies and refresh page without "theme" parameter in the url.

If you specify wrong theme name, last used theme will be applied, taken either from cookies or default.

Using this approach one can

- create "theme switcher" widget, which will add/change url parameter and refresh the page,
- provide different urls for navigating from different locations to PIM Web (like Supplier portal),
- change url parameter depending on user or usergroup logged in.

8.5.6 Customize Login/SAML/Supplier Portal integration favicon and html page title

PIM Web Login page as well as any of the SAML- and Supplier Portal logout/error pages uses extra html files and related favicons that can be customized.

The customization of the html page and favicon will be explained by means of PIM Web Login:

The plugin `com.heiler.ppm.web.login` contains the extension point `com.heiler.ppm.http.jetty.multicontext.resources`. It connect an virtual resource (*alias*) to real resource (*base-name*) in this plugin:

```
<resource
  alias="/loginstatic/*"
  base-name="/html"
  contextPath="/pim">
</resource>
```

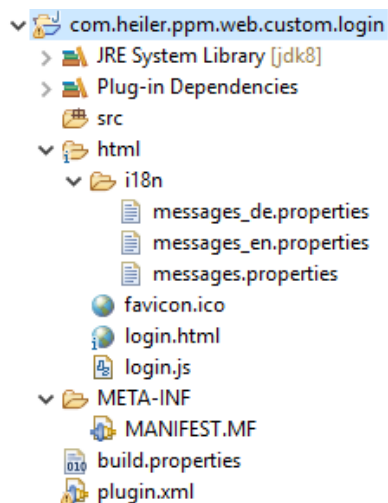
The *base-name* attribute shows to the path `/html`, which is a resource path in the *com.heiler.ppm.web.login* plugin and contains the login.html file and the favicon file, which can be customized in the following way:

1. Creating a new plugin and name it to e.g. *com.heiler.ppm.web.custom.login*.
2. The *MANIFEST.MF* file contains not that much:

```
1 Manifest-Version: 1.0
2 Automatic-Module-Name: com.heiler.ppm.web.custom.login
3 Bundle-ManifestVersion: 2
4 Bundle-Name: Custom PIM Web Login
5 Bundle-SymbolicName: com.heiler.ppm.web.custom.login;singleton:=true
6 Bundle-Version: 1.0.0.qualifier
7 Bundle-Vendor: Informatica
8 Require-Bundle: com.heiler.ppm.http.jetty.multicontext,
9   com.heiler.ppm.web.jquery;bundle-version="1.0.0",
10   org.eclipse.core.runtime;bundle-version="3.8.0"
11 Bundle-ActivationPolicy: lazy
12 Bundle-ClassPath: .
```

3. Copy the html folder to the new plugin.
4. Customized the login.html and favicon file.

The SDK comes with a ready-to-use plugin *com.heiler.ppm.web.custom.login* in the examples folder which can be used as template:



After you have include the new plugin in your PIM server, you should restart PIM, clean browsers cache and restart the browser. If the old favicon is still there, refresh the page.

8.6 Programmatic Customization of Web UI

This page shows some examples on how to programmatically extend the Web UI.



Please note, that only very few Web UI interfaces and classes have the `@api` annotation at the moment. That means, that client code potentially breaks with newer versions of the application. Providing a mature public API is in the backlog and will be part of one of the next releases. This is of course based on the feedback from customer projects.

- [SDK Basics \(see page 348\)](#)
 - [Create a new OSGi bundle skeleton \(see page 348\)](#)
 - [Launch customized application \(see page 351\)](#)
 - [Extending existing modules \(see page 352\)](#)
- [Typical Use Cases \(see page 352\)](#)
 - [Add a custom tab in the detail area \(see page 353\)](#)
 - [Add a custom context area \(see page 354\)](#)
 - [Add a custom action to the table menu bar \(see page 356\)](#)
 - [Implementing and contributing custom flex components for dashboard \(see page 358\)](#)
 - [Contribute Custom Placeholder for XML Configuration Files \(see page 359\)](#)
 - [Hide or show detail tabs dependent of arbitrary state of selected object \(see page 361\)](#)

8.6.1 SDK Basics

To extend the Web UI, a new OSGi bundle needs to be created. You can either use one of the existing examples that come with the SDK installation package or create a new bundle from scratch.

8.6.1.1 Create a new OSGi bundle skeleton

- Create a new Plug-In Project, setup basic meta data and required bundles

New Plug-in Project

Content
Enter the data required to generate the plug-in.

Properties

ID:

Version:

Name:

Vendor:

Execution Environment: [Environments...](#)

Options

☐ Generate an activator, a Java class that controls the plug-in's life cycle
Activator:

☒ This plug-in will make contributions to the UI

☐ Enable API analysis

Rich Client Application
Would you like to create a rich client application? ☐ Yes ☒ No

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

com.heiler.ppm.sdk.example [ExampleModule.java](#)

Dependencies

Required Plug-ins [ja](#) [Z](#)

Specify the list of plug-ins required for the operation of this plug-in.

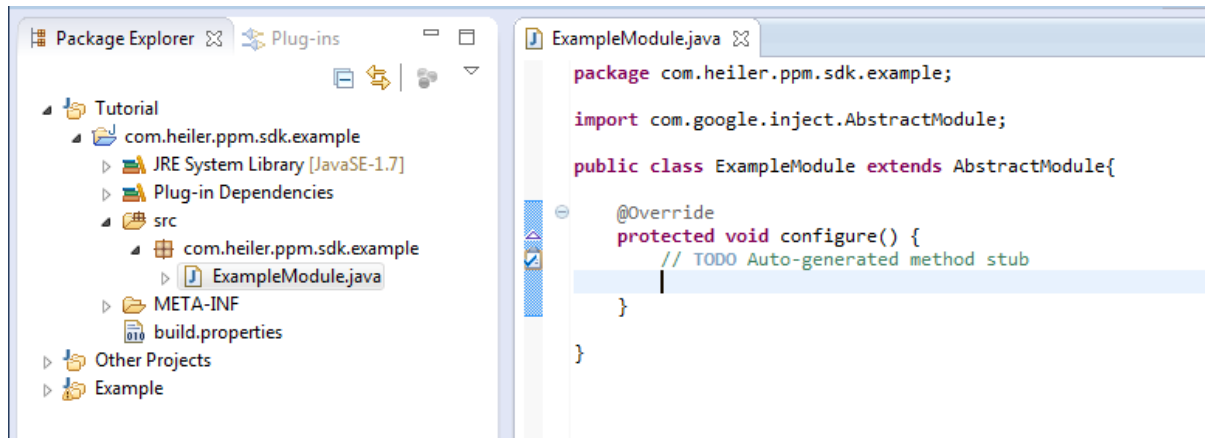
com.heiler.ppm.web.integration (1.0.0)	Add... Remove Up Down Properties...
com.heiler.ppm.web.common (1.0.0)	
com.google.inject (3.0.0)	

Total: 3

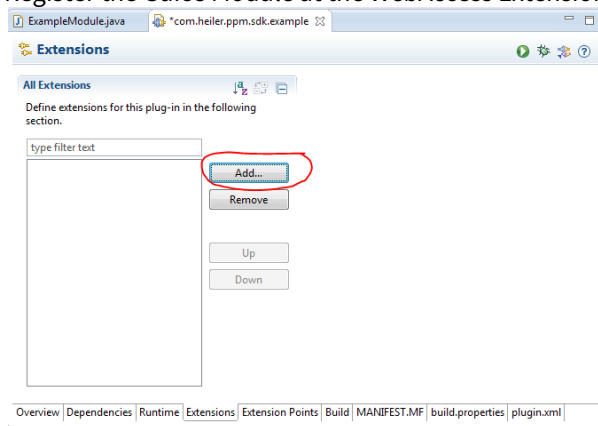
Automated Management of Dependencies [ja](#) [Z](#)

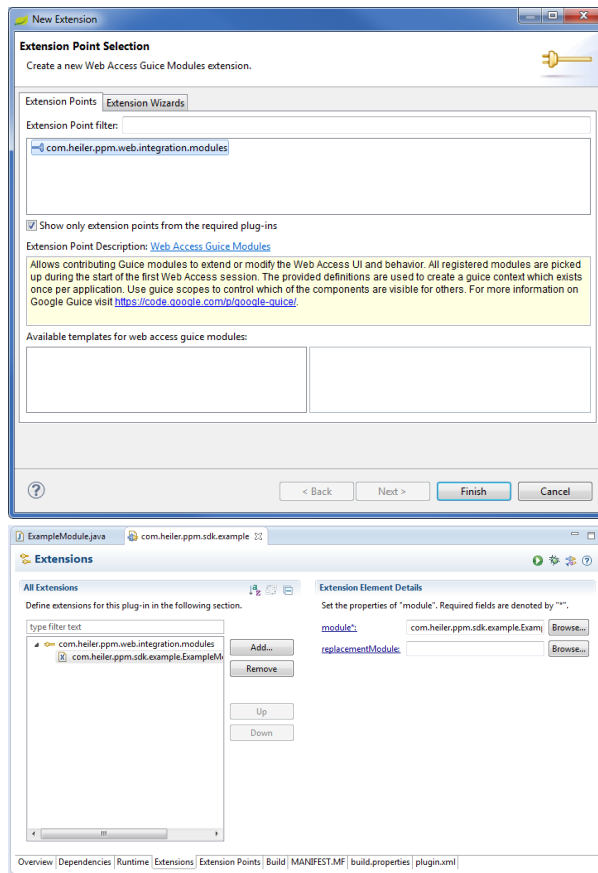
[Overview](#) [Dependencies](#) [Runtime](#) [Extensions](#) [Extension Points](#) [Build](#) [MANIFEST.MF](#) [bu](#)

- Create a new Guice module



- Register the Guice Module at the WebAccess ExtensionPoint

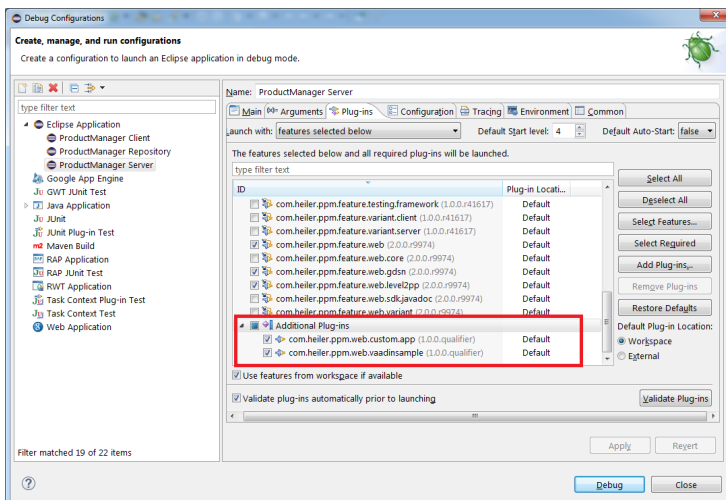




That's it. The ExampleModule class is the starting point for your customizations.

8.6.1.2 Launch customized application

Before launching the Web UI with the Application Server, all new bundles need to be added to the launch configuration.



8.6.1.3 Extending existing modules

It is often desired to replace or even hide existing functionality from existing components. This can be achieved by specifying a ReplacementModule class at the Integration Extension Point. In this case, the referenced module is replaced with the new contributed module class. Normally you want to extend the existing module and override the methods to apply your changes.

```

* Copyright (c) 2002 Heiler Software AG.

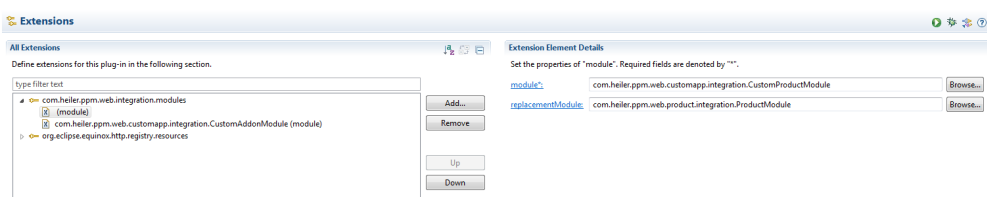
package com.heiler.ppm.web.customapp.integration;

import com.google.inject.CreationException;

/**
 * This custom module is used as a replacement of the {@link ProductModule}.
 * This can be useful to replace or disable default contributions.
 *
 * @author sroeck
 */
public class CustomProductModule extends ProductModule
{
    @Override
    protected void configure() throws CreationException
    {
        super.configure();
    }
}

```

The contribution looks like this:



8.6.2 Typical Use Cases



Working code snippets for the examples shown below can also be found as within the example project **com.heiler.ppm.web.custom.app** which is part of the SDK distribution package. There you can also find a growing number of additional snippets that illustrate how specific parts of the application can be extended.

8.6.2.1 Add a custom tab in the detail area

There are two types of tabs to contribute:

- A tab contains only fields that can be build using a FieldFormDefinition. In this case, you don't have to write JavaCode but can extend the XML definitions instead.
- A tab contains custom content that cannot be generated generically.

For the latter case, a new tab can be implemented by inheriting from *AbstractEntityDetailTab*. The following example renders a Browser widget in the tab and updates the label based on the selected item.

[Click here for full source code](#)

```
public class CustomItemDetailTab extends AbstractEntityDetailTab
{
    private static final long serialVersionUID = 1L;
    private VerticalLayout    rootLayout;
    private Embedded         browser;
    public CustomItemDetailTab()
    {
        this.rootLayout = new VerticalLayout();
        this.rootLayout.setMargin( true );
        this.rootLayout.setSizeFull();
        setCompositionRoot( this.rootLayout );
        setSizeFull();
        setCaption( "Custom detail tab" );
    }
    @Override
    public void update( EntityProxy model, boolean force ) throws CoreException
    {
        if ( model != null )
        {
            setDetailModel( model.getDetailModel( LoadHint.EVERYTHING ), force );
            this.browser.setCaption( "Latest news for " + this.detailModel.getDisplayLabelLongNoError() );
        }
    }
    @Override
    public String getId()
    {
        return ""; // empty id because no permissions should be checked
    }
    @Override
    public int getPosition()
    {
        return Positions.DetailTabs.Article.Media + 1;
    }
    @Override
    protected boolean buildMainLayout()
    {

```

```

    this.browser = new Embedded( "", new ExternalResource( "http://www.heiler.com"
) );
    this.browser.setType( Embedded.TYPE_BROWSER );
    this.browser.setSizeFull();
    this.rootLayout.addComponent( this.browser );
    return true; // return true to make sure the widgets are only rendered once
}
}

```

Register the new Tab implementation in your ExampleModule using the following binder:

```

private void contributeItemDetailTab()
{
    Multibinder< DetailTab< EntityProxy > > detailBinder =
    Multibinder.newSetBinder( binder(),

    new TypeLiteral< DetailTab< EntityProxy >>()

    { /*nothing is required here*/

    },

    Names.named( Components.DetailTabs.ARTICLE ) );
    detailBinder.addBinding()
        .to( CustomItemDetailTab.class );
}

```

8.6.2.2 Add a custom context area

Similar to custom tabs, the Web UI navigation area on the left can also be extended programmatically. In this case, the base class to inherit from is *AccordionTreeItem*. A sample implementation that provides a link to open a new Browser Window looks like this:

[Click here for full source code](#)

```

public class CustomContextItem extends AccordionTreeItem
{
    private static final long serialVersionUID = 1L;
    public CustomContextItem()
    {
        setCaption( "Custom Context Tab" );
    }
    @Override
    public String getId()
    {
        return getClass().getName();
    }
}

```

```

@Override
public int getPosition()
{
    return Positions.Accordion.Catalogs + 1;
}
@Override
protected void buildMainLayout()
{
    HorizontalLayout rootLayout = new HorizontalLayout();
    rootLayout.setMargin( true );
    Label customWidget = new Label( "Open External Page: " );
    rootLayout.addComponent( customWidget );
    rootLayout.addComponent( new Link( "Click me", new ExternalResource( "http://
www.google.com" ), "_blank", 0, 0, 0 ) );
    setCompositionRoot( rootLayout );
}
@Override
protected void initTree()
{
    // do initialization here
}
@Override
protected String getCaptionPropertyId()
{
    return null;
}
@Override
public boolean hasReadPermission()
{
    return true;
}
@Override
public String getPermissionId()
{
    // return id of action right if permission checks are desired
    return "";
}
}

```

Register the new Context tab implementation in your ExampleModule using the following binder:

```

private void contributeAccordionItem()
{
    Multibinder< AccordionTreeItem > accordionTabBinder =
Multibinder.newSetBinder( binder(),

AccordionTreeItem.class,

Names.named( Components.Accordions.MAIN ) );
    accordionTabBinder.addBinding()
        .to( CustomContextItem.class );
}

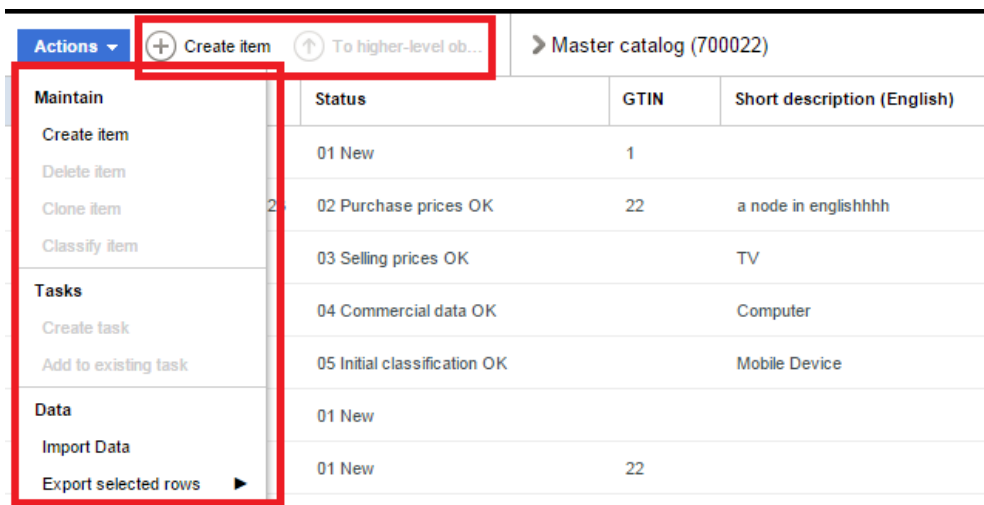
```

```
}

```

8.6.2.3 Add a custom action to the table menu bar

The action bar above all main tables can be extended too. Therefore, provide an implementation that inherits from MenuItemWrapper. You need to provide caption, position, group identifier (if used in action menu) and a command callback that contains the logic. Also you can make item invisible by default in action toolbar or added to favorite area.



The following example simply counts the number of all selected products:

```
public class CustomProductListMenuItem extends MenuItemWrapper
{
    private static final String MENU_IDENTIFIER = "menu.custom.product.list.count";
    @Inject
    MasterListViewProvider listViewProvider;
    @Inject
    Window mainWindow;
    public CustomProductListMenuItem()
    {
        super( MENU_IDENTIFIER );
        this.menuPath = "?after=" + Components.Menu.Product.ACTION_DELETE;
        this.caption = "Count";
        this.setGroupIdentifier( CustomMenuItemGroup.ID );
        this.command = new Command()
        {
            private static final long serialVersionUID = 1L;
            @Override

```

```

    public void menuSelected( MenuItem selectedItem )
    {
        openPopup();
    }
};
}
private void openPopup()
{
    ManagedListView productListView = getProductListView();
    Set< ListEntry > selection = productListView.getValues();
    UI.getCurrent()
        .showNotification( "You have selected " + selection.size() + " products." );
}
private ManagedListView getProductListView()
{
    return this.listViewProvider.get( Product2G._identifier );
}
}

```

Register the new menu item in the ExampleModule. Make sure to use the correct component name from Components.Menu to select the product list.

```

private void contributeProductListMenuItem()
{
    Multibinder< MenuItemWrapper > productMenuItemBinder =
getMenuItemBinder( Components.Menu.PRODUCT_LIST_ACTIONS );
    productMenuItemBinder.addBinding()
        .to( CustomProductListMenuItem.class );
}

```

Please note that a registered menu item is displayed for lists in different contexts, e.g. when displaying items of a catalog, structure or after running a search. To restrict the visibility of a menu item to a specific context, the MenuItem implementation can register a listener at the underlying ManagedListView and toggle the visibility whenever the data of the list are updated. Sample:

```

getProductListView().addListener( new ListView.DataUpdateListener() {
    private static final long serialVersionUID = 1L;
    @Override
    public void onDataUpdate( DataUpdateEvent event )
    {
        // Hide menu when list shows search results
        boolean isVisible = event.getData() instanceof ListViewChildrenData;
        setVisible( isVisible );
    }
});

```


You also can provide new menu groups by extending MenuItemGroup class and defining identifier, i18nKey and order weight (optional):

```
public class CustomMenuItemGroup extends MenuItemGroup
{
    public static final String ID = "menu.item.group.custom"; //$NON-NLS-1$
    public NavigationMenuItemGroup()
    {
        super( ID, "%web.common.menu.item.group.custom", 300 ); //$NON-NLS-1$
    }
}
```

Register it in module. Make sure to use the correct component name from Components.Menu to select the product list.

```
Multibinder< MenuItemGroup > menuItemGroupBinder =
Multibinder.newSetBinder( binder(), MenuItemGroup.class,

Names.named( Components.Menu.PRODUCT_LIST_ACTIONS ) );
menuItemGroupBinder.addBinding()
    .to( CustomMenuItemGroup.class );
```

If you creating new menu for your own list and want to enable xml customization, just add menu identifier to string binding:

```
Multibinder< String > actionMenus = Multibinder.newSetBinder( binder(), String.class,

Names.named( Components.Menu.ACTION_MENU_FILTER_QUALIFIER ) );
actionMenus.addBinding()
    .toInstance( Components.Menu.YOUR_MENU_IDENTIFIER );
```

8.6.2.4 Implementing and contributing custom flex components for dashboard

It is possible to define customized dashboard components which may then be usable in every dashboard. The CustomDashboardComponent and CustomDashboardComponentFactory classes show how to implement own dashboard components.

The type attribute in a component element inside a dashboard definition defines which component to use. The final contribution is done in CustomAddonModule#bindComponentFactory().

```
Multibinder< ComponentFlexComponentFactory > componentFactories =
Multibinder.newSetBinder( binder(),

ComponentFlexComponentFactory.class );
componentFactories.addBinding()
    .to( CustomDashboardComponentFactory.class );
```

8.6.2.5 Contribute Custom Placeholder for XML Configuration Files

Since Product 360 8.0.03 placeholders can be used in XML configuration files to inject context specific values. For example, `${user.language}` refers to the UI language that the user selected for log in.

It is also possible, to contribute custom logic to add new placeholders. The following example shows, how to define a placeholder `${user.targetMarket}`. The value is derived at runtime based on logged in user properties in the new class `TargetMarketLKDefinitionPlaceholderHandler`. The placeholder can be used for all logical keys that are based on the enumeration `Enum.Territory`.

```
import com.google.inject.Inject;
import com.heiler.ppm.acl.commons.principal.Group;
import com.heiler.ppm.security.core.Impersonator;
import com.heiler.ppm.security.core.LoginToken;
import
com.heiler.ppm.web.common.definition.placeholder.LKDefinitionPlaceholderHandler;
/**
 * @author sroeck
 */
@SuppressWarnings( "nls" )
public class TargetMarketLKDefinitionPlaceholderHandler implements
LKDefinitionPlaceholderHandler
{
    @Inject
    Impersonator impersonator;

    @Override
    public boolean handles( String lkName )
    {
        return lkName.endsWith( ".LK.TargetMarket" );
    }
    @Override
    public String getUIPlaceholderRepresentation()
    {
        return "${user.targetMarket}";
    }
    @Override
    public Object handle( Object stub, String lkName )
    {
        String result = "US"; // Default Target Market
        LoginToken loginToken = this.impersonator.getLoginToken();
        if ( !loginToken.isGuest() )
        {
            Group[] groups = loginToken.getGroups();
            if ( groups.length > 0 )
            {
                // TODO: Extract key for Enum.Territory from user group
                result = "MX";
            }
        }
        return result;
    }
}
```

```

    }
}

```

To contribute the custom handler, add the following method to the module class:

```

private void bindCustomPlaceholder()
{
    MapBinder< String, LKDefinitionPlaceholderHandler > handlers =
    MapBinder.newMapBinder( binder(),

String.class,

LKDefinitionPlaceholderHandler.class,

Names.named( Components.PlaceholderHandlers.LKValuePlaceholderHandlers ) );
    handlers.addBinding( "${user.targetMarket}" )
        .to( TargetMarketLKDefinitionPlaceholderHandler.class );
}

```

The new variable can now be used either in a programmatic definition of forms or list definitions or directly within the XML configuration.

Example:

```

<definition debugId="article_gdsn_targetmarket_tab" i18NKey="%web.article.detail.
tab.targetMarket" permissionId="web.article.detail.tab.targetMarket" position="2"
rootEntity="Article">
    <column>
        <fieldGroup displaySectionWidget="true" subEntityId="GDSNTargetMarketExte
nsion">
            <field identifier="GDSNTargetMarketExtension.IsActiveInMarket"/>
            <field identifier="GDSNTargetMarketExtension.IsOrderableUnit"/>
            <field identifier="GDSNTargetMarketExtension.IsDispatchUnit"/>
            <field identifier="GDSNTargetMarketExtension.IsInvoiceUnit"/>
            <field identifier="GDSNTargetMarketExtension.IsPackagingMarkedReturna
ble"/>
            <field identifier="GDSNTargetMarketExtension.StartAvailabilityDate"/>
            <field identifier="GDSNTargetMarketExtension.EndAvailabilityDate"/>
            <field identifier="GDSNTargetMarketExtension.DiscontinuedDate"/>
            <logicalKey identifier="ArticleMarketExtensionType.LK.TargetMarket"
selectable="true" value="${user.targetMarket}"/>
        </fieldGroup>
    </column>
</definition>

```

8.6.2.6 Hide or show detail tabs dependent of arbitrary state of selected object

- `com.heiler.ppm.web.common.view.detail.VetoTabStrategy` interface was created. It has a single method `< T > boolean veto(T model, DetailTab< T > detailTab);`
- All guice-bound implementations of `VetoTabStrategy` will be called on the selected item change and the selected tab change.
- If one of the strategies returns `true` as the result of `veto()` method call, the tab will be hidden.
- Use `detailTab.getId()` to identify tabs. For the web definition tabs `getId()` returns the value specified as the `debugId` in the definition. For example, `product.detailtab.xml` contains a tab definition with the following `debugId="product_attributes_tab"`.
- Identificators for the tabs of other types are not well-defined.

Example - Hide Price tab for products when the status field is "01 New":

```
public class CustomVetoTabStrategy implements VetoTabStrategy
{
    private static final String PRICES_TAB_DEBUG_ID = "detail_selling_prices";
    private static final int STATUS_NEW = 100;
    private static final Log LOG = LogFactory.getLog( CustomVetoTabStrategy.class );

    @Inject
    RepositoryService repositoryService;

    private FieldPath statusfieldPath;

    @PostConstruct
    public void postConstruct()
    {
        this.statusfieldPath =
        RepositoryUtils.getFieldPath( repositoryService.getFieldByIdentifier( Product2G.CurrentStatus ) );
    }

    @Override
    public < T > boolean veto( T model, DetailTab< T > detailTab )
    {
        boolean result = false;
        // Check selected tab first before loading data
        if ( isProductSelected( model ) && isPricesTab( detailTab ) )
        {
            try
            {
                EntityProxy proxy = ( ( EntityProxy ) model );
                EntityType entityType = proxy.getEntityType();
                LoadHint loadHint = new LoadHintBuilder( entityType ).build();
                EntityDetailModel detailModel = proxy.getDetailModel( loadHint );
                Integer status = ( Integer ) detailModel.getFieldValue( statusfieldPath );
                if ( status != null && status.intValue() > STATUS_NEW )
                {

```

```

        // Hide Tab for status other than NEW
        result = true;
    }
}
catch ( CoreException e )
{
    LOG.warn( "Cannot check status field for tab visibility. Returning false to
show tab anyway.", e );
}
}
return result;
}

private boolean isPricesTab( DetailTab<?> detailTab )
{
    return PRICES_TAB_DEBUG_ID.equals( detailTab.getId() );
}

private < T > boolean isProductSelected( T model )
{
    String entityIdIdentifier = null;
    if ( model instanceof EntityProxy )
    {
        EntityProxy proxy = ( EntityProxy ) model;
        entityIdIdentifier = proxy.getEntity().getIdentifer();
    }
    return Product2G._identifer.equals( entityIdIdentifier );
}
}

```

9 Web Search



The sub pages describes supported repository customizing needed by Product 360 Web Search.

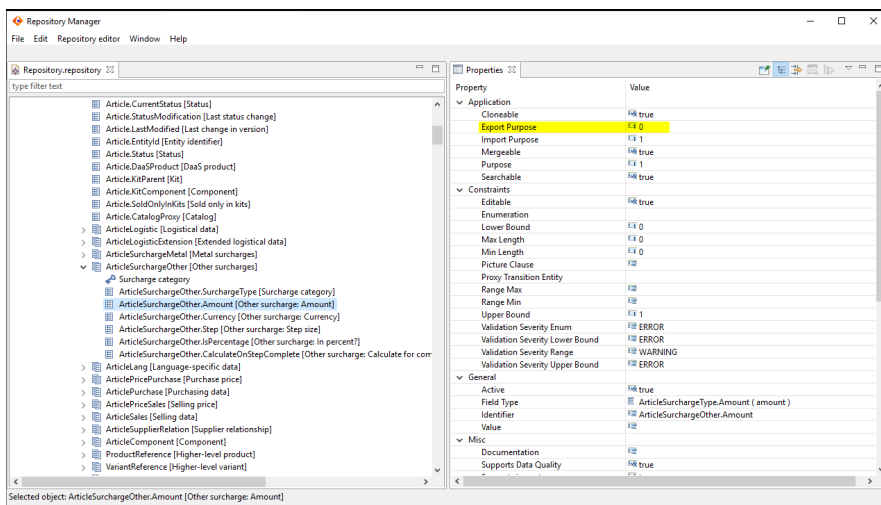
Sub pages:

- [Repository Customizing \(see page 362\)](#)

9.1 Repository Customizing

Currently all that can be exported, can be indexed on elastic search server with exception of characteristics.

In order to make a field available to be indexed, obviously the field should be made available for export. This can be done using the Export purpose property available in the repository for any field. As per the use case, the purpose must be set.



10 Supplier Portal

10.1 Overview Customizing

PIM Supplier Portal is a Web Application that can be configured and customized in many different ways. For further information about installation and configuration see the configuration guide.

Many customizations can be done without writing code with the help of JSON configuration files. There's also a Remote API for integration with other systems.

Please follow the links in the table to jump to a specific customization option.

Customization Option	Description
Mail Templates (see page 365)	Change the e-mails that are send by the portal to suppliers and portal users.
Styling (CSS) (see page 365)	Change the design e.g. fonts, colors and images. Also includes customization of the application logo.
Customer landing pages (imprint, support box, etc.) (see page 365)	Change static pages like login, registration, imprint, terms and conditions and others. Also includes the application page title and favicon displayed in the browser bar.

Customization Option	Description
Application Navigation Bar (since 7.0.4) (see page 373)	Extend the application navigation menu for suppliers and/or portal user, e.g. to integrate external content.
Workflow Definitions (since 7.0.4) (see page 383)	Extend or modify the system behavior for registration, invitation and data upload workflows.
Remote API (since 7.1.1) (see page 399)	Integrate PIM Supplier Portal into other applications. Can be used for supplier data batch upload during system setup.
Supplier Portal REST Interface (see page 386)	Interact with dedicated Supplier Portal resources by using a REST API. This includes posting messages to the supplier's timeline, for instance.
Customized Data Model for Supplier Data (since 7.1.1) (see page 405)	Store additional data for suppliers and allow them to maintain them as self-service.

10.2 Subpages

- [Styling and E-Mail Templates \(see page 365\)](#)
- [Customize User Interface \(see page 373\)](#)
- [Manage and Customize Workflows \(see page 383\)](#)
- [Supplier Portal REST Interface \(see page 386\)](#)
- [Supplier Portal Remote API \(see page 399\)](#)
- [Supplier Portal I18n \(see page 404\)](#)
- [Customized Data Model for Supplier Data \(see page 405\)](#)

10.3 High Level Architecture

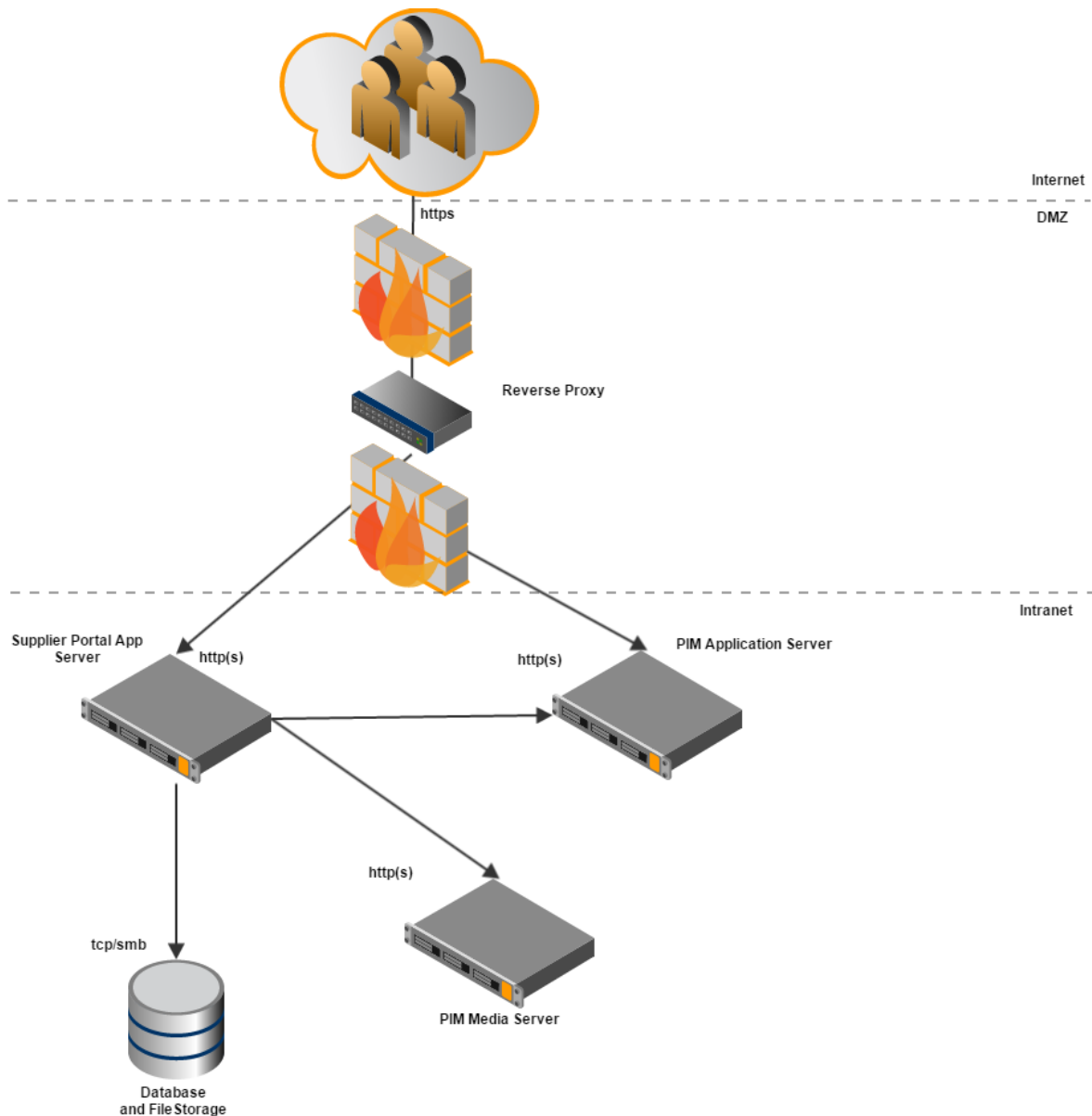
PIM Supplier Portal is a Web Application that is deployed in a JEE web container (Apache Tomcat). The communicates via http with both the PIM Application Server and PIM Media Manager following REST principles. It also comes with a persistence layer that holds supplier user specific data, e.g. the timeline communication feeds.

The application scales horizontally by setting up a Web Loadbalancer, e.g. Apache Webserver.

The application is build on top of these frameworks:

- JPA2 (Hibernate)
- Spring (Dependency Injection)
- Spring Security (Authentication, Authorization)

- Jersey (REST)
- GWT (UI Toolkit)
- Activiti (Workflow Engine)



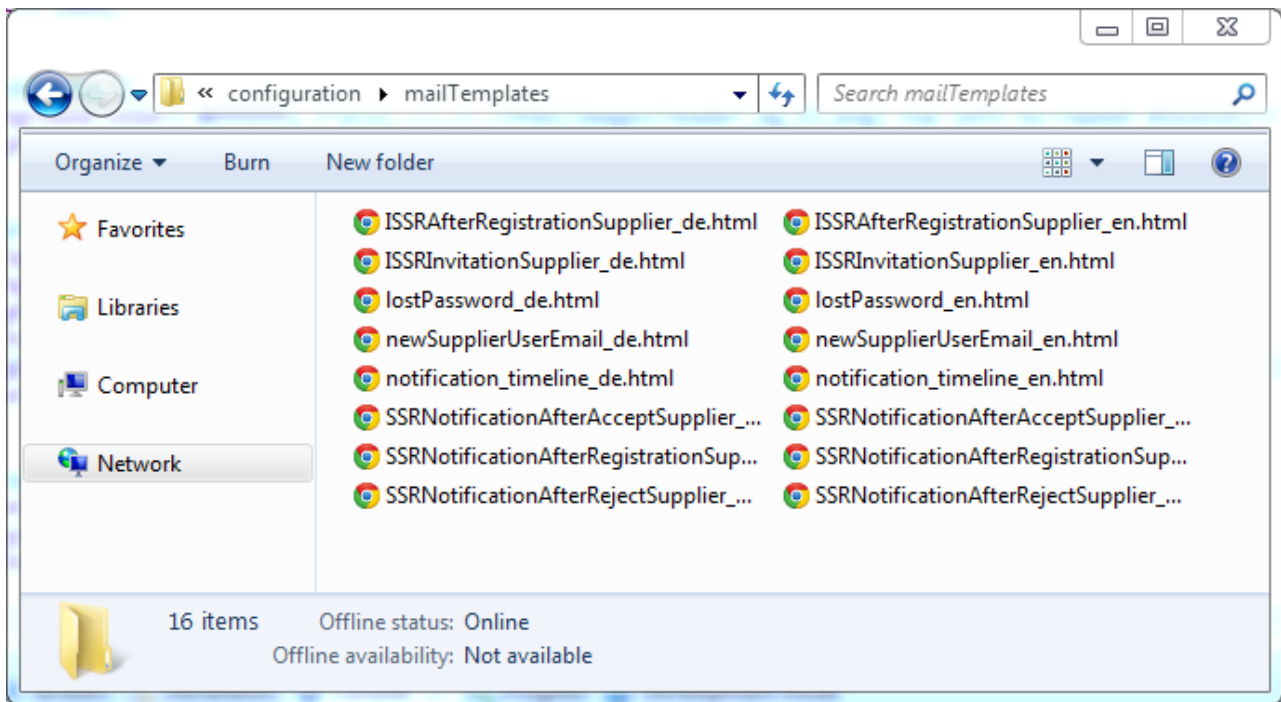
10.4 Styling and E-Mail Templates

- [Customizing Mail Templates](#) (see page 366)
 - [List of all Templates](#) (see page 366)
- [Customizing Static HTML Files, e.g. Imprint](#) (see page 369)
 - [List of Static HTML Files](#) (see page 370)
- [Customizing Styles \(CSS\)](#) (see page 371)
 - [Example](#) (see page 371)
- [Build and Deploy Customizings](#) (see page 372)

10.4.1 Customizing Mail Templates

As described in the Supplier Portal Configuration guide, users can specify a directory from which the mail templates are loaded. The default location is

/configuration/mailTemplates



All default templates are HTML files that are provided in German and English. If desired more languages can be added by providing a file with the corresponding file name suffix.

Download all templates.

10.4.1.1 List of all Templates

Most of the templates contain several variables in the notation **\${variableName}**. Variables will be replaced with context specific values during mail generation.

Filename	Purpose	Recipient	Supported Variables
ISSRInvitationSupplier	Invitation mail to suppliers by portal administrator	supplier admin	\${portalUrl}

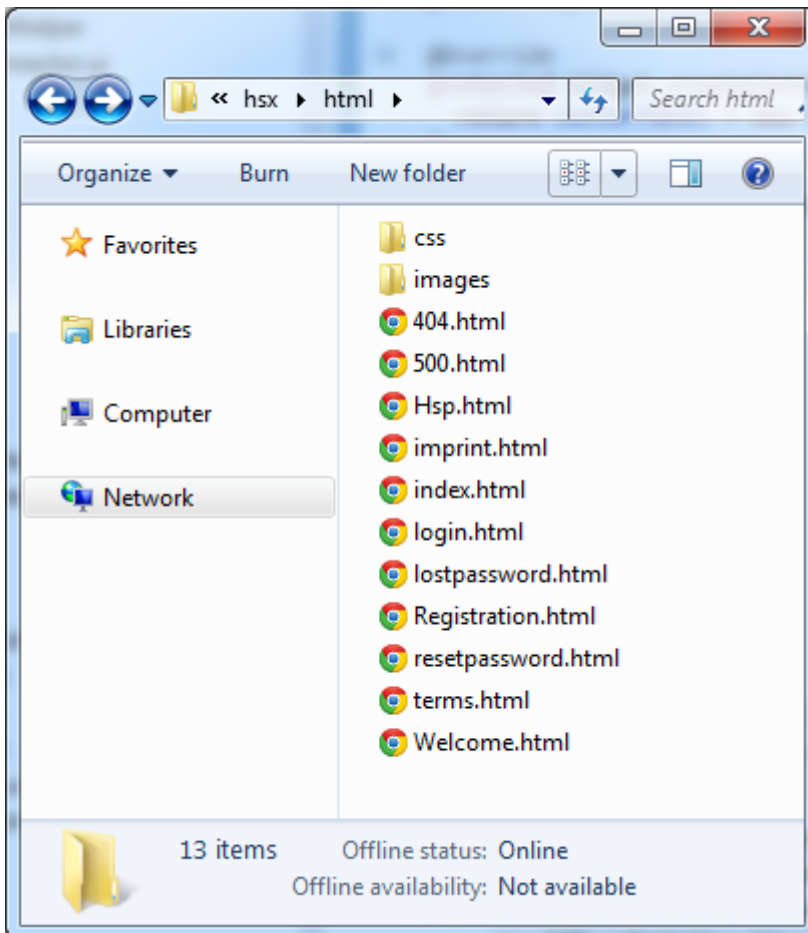
Filename	Purpose	Recipient	Supported Variables
ISSRAfterRegistrationSupplier	Confirmation mail to suppliers after they accepted an invitation mail and set a password	supplier admin	\$ {supplier.firstName} \$ {supplier.lastName} \${supplier.email} \$ {supplier.loginName} \$ {supplier.displayName} e} \${loginPage}
SSRNotificationAfterRegistrationSupplier	Mail after self service registration	supplier admin	\$ {supplier.firstName} \$ {supplier.lastName} \${supplier.email} \$ {supplier.loginName} \$ {supplier.displayName} e}
SSRNotificationAfterAcceptSupplier	Mail after self service acceptance	supplier admin	\$ {supplier.firstName} \$ {supplier.lastName} \${supplier.email} \$ {supplier.loginName} \$ {supplier.displayName} e} \${loginPage}

Filename	Purpose	Recipient	Supported Variables
SSRNotificationAfterRejectSupplier	Mail after self service reject	supplier admin	\$ {supplier.firstName} \$ {supplier.lastName} \${supplier.email} \$ {supplier.loginName} \$ {supplier.displayName}
newSupplierUserEmail	Supplier admin creates a new user	supplier user	\$ {supplier.firstName} \$ {supplier.lastName} \${supplier.email} \$ {supplier.loginName} \$ {supplier.displayName} \${portalUrl}
lostPassword	Supplier admin or user requested to reset their password	supplier user / admin	\$ {supplier.firstName} \$ {supplier.lastName} \${supplier.email} \$ {supplier.loginName} \$ {supplier.displayName} \${portalUrl}

Filename	Purpose	Recipient	Supported Variables
notification_timeline	Notification of a new timeline message	supplier user/ admin portal administrator	<code>\${message.subject}</code> <code>\$</code> <code>{message.username}</code> <code>}</code> <code>\${message.text}</code> <code>\${message.email}</code> <code>\${message.author}</code> <code>\${message.url}</code>

10.4.2 Customizing Static HTML Files, e.g. Imprint

The Supplier Portal is a dynamically created rich internet application that is build upon only few static html pages. These pages define a basic page skeleton with document title, background, favicon, etc.



To customize these pages follow these steps:

1. Unzip the file `/tomcat/webapps/hsx/hsx.war` into a temporary directory
2. Navigate to `/hsx/html` and select the html file to customize

3. After applying the desired changes zip the war file again and redeploy the application in Tomcat.

10.4.2.1 List of Static HTML Files

File Name	Purpose
login.html	Login page that is shown to unauthenticated user (both supplier and portal administrators)
Hsp.html	Application main page that is shown after successful authentication (both supplier and portal administrators)
Registration.html	Registration form for suppliers.
terms.html	Terms of use template. Is empty by default should be customized for production.
imprint.html	Company imprint. Is empty by default should be customized for production.
lostpassword.html	Page to request a password reset. Suppliers only.
resetpassword.html	Page to set up an initial password or to assign a new password after password reset request. Suppliers only.
index.html	Performs a client-side redirect to login.html. Normally no customization needed.
404.html	404 error page when accessing an unknown url.
500.html	500 error page in case of an severe internal server error.

During customization of the application some of the links may not be required any longer. F.e. if the registration process is heavily customized and is not done via Supplier Portal directly you may want to remove some of the links concerning registration in the terms page.

10.4.3 Customizing Styles (CSS)

You can change the appearance of the supplier portal by re-placing the default CSS styles with custom styles. This can be done on css-class level.

```
<link rel="Shortcut icon" href="../../favicon.ico">
<link type="text/css" rel="stylesheet" href="../../ext/ext/css/ext-all.css">
<link type="text/css" rel="stylesheet" href="../../compiled/theme/css/default.css">
<!--
  <link type="text/css" rel="stylesheet" href="default.css"/>
-->
<script type="text/javascript" language="javascript">
  init( document );
</script>
<meta name="gwt:property" content="locale=en_US">
<link type="text/css" rel="stylesheet" href="css/custom.css">
</head>
```

After installing PIM Supplier Portal, the css file for customized styles can be found here:

<SupplierInstallationRoot>\tomcat\webapps\hsx\html\css\custom.css

It is recommended to use Browser inspection (Chrome, Firefox) to identify the elements in DOM and their css classes. Most browsers support to change the styles directly to verify the style customization before putting it into the custom.css.

10.4.3.1 Example

To customize the color scheme of the login screen, the follow css can be used:

```
/**
Change login box background and font colors.
*/
.hsx-content-area{
    background-color: magenta;
}
.hsx-content-area a, .hsx-content-area div {
    color: white;
}
```

Result:

You can also add additional images and reference them in the custom.css. This might be useful to replace the application logo for instance.

10.4.4 Build and Deploy Customizings

All changes inside the war file structure of Supplier Portal need to be re-packaged and deployed. The includes changes to styling and static html pages as described above.

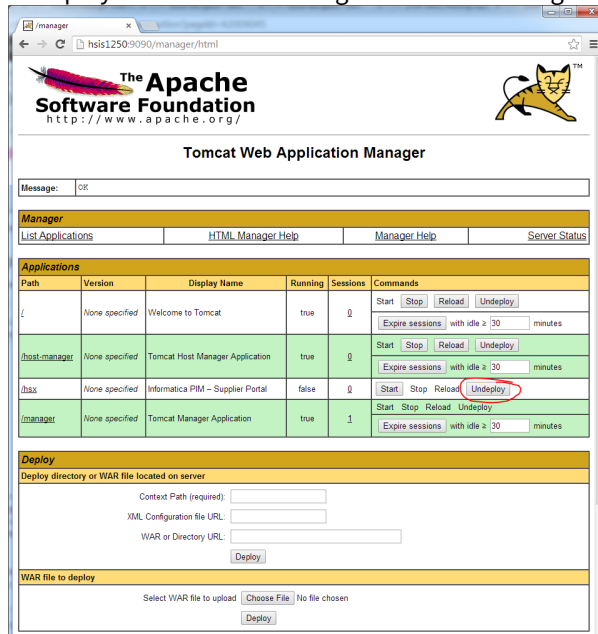
Best practice steps are:

1. Unzip war file into an empty directory
2. Add files that need to be changed to SVN (or any other versioning tool of your choice). In most cases this will be the folder **/html**.

Name	Date modified	Type	Size
compiled	19.12.2013 14:18	File folder	
ext	19.12.2013 14:18	File folder	
help	19.12.2013 14:18	File folder	
html	18.03.2014 10:40	File folder	
META-INF	19.12.2013 14:18	File folder	
WEB-INF	19.12.2013 14:18	File folder	
favicon.ico	16.12.2013 14:10	Icon	5 KB
index.html	16.12.2013 14:10	Chrome HTML Do...	1 KB

3. Apply changes to extracted files
4. Repackage by zipping the content and renaming the zip file to a war file again, e.g. by using 7zip.

5. Undeploy the old war file using the Tomcat manager application (e.g. <http://localhost:9090/manager/html>)



6. Deploy new war file dropping the file into the tomcat/webapps folder

Please do not:

- Change the content of /tomcat/webapps/hsx directly. All changes will be lost whenever the war file is redeployed
- Change anything in /tomcat/work or /tomcat/temp. These are internal folders of tomcat, the results might be non-deterministic. Changes might be lost after Tomcat restart.

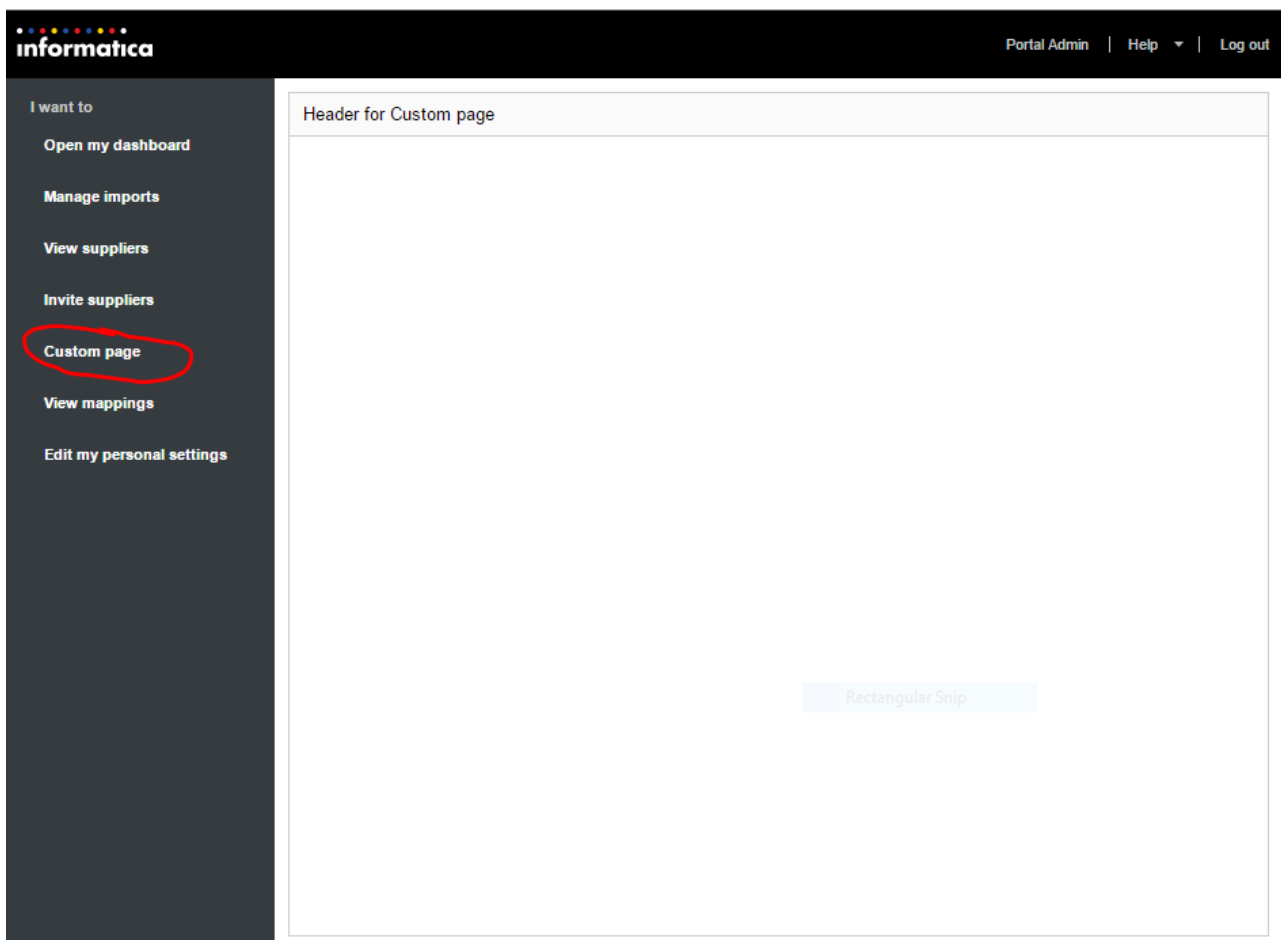
10.5 Customize User Interface

- [Customize Application Navigation Bar](#) (see page 373)
 - [Add new element to Navigation Bar](#) (see page 374)
- [User Permissions control available list of Actions](#) (see page 379)
- [Supplier Portal as embedded application](#) (see page 380)
 - [Display as embedded application](#) (see page 381)
 - [Initialize with a specific locale](#) (see page 381)
 - [Deep links to the application](#) (see page 381)
- [Add custom pages to the support box](#) (see page 382)

10.5.1 Customize Application Navigation Bar

The navigation bar is used to access different areas of the application. The displayed elements depend on the logged-in user's role (i.e. Supplier User, Supplier Admin, Portal Admin).

New elements can be configured to be added to the bar. The main use case is to easily integrate external content or external systems.



10.5.1.1 Add new element to Navigation Bar

Open the configuration file <INSTALLATION_ROOT>/configuration/uiCustomization.json. An example navigation bar contribution looks like this:

```

1  {
2    "customNavigation":[
3      {
4        "position":50,
5        "actionCaptions":[
6          {
7            "locale":"en_US",
8            "value":"Custom page"
9          },
10         {
11           "locale":"de_DE",
12           "value":"Beispielseite"
13         }
14       ],
15       "headerCaptions":[
16         {

```

```

17         "locale":"en_US",
18         "value":"Header for Custom page"
19     },
20     {
21         "locale":"de_DE",
22         "value":"Titel für Beispielseite"
23     }
24 ],
25     "userRoles":["ROLE_SUPPLIER_ADMIN", "ROLE_SUPPLIER_USER",
26     "ROLE_PORTAL_ADMIN"],
27     "url_":"/customPage.html?supplierId=${organization.hpmSupplierId}
28     &locale=${user.uiLocaleAsString}"
29 }

```

Allowed values for the json attributes are described below:

Json Attribute	Description
customNavigation	Contains a json array whereas each element describes a navigation bar element.

Json Attribute	Description																								
position	<p>Relative position of the contribute element. Elements are ordered by position number ascending. The default element have the following position indices:</p> <table> <tr> <th>Position</th><th>Supplier Action Caption</th></tr> <tr> <td>10</td><td>Open my dashboard</td></tr> <tr> <td>20</td><td>Upload to catalog</td></tr> <tr> <td>30</td><td>View my organization (<i>supplier admins only</i>)</td></tr> <tr> <td>120</td><td>Edit my personal settings</td></tr> <tr> <th>Position</th><th>Portal Admin Caption</th></tr> <tr> <td>10</td><td>Open my dashboard</td></tr> <tr> <td>20</td><td>Manage Imports</td></tr> <tr> <td>30</td><td>View suppliers</td></tr> <tr> <td>50</td><td>Invite suppliers</td></tr> <tr> <td>60</td><td>View mappings</td></tr> <tr> <td>120</td><td>Edit my personal settings</td></tr> </table>	Position	Supplier Action Caption	10	Open my dashboard	20	Upload to catalog	30	View my organization (<i>supplier admins only</i>)	120	Edit my personal settings	Position	Portal Admin Caption	10	Open my dashboard	20	Manage Imports	30	View suppliers	50	Invite suppliers	60	View mappings	120	Edit my personal settings
Position	Supplier Action Caption																								
10	Open my dashboard																								
20	Upload to catalog																								
30	View my organization (<i>supplier admins only</i>)																								
120	Edit my personal settings																								
Position	Portal Admin Caption																								
10	Open my dashboard																								
20	Manage Imports																								
30	View suppliers																								
50	Invite suppliers																								
60	View mappings																								
120	Edit my personal settings																								

Json Attribute	Description
actionCaptions	<p>The caption of the contribute navigation bar element.</p> <ul style="list-style-type: none"> • locale - Java-Locale to specify the caption language E.g. de_DE, en_US. • value - localized caption
headerCaption	<p>The caption of the header bar on top of the content element. Syntax is the same as for actionCaptions. If empty, no header bar will be displayed.</p>
userRoles	<p>A json array specifying the user roles, for which an element should be shown. Possible values are any combination of:</p> <ul style="list-style-type: none"> • ROLE_SUPPLIER_ADMIN • ROLE_SUPPLIER_USER • ROLE_PORTAL_ADMIN • ROLE_BROKER_USER

Json Attribute	Description																				
url	<p>An absolute or relative url that is used for loading the iFrame content. Supports variables in the syntax of \${variableName}. Supported variables:</p> <table> <tr> <th>Variable</th><th>Description</th></tr> <tr> <td>user.loginName</td><td>Value that has been used for log in to Supplier Portal. Equals email address for suppliers.</td></tr> <tr> <td>user.uiLocaleAsString</td><td>Java Locale specifying the user's ui language. E.g. de_DE, en_US.</td></tr> <tr> <td>user.lastName</td><td>User last name.</td></tr> <tr> <td>user.firstName</td><td>User first name</td></tr> <tr> <td>user.id</td><td>Internal database id of user.</td></tr> <tr> <td>user.email</td><td>User email address. Never empty for suppliers, potentially empty for portal administrators.</td></tr> <tr> <td>user.stateAsString</td><td>One of ACTIVE, REGISTERED, INVITED, NOT_INVITED, DEACTIVATED</td></tr> <tr> <td>user.isSupplierAdmin</td><td>True, if the user is supplier administrator.</td></tr> <tr> <td>user.isPortalUser</td><td>True, if the user is a portal administrator.</td></tr> </table>	Variable	Description	user.loginName	Value that has been used for log in to Supplier Portal. Equals email address for suppliers.	user.uiLocaleAsString	Java Locale specifying the user's ui language. E.g. de_DE, en_US.	user.lastName	User last name.	user.firstName	User first name	user.id	Internal database id of user.	user.email	User email address. Never empty for suppliers, potentially empty for portal administrators.	user.stateAsString	One of ACTIVE, REGISTERED, INVITED, NOT_INVITED, DEACTIVATED	user.isSupplierAdmin	True, if the user is supplier administrator.	user.isPortalUser	True, if the user is a portal administrator.
Variable	Description																				
user.loginName	Value that has been used for log in to Supplier Portal. Equals email address for suppliers.																				
user.uiLocaleAsString	Java Locale specifying the user's ui language. E.g. de_DE, en_US.																				
user.lastName	User last name.																				
user.firstName	User first name																				
user.id	Internal database id of user.																				
user.email	User email address. Never empty for suppliers, potentially empty for portal administrators.																				
user.stateAsString	One of ACTIVE, REGISTERED, INVITED, NOT_INVITED, DEACTIVATED																				
user.isSupplierAdmin	True, if the user is supplier administrator.																				
user.isPortalUser	True, if the user is a portal administrator.																				

Json Attribute	Description	
	Variable	Description
	user.isSupplier	True, if the user is either a supplier user or a supplier administrator.
	organization.name	Name of the supplier organization.
	organization.id	Internal database id of the supplier organization
	organization.hpmSupplier Id	Supplier Id in PIM Core.
	organization.isActive	Active state of supplier organization.

10.5.2 User Permissions control available list of Actions

While you can contribute your own navigation bar actions, its also possible to control the available navigation bar action by adding or removing appropriate permissions for the available user roles.

For example if the INVITE_SUPPLIER permission is revoked, the appropriate action in the navigation bar will not be available anymore for the portal admin. To configure the list of available permissions for the PORTAL_ADMIN_ROLE or the SUPPLIER_ADMIN_ROLE, there are two configuration properties within your **<INSTALLATION ROOT>/configuration/configuration.properties** file.

```
permissions.portalAdmin=INVITE_SUPPLIER,VIEW_IMPORT_MANAGER,MANAGE_SUPPLIER_USER,MANAGE_SUPPLIER
permissions.supplierAdmin=START_DRY_RUN,MANAGE_SUPPLIER_USER
```

All available Permissions can be found in the following table:

Permission	Description
INVITE_SUPPLIER	Will control the navigation bar action "Invite suppliers" to start the supplier invitation workflow. (Default: only portal admins are able to invite suppliers)
VIEW_IMPORT_MANAGER	Will control the navigation bar action "Manage imports" to show the Import Overview. (Default: only portal admins are able to invite suppliers)
MANAGE_SUPPLIER_USER	Will control the "Create new User" Link within the Supplier Details, as well as all actions which will be corresponding to the action of creating new users like resending the invitation mail, activating or deactivating users. (Default: portal admins and supplier admins are able to create new users.)
MANAGE_SUPPLIER	Will control the actions to activate or deactivate an supplier organizations.
START_DRY_RUN	Will control the navigation bar action "Upload data to a catalog" which start give's the user the possibility to start a dry run. (Default: Only supplier admins are able to start dry run imports.)
MANAGE_BROKER_USER	Provides access to Broker user administration UI, e.g. for catalog assignment.

10.5.3 Supplier Portal as embedded application

There may be scenarios where Supplier Portal should be embedded into other web applications. This raises two requirements:

- it is desired to display only the main part of the application showing all the data and actions necessary to execute all tasks.
Navigation bars and header containing banners, login information etc. are not wanted.
- it should still be possible to navigate to the several places to do certain tasks, like going to the supplier list overview, invitation, upload data etc.
This should be possible by providing additional url parameters.

10.5.3.1 Display as embedded application

Hiding navigation bar and application header can simply be done by passing the url parameter *embedded* with the value *true* to the Supplier Portal url (*false* is default). F.e. <http://localhost:9090/hsx/html/Hsp.html?embedded=true>

10.5.3.2 Initialize with a specific locale

Users can switch their UI language on the login page. For initialization purposes it is also possible to use a URL parameter. This also works for application deep links (see next paragraph). Example:

http://localhost:9090/hsx/html/Hsp.html?locale=de_DE

The locale format is equivalent to the String representation of `java.util.Locale`.

10.5.3.3 Deep links to the application

Without the navigation bar it should still be possible to reach all areas of the application. Pages displayed in the main area are in the following termed as places.

To reach a certain place the identifier of the place is added and the end of the url beginning with a # symbol. Parameters for the place are followed behind a : symbol.

The syntax looks like this: <http://localhost:9090/hsx/html/Hsp.html#{placeIdentifier}:{parameters}> (see page 373). Places parameters follow the normal url parameter syntax: `parameterKey1=value1¶meterKey2=value2...`

Note that actions performed in the ui are always reflected in the url if it affects a certain place. F.e. opening the supplier details of a supplier in the suppliers list would change the url from <http://localhost:9090/hsx/html/Hsp.html#suppliers> to <http://localhost:9090/hsx/html/Hsp.html#supplier:supplierId={id}> (see page 373).

Below is a full list of places that are currently available in the application, also containing all parameters.

Navigable Site	Url	Parameters	Example
Dashboard, default place after login	#dashboard or nothing (because default)	<ul style="list-style-type: none"> feedId: internal id of feed, optional, only relevant after login (loads and displays only the specified feed) 	#dashboard:feedId=23
List of suppliers	#suppliers	-	#suppliers
Supplier details	#supplier	<ul style="list-style-type: none"> supplierId: internal id of supplier, mandatory 	#supplier:supplierId=4

Navigable Site	Url	Parameters	Example
List of mappings	#mappings	-	#mappings
Mapping details	#mapping	<ul style="list-style-type: none"> mappingId: internal id of mapping, mandatory 	#mapping:mappingId=5
Catalog details	#catalog	<ul style="list-style-type: none"> catalogId: internal portal id of catalog, mandatory 	#catalog:catalogId=46
User details	#user	<ul style="list-style-type: none"> userId: internal id of user (), mandatory 	#user:userId=8
Invite suppliers	#inviteSuppliers	-	#inviteSuppliers
Upload data (test run)	#upload	<ul style="list-style-type: none"> catalogId: internal portal id of catalog, mandatory 	#upload
Edit catalog	#editCatalog	<ul style="list-style-type: none"> catalogId: internal portal id of catalog, mandatory 	#editCatalog:catalogId=32
Manage imports	#imports	<ul style="list-style-type: none"> dataLoad filter, only for initial load 	#imports

10.5.4 Add custom pages to the support box



Available since 8.1.1.01

It is now possible to add up to 3 custom pages to the support box of the supplier portal.

In the directory `<SUPPLIER_PORTAL_INSTALLATION_DIR>/tomcat/webapps/hsx/html` there are 3 HTML-Pages named custom1.html to custom3.html which can be edited to the customer needs.

To add for example a link from the existing login page `login.html` to the new custom1.html page it is necessary to add the custom html to the following div:

login.html

```
[...]
<div id="hsx-page-navigation-all" class="hsx-page-navigation" style="display: none;">
  <span id="hsx-login-link"><a href="login.html" id="hsx-login-label">Log In</a>
| </span>
  <span id="hsx-registration-link"><a id="hsx-registration-label" href="Registr
ation.html">Register</a> | </span>
  <span id="hsx-terms-link"><a id="hsx-terms-label" href="terms.html">Terms
& Conditions</a> | </span>
  <span id="hsx-imprint-link"><a id="hsx-imprint-label" href="imprint.html">Leg
al disclaimer</a></span>
  <!-- new code -->
  <span id="hsx-custom1-link"><a id="hsx-custom1-label" href="custom1.html">Cus
tom Page Name</a></span>
</div>
[...]
```

This has to be done on every html page where the link to the custom page should be shown.

To make the link name language specific it is necessary to change the `SupportBoxMessages_<LOCALE>.properties` in every desired language located here `<SUPPLIER_PORTAL_INSTALLATION_DIR>\tomcat\webapps\hsx\WEB-INF\classes\com\heiler\hsp\ui\app\client`

As we have added the custom1.html we have to edit the property `hsx-custom1-label` to the desired name.

SupportBoxMessages_de_DE.properties

```
[...]
hsx-custom1-label=<Custom Name of the Page>
[...]
```

After editing the html and properties files a restart of the supplier portal server is required.

If the changes are not visible the customer will have to clear his browser cache and hard reload the site.

10.6 Manage and Customize Workflows

- [Introduction](#) (see page 384)
- [Customizing Workflows Definitions](#) (see page 384)
 - [Changing a workflow definition](#) (see page 385)
 - [Example - Proceed a running workflow \(receive task\)](#) (see page 385)

10.6.1 Introduction

PIM Supplier Portal uses the workflow engine Activiti internally. By default the following processes are modeled as workflows:

- Supplier Registration
- Supplier Invitation
- Data Upload Test Run
- Data Upload Import Run

From a customization point of view there are two aspects:

- Customize Workflow definition by changing the bpmn files
- Interact with the workflow engine from external systems by using the workflow REST API



Both options are available starting with PIM 7.0.4

10.6.2 Customizing Workflows Definitions

All workflows used in PIM Supplier Portal may be overwritten by customized workflows. The property **workflows.customizationFolderPath** defines the location of the folder where the customized workflows are contained. The default location is

<INSTALLATION_ROOT>\configuration\workflows

Currently the following workflows can be customized:


Workflow	Filename	Purpose (default behavior)
Registration	registration.activiti.bpmn20.xml	Self registration of a user to PIM Supplier Portal if no account is there yet.
Invitation	invitation.activiti.bpmn20.xml	Invitation by Portal Admin to join PIM Supplier Portal.
Upload	initialDataLoad.activiti.bpmn20.xml	Data upload by supplier user. An initial test run is done with this data. It is also later used for the actual import if the data has been approved by the portal admin.

Workflow	Filename	Purpose (default behavior)
Import	initialDataLoad.activiti.bpmn20.xml	Actual import of the data that has been uploaded in the upload workflow.

Note that the full file name is important, the file extension must always be **.bpmn20.xml**.

10.6.2.1 Changing a workflow definition

To change a workflow definition, simply open the bpmn20.xml file using your favorite XML editor.

 You might also want to use the Activiti Designer in Eclipse. However, this editor (last tested with Activiti 5.14) doesn't support all XML elements that are used in the Supplier Portal workflow definitions ("extensionElements" in particular). Hence the Activiti Designer should only be used for viewing but not for editing workflows.

Changing a workflow definition could incorporate things like adding/removing specific tasks or adding additional flows. To add custom logic to a workflow definition, integrate a new service task. Within the service task you can either use **activiti:class** to specify a class that is instantiated and called by the workflow engine. As an (preferred) alternative, use **activiti:delegateExpression** to reference a Spring bean that is called. The latter approach is especially preferred if other parts of the application are needed because Spring's dependency injection can be used to acquire the necessary references.

Interact with Workflow Engine

PIM Supplier Portal exposes the REST API of the internal workflow engine Activiti. This gives external systems a great flexibility to monitor and interact with running workflows or to manage workflow definitions. A typical use case is to proceed (ie. "signal") a paused workflow instance.


The full documentation can be found at <http://www.activiti.org/userguide/>

By default, the Activiti rest interface may be accessed in using the following url :

`http://localhost:9090/hsx/activiti-rest`

Please change host name, port and web app name to your local environment accordingly.

Authentication for all rest calls is done using http Basic Authentication. Clients are required to specify username and password of an existing PIM user.

 As username and password are not encrypted when using http Basic Authentication, the usage of SSL is recommended in production environments.

10.6.2.2 Example - Proceed a running workflow (receive task)

This example illustrates how a paused workflow can be continued ("signalled") using the Activiti REST API. This is done by using the call

POST /process-instance/{processInstanceId}/signal

as described in <http://www.activiti.org/userguide/#N160BB>.

The authenticated user is "portal/portal".

Request Header

```
POST /activiti-rest/process-instance/4e65e489-7c38-11e3-a7c0-6c88140eb6d0/signal
HTTP/1.1
Host: localhost:8888
Authorization: Basic cG9ydGFsOnBvcnRhbmA==
Cache-Control: no-cache
```

Request Body (JSON)

```
{
  "activityId": "approveInMDM",
  "supplierApprovalState": "reject",
  "supplierApprovalReason": "Missing bank account."
}
```



The parameter activityId is reserved in that context and must not be used for any other variables. The value is the name of the receive task that should be signaled. For example the activityId would be "approveInMDM" if the corresponding element in the activiti bpmn xml looks like this:

```
<receiveTask id="approveInMDM" name="Wait for approval in MDM" />
```

10.7 Supplier Portal REST Interface

10.7.1 General Remarks

Note

Please note that only few REST resources are available in the current version. The [RemoteAPI](#) (see [page 399](#)) can be used for session-based communication with the PIM Supplier Portal server and exposes most of the system's functionality.

The PIM Supplier Portal REST interface can be accessed using the url

/rest

When running the server locally, the full url looks like this with the default settings:

localhost:9090/hsx/rest/status

10.7.1.1 Authentication and Localization

Authentication is done use http Basic Authentication using username and password. As credentials are send plain text, using a secured connection over SSL is highly recommended for production usage.

Example using the credentials portal/portal:

```
GET /hsx/rest/status HTTP/1.1
Authorization: Basic cG9ydGFsOnBvcnRhbnA==
Accept-Language: en-US
```

With PIM8, a token based authentication is possible, too. A token can be generated over the [authToken interface \(POST\)](#) (see page 388). The returned token needs to be send base64 encoded. Example:

```
GET /hsx/rest/status HTTP/1.1
Authorization: Token
NzY4MzA2MEY4QUY3NDNFUFBDc0QUZDRTNFODg0MTEwMDBGQjM4NEU1RDA4QjY1MzA5NDZDMDBGQjAyMTg5Rj
jk4QkI5NUI0MEE2NkMyRDU2NzI1QjAxRjM1OTMwMkMxRDRBNzAwNUZDOERBNkU1NUM1MkFEOTE5NUY4NkFFNk
Y=
Accept-Language: en-US
```

In case of success http 200 is returned, in case of failure http 401.

Authentication can be done both for portal and supplier users. For successful authentication, the following conditions must be met:

- Credentials must be valid.
- Supplier user must exist in Supplier Portal. Supplier organization must be existing in PIM Application Server, too.
- Portal users must exist in PIM Application server and state active must be set.
- Supplier user state must be either active or registration_pending.
- The requested locale (Accept-Language) must be covered by the PIM license.

10.7.2 REST API Resources

10.7.2.1 Status & SSO

Resource	Description
GET /status (see page 392)	Returns system status. Can be used to test authentication, too.

Resource	Description
POST /authToken (see page 388)	Generates a SSO token that can be used for login.

10.7.2.2 Feeds

Feeds represent a chunk of messages with a feature message which follows multiple comments. With the REST API your able to send a single Feed with a feature message with or without an attachment.

Resource	Description
POST /feeds (see page 396)	Creates a new feed on the timeline.

10.7.2.3 Invitation

With the invitation related REST Interfaces it is possible to invite a suppliers which are not already active on the Heiler Supplier Exchange Portal.

Resource	Description
GET /invitable-suppliers (see page 390)	Returns a list of all supplier which are invitable.
GET /invitable-suppliers/{hpmSupplierId} (see page 391)	Get invitable supplier by its id.
PUT /invitable-suppliers/{hpmSupplierId}/users (see page 393)	Invite a (invitable) supplier to the PIM Supplier Portal.

10.7.3 Generate SSO Token



Since PIM 7.1.01

To improve integration of PIM Supplier Portal into other host applications, a token based mechanism for authentication can be used. From a user perspective, the goal is to log in only once (SSO).

This approach can be used if

- PIM Supplier Portal is authentication provider (holds identity information of users).
- PIM Supplier Portal is embedded into another application.

Authentication of both supplier users and portal users is supported.

The generated token can be used by attaching it to a link into the PIM Supplier Portal application. As of 7.1.01, the following link will authenticate the user with the given token:

`http://localhost:9090/hsx/html/Hsp.html?authToken=0x12345`

With the support of deep links into other parts of the application, the same approach can be used.

Starting with PIM8, the generated token can also be used for authentication of subsequent REST request. An example can be found on the page [Supplier Portal RESTInterface](#) (see page 386).

10.7.3.1 HTTP Method Type

POST

10.7.3.2 Resource URL

`/authToken/`

10.7.3.3 Response

Return Value	MediaType	Description
token object	JSON (Response Entity)	returns token information
Location-Header	http Header	URL that points to the token resource.

10.7.3.4 Example

POST `http://localhost:9090/hsx/rest/authToken`


```
{
  token: {
    "key": "0x12345",
    "creationDate": "2014-09-07T14:35:19Z",
    "user": "portal"
  }
}
```

10.7.4 Get a list of invitable suppliers

Get a list of invitable suppliers.

10.7.4.1 HTTP Method Type

GET

10.7.4.2 Resource URL

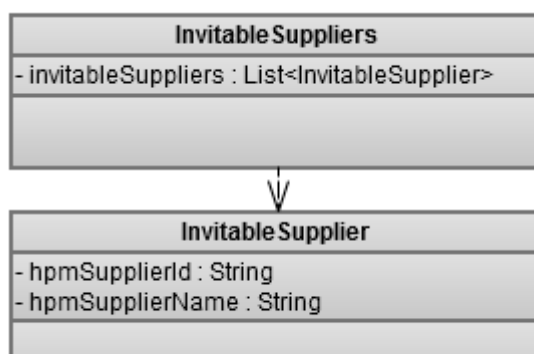
/invitable-suppliers/

10.7.4.3

Response

Return Value	MediaType	Description
invitable-suppliers object	JSON (Response Entity)	returns a invitable-suppliers object.

The following diagram describes the response object in UML:



Example Usage:

```
GET http://localhost:8080/hsx/rest/invitable-suppliers/
```

```
{
  "invitableSuppliers": [
    {
      "hpmSupplierId": "5196",
      "hpmSupplierName": "Gafisa S/A"
    },
    {
      "hpmSupplierId": "2405",
      "hpmSupplierName": "Schwarz-Gruppe"
    },
    {
      "hpmSupplierId": "2900",
      "hpmSupplierName": "Schrott Wenzel"
    }
  ]
}
```

10.7.5 Get invitable supplier by HpmSupplierId

Get a single invitable supplier by its id.

10.7.5.1 HTTP Method Type

```
GET
```

10.7.5.2 Resource URL

```
/invitable-suppliers/{hpmSupplierId}
```

10.7.5.3 Parameters

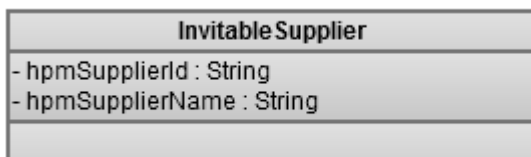
Parameter	Type	Required	Description
{hpmSupplierId}	URL Path Parameter	yes	specifies the HPM supplier id.

10.7.5.4

Response

Return Codes		Description
HTTP 404 Status Code		No invited supplier with id=xxx found.
Return Value	MediaType	Description
invitable-suppliers object	JSON (Response Entity)	returns a invitable-supplier object.

The following diagram describes the response object in UML:



Example Usage:

```
GET http://localhost:8080/hsx/rest/invitable-suppliers/2405
```

```
{
  "hpmSupplierId": "2405",
  "hpmSupplierName": "Schwarz-Gruppe"
}
```

10.7.6 Get status

Returns the system status. Is a very inexpensive operation and can be used to test authentication credentials, too.

 Since PIM 7.1.01

10.7.6.1 HTTP Method Type

GET

10.7.6.2 Resource URL

/status/

10.7.6.3 Response

Return Value	MediaType	Description
status object	JSON (Response Entity)	returns static system status information, like application version.

10.7.6.4 Example Usage:

GET http://localhost:8080/hsx/status/

```
{ (see page 392)
  "applicationVersion":[ (see page 392)
    "8.0.0.20140912-1405"
  ],
  "applicationServer":[ (see page 392)
    "localhost"
  ]
}
```

10.7.7 Invite a supplier user

Invite an invitable supplier given by the PIM Product 360 supplier id to the PIM Supplier Portal together with supplier user data. The given user data is used to create the user as supplier administrator in case the supplier is in an invitable state (invitation not initiated yet for that supplier).

Please have a look at the diagram below to understand the response behavior for different scenarios.

You may want to execute a Rest api call [Get a list of invitable suppliers \(see page 390\)](#) in advance to determine whether it makes sense to execute the call with a specific supplier.

10.7.7.1 HTTP Method Type

PUT

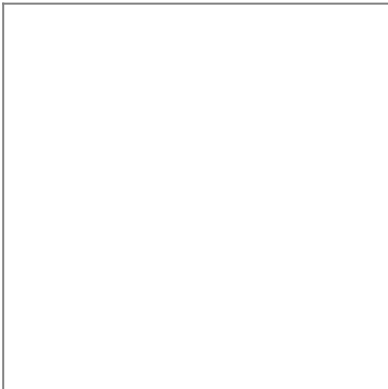
10.7.7.2 Resource URL

/invitable-suppliers/{hpmSupplierId}/users/

10.7.7.3 Parameters

Parameter	Type	Required	Description
{hpmSupplierId}	URL Path Parameter	yes	specifies the HPM supplier id.
user object	JSON (Request Entity)	yes	User object which sets the attributes of the user to invite.

The following diagram describes the request object in UML:



The following data describes the request object in JSON format

```
{
  "email": "happy@hsx.com",
  "firstName": "Happy",
  "lastName": "Gilmore",
  "language": "en_US"
}
```

Example Usage:

```
PUT http://localhost:9090/hsx/rest/invitable-suppliers/1768/users HTTP/1.1
{
  "email":"happy@hsx.com",
  "firstName":"Happy",
  "lastName":"Gilmore",
  "language":"en_US"
}
```

10.7.7.4 Response

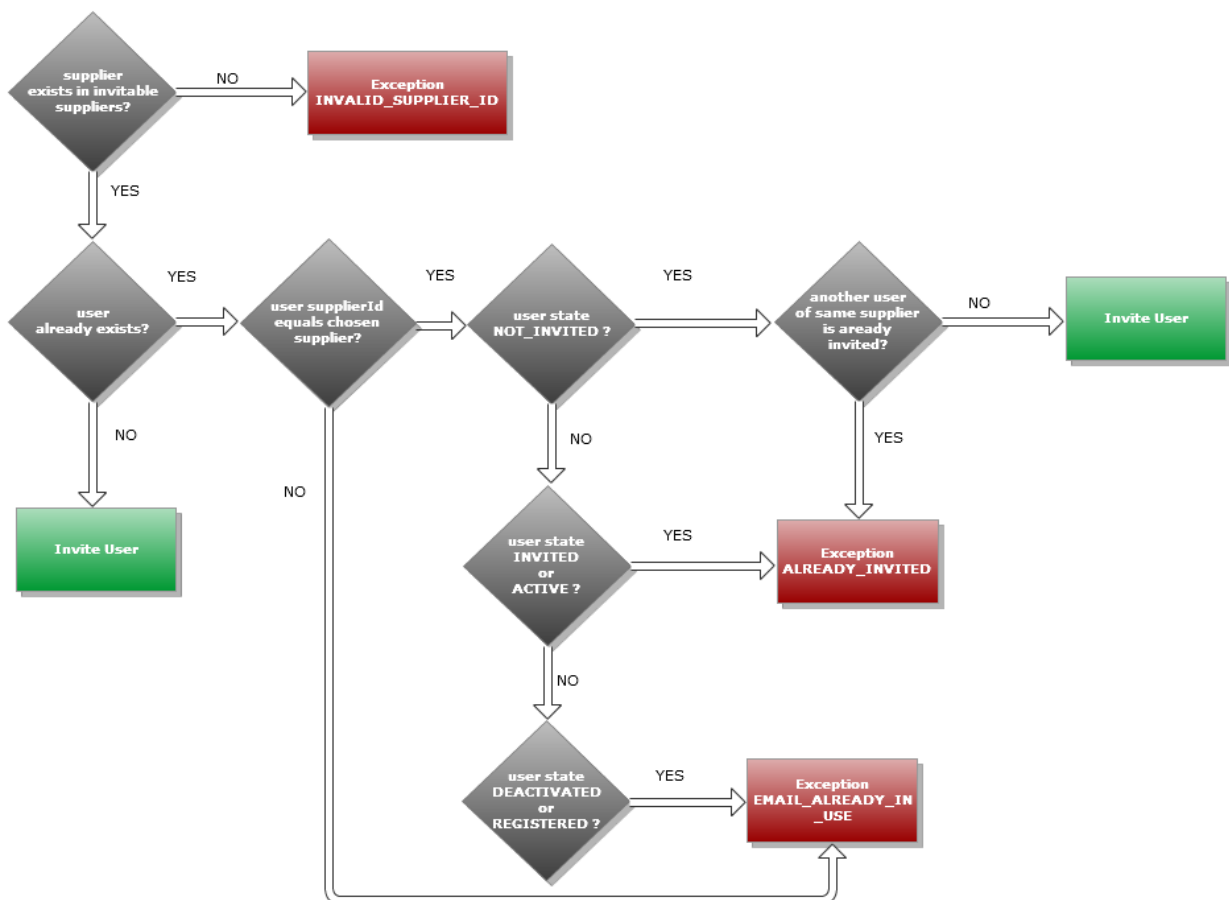
Example Response Header

```
HTTP/1.1 202 Accepted
Server: Apache-Coyote/1.1
Content-Length: 0
Date: Wed, 20 Jun 2012 08:53:34 GMT
```

Return Codes	Description
HTTP 202 Status Code	The invitation of the user was successful.
HTTP 400 Status Code INVALID_LANGUAGE_CODE	Invalid locale format: xxx.
HTTP 400 Status Code EMAIL_ALREADY_IN_USE	Email already in use.
HTTP 400 Status Code ALREADY_INVITED	The supplier xxx already has been invited.
HTTP 400 Status Code	The supplier xxx cannot be invited because no catalog has been assigned.

Return Codes	Description
HTTP 404 Status Code INVALID_SUPPLIER_ID	No invitable supplier with id=xxx found.

10.7.7.5 Decision diagram: Is user invitable?



10.7.8 Post a timeline message

Posting a timeline message to suppliers, with or without attachments.

10.7.8.1 HTTP method Type

POST

10.7.8.2 Resource URL

/feeds

10.7.8.3 Parameters

Parameter	Type	Required	Description		
multipart	JSON (multipart/mixed) (Request entity)	yes	The multipart is transferred in the request body. To send only a message without attachment, don't set the attachment and attachmentOriginalFileName body parts in the multipart object. To send a message with attachment you have to set all of the following body part parameters in the multipart object. An HTTP multipart consists of the following bodypart order:		
			BodyPart	MIME Type	Description
			1. feed object	application/json	includes the message of the feed and a hpmSupplierId to which the Feed will be posted.
			2. attachment	application/octet-stream	the attachment binary data.
			3. attachmentOriginalFileName	text/plain	the fileName plus extension of the attachment (e.g avatar.png)

the following diagram describes the multipart feed object object in UML:

Feed
- message : String
- hpmSupplierId : String

10.7.8.4

Response

Return Value	MediaType	Description
Only HTTP Header	-	Returns the URL to the created Feed. The URL is located in the HTML location header attribute.

Example Response Header

```
HTTP/1.1 201 Created
Server:Apache-Coyote/1.1
Set-Cookie:JSESSIONID=3421DFFD6EA3F0ED7EE154124B4FF51F; Path=/hsp/; HttpOnly
Location: http://hsis961:9090/hsp/rest/feeds/2577
Content-Length: 0
Date: Thu, 29 Mar 2012 11:35:50 GMT
```

Return Codes	Description
HTTP 201 Status Code	if creation of the feed was successful.
HTTP 400 Status Code	if multipart body part size not of the correct size.
	if multipart is invalid (wrong bodyPart sequence or insufficient/invalid bodyParts).
	if supplier try to send feed to another supplier.

10.7.8.5 Example Usage:

```

POST http://hsis961:9090/hsp/rest/feeds HTTP/1.1
Content-Type: multipart/mixed; boundary=Boundary_1_1423506714_1333020950758
Authorization: Basic cG9ydGFsOnBvcnRhbnA==
MIME-Version: 1.0
User-Agent: Java/1.6.0_21
Host: hsis961:9090
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 12495

--Boundary_1_1423506714_1333020950758
Content-Type: application/json

{"hpmSupplierId":"1323291296","message":"look what I got."}
--Boundary_1_1423506714_1333020950758
Content-Type: application/octet-stream
...
BINARY DATA
...
--Boundary_1_1423506714_1333020950758
Content-Type: text/plain

avatar.png
--Boundary_1_1423506714_1333020950758--

```

10.8 Supplier Portal Remote API

- [Overview](#) (see page 399)
 - [Authentication & Security](#) (see page 400)
- [Getting Started](#) (see page 400)
 - [Sample Code - Create new supplier](#) (see page 401)
 - [Sample Code - Get information about logged in user](#) (see page 402)
 - [List of Available Services](#) (see page 402)
- [Further Reading](#) (see page 404)

10.8.1 Overview



Since PIM 7.1.01

PIM Supplier Portal comes with a http-based remote API. This API can be used to remotely trigger functionality or to retrieve supplier related data. This API is complete in the sense of that every feature that is available in the Browser UI can also be accessed over the Remote API.

In contrast to the [Supplier Portal REST API](#) (see page 386) the application protocol of the API is proprietary (defined by the GWT Request Factory). Hence, clients should use a wrapper library that takes care of assembling http requests and parsing the http responses.

10.8.1.1 Authentication & Security

All communication is http based. User credentials are transferred within http post requests. Hence, all communication should be encrypted using SSL over https in production scenarios.

Authentication is done the same way as for the Browser-based UI. That means, that after successful authentication of valid user credentials a sessionId is generated and can be used for all subsequent requests. Because of that, explicit logout is required by clients. Also the configured http session time-out needs to be considered after a longer time of inactivity.

Both user types PIM portal administrators as well as supplier user can be used for authentication.

For localization the Accept-Language http header is used to request information using a specific language. If omitted, the server default locale will be used. This affects some localized data, e.g. when fetching the timeline for a user.

10.8.2 Getting Started

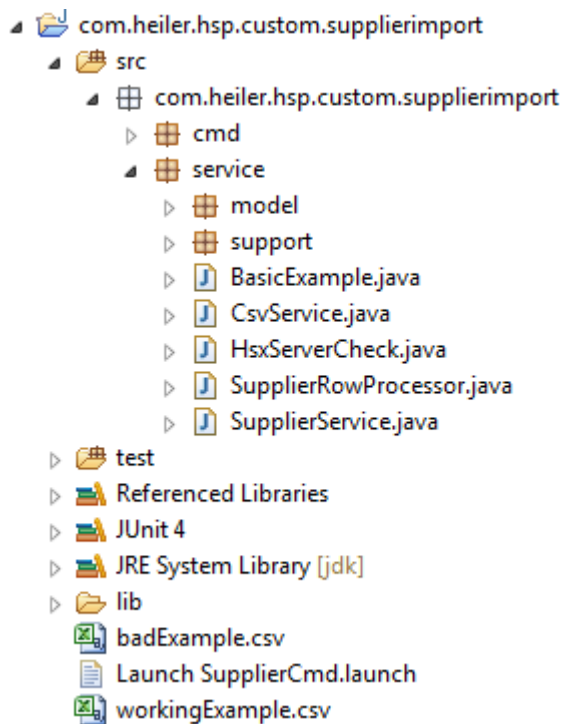


The Java Client for the Remote API and a sample project are available on request from Product Management.

The sample application shows how to implement a batch import for an existing supplier user database from an CSV file.

The sample project "com.heiler.hsp.custom.supplierimport" is available in the SDK package.

The source code itself is rather self explanatory, most communication with the PIM Supplier Portal happens in the class SupplierService.



Note that the Remote API alone is not enough to get the sample running, since the Remote API is built on top of the GWT Request Factory. Thus, additional dependent libraries are needed. These reside in the lib folder of the sample project. Although there are already Remote Api jars api-<version>-SNAPSHOT.jar inside this folder, please always use the Remote Api jars provided to you by the Product Management that fit to your Supplier Portal version. Keep all this in mind when setting up own projects using the Remote Api.

10.8.2.1 Sample Code - Create new supplier

This example shows how to create a new supplier organization with a supplier administrator user:

```
// Login with a PIM portal administrator user
FactoryHelper helper = new FactoryHelper( "http://localhost:9090/hsx" );
helper.login( "portal", "portal" );

// Create a factory and context to assemble a http request
UserRequestFactory factory = helper.create( UserRequestFactory.class );
UserRequestContext context = factory.context();

// Create a supplier entity and set attributes
OrganizationProxy supplier = context.create( OrganizationProxy.class );
supplier.setName( "New Supplier" );
```

```

supplier.setActive( true );
supplier.setAutoImportConfiguration( AutoImportConfig.AUTO_IMPORT_NO_WARNINGS );

// Create a supplier user, link with supplier and set attributes
UserProxy user = context.create( UserProxy.class );
supplier.setUsers( Arrays.asList( user ) );
user.setEmail( "admin@supliertest.com" );
user.setFirstName( "Mister" );
user.setLastName( "Administrator" );
user.setStateAsString( UserProxy.STATE.ACTIVE );

// Send the requests synchronously
UserProxy savedUser = FactoryHelper.fire( context.saveWithPassword( user,
"secret123" ) );

// Handle the result. Supplier user and supplier are persisted now.
System.out.println(savedUser);

// Logout PIM portal administrator user
helper.logout();

```

10.8.2.2 Sample Code - Get information about logged in user

This example shows how to retrieve information about the logged in supplier user.

```

// Login as supplier user at default url http://localhost:9090/hsx
FactoryHelper helper = new FactoryHelper().login( "admin@supliertest.com",
"secret123" );

// Create a factory and context to assemble a http request
SecurityRequestFactory factory = helper.create( SecurityRequestFactory.class );
SecurityRequestContext context = factory.securityContext();

// Load data about logged in user with supplier organization details.
UserProxy loggedInUser = FactoryHelper.fire( context.getLoggedInUser().with(
"organization" ) );

System.out.println( loggedInUser );

// Logout supplier user
helper.logout();

```

10.8.2.3 List of Available Services

The following tables gives an overview about the most important services that can be access remotely.

As a general note, all factories with the suffix **NoAuth** are accessible without being logged in as a user.

Service Factory Class Name	Purpose
UserRequestFactory	Manage supplier users, load list of users in batch.
OrganizationRequestFactory	Manage supplier organizations.
FeedRequestFactory	Handles timeline messages and feeds (a feed contains one or multiple messages). Provides methods to load a timeline, to create new feeds or to respond to existing messages.
InitialDataLoadRequestFactory	Allows to trigger the test run workflow for supplier users.
ImportRequestFactory	Allows to trigger the import run workflow for supplier users.
DataLoadRequestFactory	Provides access to the results of a test/import run. The DataLoad entity references also the destination catalog and supplier as well as the user that actually triggered the job.
MappingRequestFactory	Provides access and management of import mappings that are necessary to run a import job.
RegistrationRequestFactory RegistrationRequestFactoryNoAuth	Manages the supplier registration workflow. The *NoAuth interface is accessible without authentication.
InvitationRequestFactory	Manages the supplier invitation workflow.
SecurityRequestFactory	Provides information about the authenticated user and its permissions.
SystemRequestFactory	Provides information about the system configuration, e.g. available languages and application version.
LostPasswordRequestFactoryNoAuth	Generates a email for lost password mechanism.

10.8.3 Further Reading

Details about the semantics of `RequestFactory#create()` the `#with()` syntax can be found in the official guide <http://www.gwtproject.org/doc/latest/DevGuideRequestFactory.html#using>.

10.9 Supplier Portal I18n

10.9.1 Overview



Since PIM 7.1

PIM Supplier Portal is able to handle users with different UI languages. Users can change their language on the login page.

The available list of languages is restricted by two factors:

- Language packs being installed
- PIM license stored in PIM Server

The last recently language is stored in a Browser cookie. Also, a locale can be specified using a query parameter, e.g. http://localhost:9090/hsx/html/login.html?locale=de_DE.

10.9.2 Install Additional Language Pack



This functionality is available as of PIM 7.1.

Administrators can configure where language packs are loaded from. This is done with the following two parameters in configuration.properties file:

```

# Spring resource path pointing the message property files for the UI
# This is a SPRING resource path, more information: http://static.springsource.org/
spring/docs/current/spring-framework-reference/html/resources.html
# Use this pattern to pick up files from configuration area:
# file:${hsx.configurationArea}/i18n/ui/**/*Messages.properties
i18n.uiResourcesPath=classpath*:com/heiler/hsp/**/ui/**/*Messages.properties

# Spring resource path pointing the message property files for the Server/Backend
# This is a SPRING resource path, more information: http://static.springsource.org/
spring/docs/current/spring-framework-reference/html/resources.html
# Use this pattern to pick up files from configuration area:
# file:${hsx.configurationArea}/i18n/server/**/*Messages.properties
i18n.serverResourcesPath=classpath*:com/heiler/hsp/**/i18n/*Messages.properties

```

Per default, all property files are loaded from within the WAR file. By using the option beginning with **file:*** any location on the local disc can be specified. For convenience, the Supplier Portal binary already contains a full set of all available languages in the directory **<Installation_Root>/configuration/i18n**.

To add another language, make sure to provide translated property files next to the existing files within the configured directory structure.

As all property files are stored in the given directory structure, changing the existing texts for the standard language is possible, too. Please make sure to keep track of all changes to smoothen migration when applying a new hotfix, e.g. in SVN.

10.10 Customized Data Model for Supplier Data

- [Introduction](#) (see page 405)
- [Adding User and Supplier fields](#) (see page 406)
- [Field Permissions](#) (see page 416)
 - [User Fields](#) (see page 416)
 - [Supplier Fields](#) (see page 416)

10.10.1 Introduction

Supplier Portal already comes with a fix set of attributes for suppliers and users like supplier name or a user's first and last name. In order to be able to improve supplier communication and manage data at a central place, Supplier Portal offers the possibility since PIM 7.1.01 to add and manage additional data fields. This could be user data like:

- Contact information, e.g. phone, messenger
- Address data
- General remarks, e.g. reason for deactivation

User data resides within the Supplier Portal. Data of a supplier is directly linked to repository fields in PIM Core, which also means that metadata like field types are retrieved from there. Supplier fields are for example:

- Duns
- Iln

- or any arbitrary customized reserved field

Persistence of supplier data is managed within PIM Core.

Custom field data are available at multiple places in the UI, e.g. in registration forms or in details views.

10.10.2 Adding User and Supplier fields

Additional fields are managed by configuring the file <INSTALLATION_ROOT>/configuration/dataModelCustomization.json. This is the default path of the customization file. It may be changed via the configuration property *dataModelCustomization.file*.

Here is an example of a customized data model:

```

1  {
2      "customUserFields": [
3          {
4              "_comment": "Required unique identifier. Has no particular
5              pattern, but we use the same identifier pattern as PIM Service API
6              fieldIdentifiers as a convention.",
7              "identifier": "User.Phone",
8              "_comment": "for custom supplier user fields you need to add
9              the labels within this configuration file. Add a locale/value tuple for
10             each language you want to support.",
11             "labels": [
12                 {
13                     "locale": "de_DE",
14                     "value": "Telefon"
15                 },
16                 {
17                     "locale": "en_US",
18                     "value": "Phone"
19                 }
20             ],
21             "_comment": "4 different custom field types are supported at
22             the moment: java.lang.String, java.lang.Boolean, java.sql.Date,
23             java.lang.Integer",
24             "type": "java.lang.String",
25             "_comment": "custom supplier user fields can have a maxLength
26             definition from 1 to 255. If none is defined the default value will be
27             255. A maximum length restriction is only effective in the ui for
28             java.lang.String types. In such cases entering more characters than the
29             defined maximum will be prevented by ignoring the key press.",
30             "maxLength": "255"
31         },
32         {
33             "identifier": "User.DateOfBirth",
34             "labels": [
35                 {
36                     "locale": "de_DE",
37                     "value": "Geburtsdatum"
38                 }
39             ]
40         }
41     ]
42 }
```

```

28         },
29         {
30             "locale": "en_US",
31             "value": "Date of Birth"
32         }
33     ],
34     "type": "java.sql.Date",
35 }
36 ],
37 "_comment": "Supplier data is located in PIM server and need to be
contributed there. Don't forget to make the custom field available over
the Service API, cause the supplier portal is fetching all the custom
supplier fields over the Service API. Only the field identifiers need to
be specified, all other information are taken from the PIM Repository.",
38 "customSupplierFields": [
39     {
40         "_comment": "to add an custom supplier field to the supplier
portal, just add it's Service API compatible field identifier within this
configuration section.",
41         "fieldIdentifier": "Party.Duns"
42     },
43     {
44         "fieldIdentifier": "Party.Iln"
45     }
46 ]
47 }
48

```

The JSON file consists of two sections, one for user data ("customUserFields") and one for supplier data ("customSupplierFields"). Each section contains a list of field definitions.

In general all such custom fields have the same attributes, but the value of some of them cannot be specified via the json file but is determined dynamically. For example whether a field is editable or not is always determined dynamically.

Below is a definition of a well-formed json file for customized fields:

Json Attribute	Mandatory	Definition								
customUserFields	No	<div>Contains a json array whereas each element describes a custom user field element. Such a field element contains following attributes</div> <table><tr><th>Json Attribute</th><th>Mandatory</th><th>Example</th><th>Definition</th></tr><tr><td>identifier</td><td>Yes</td><td>User.Phone</td><td>Required unique identifier. Has no particular pattern, but we use the same identifier pattern as PIM Service API fieldIdentifiers as a convention.</td></tr></table>	Json Attribute	Mandatory	Example	Definition	identifier	Yes	User.Phone	Required unique identifier. Has no particular pattern, but we use the same identifier pattern as PIM Service API fieldIdentifiers as a convention.
Json Attribute	Mandatory	Example	Definition							
identifier	Yes	User.Phone	Required unique identifier. Has no particular pattern, but we use the same identifier pattern as PIM Service API fieldIdentifiers as a convention.							

Json Attribute	Mandatory	Definition								
		Json Attribute	Mandatory	Example	Definition					
		labels	No, but should contain entries for each desired localized language	[{ "locale": "de_DE", "value": "Telefon" }, { "locale": "en_US", "value": "Phone" }]	Contains a json array whereas each element describes a locale key / value pair for localization of field labels. Such a pair contains following attributes					
						J s o n A t t r i b u t e	E x a m p l e	M a n d a t o r y	D e f i n i t i o n	
						l o c a l e	d e - D E	Y e s	L o c a l e o f t h e l o c a	

Json Attribute	Mandatory	Definition						
		Json Attribute	Mandatory	Example	Definition			
					Json Attribute	Example	Mandatory	Definition
								lized field label in the conventional

Json Attribute	Mandatory	Definition			
		Json Attribute	Mandatory	Example	Definition
					Json Attribute Example Mandatory Definition
					Java pattern - Language initials ISO 639 + " - " + country

Json Attribute	Mandatory	Definition			
		Json Attribute	Mandatory	Example	Definition
					Json Attribute Example Mandatory Definition
					value Telephone No, but then will contain only file identifier as

Json Attribute	Mandatory	Definition			
		Json Attribute	Mandatory	Example	Definition
					Json Attribute Example Mandatory Definition
					is visible at ULI in case user is logged in with the

Json Attribute	Mandatory	Definition			
		Json Attribute	Mandatory	Example	Definition
		type	Yes	java.sql.Date	4 different custom field types are supported at the moment: java.lang.String, java.lang.Boolean, java.sql.Date, java.lang.Integer. The types are displayed in the UI with corresponding widgets, f.e. String as text input field or Boolean as checkboxes. Fields with unsupported types are ignored.

Json Attribute	Mandatory	Definition											
		<table><tr><th>Json Attribute</th><th>Mandatory</th><th>Example</th><th>Definition</th></tr><tr><td>maxLength</td><td>No</td><td>25</td><td>User fields can have a maxLength definition from 1 to 255. If none is defined the default value will be 255. A maximum length restriction is only effective in the ui for java.lang.String types. In such cases entering more characters than the defined maximum will be prevented by ignoring the key press.</td></tr></table>				Json Attribute	Mandatory	Example	Definition	maxLength	No	25	User fields can have a maxLength definition from 1 to 255. If none is defined the default value will be 255. A maximum length restriction is only effective in the ui for java.lang.String types. In such cases entering more characters than the defined maximum will be prevented by ignoring the key press.
Json Attribute	Mandatory	Example	Definition										
maxLength	No	25	User fields can have a maxLength definition from 1 to 255. If none is defined the default value will be 255. A maximum length restriction is only effective in the ui for java.lang.String types. In such cases entering more characters than the defined maximum will be prevented by ignoring the key press.										
customSupplierFields	No	<p>Contains a json array whereas each element describes a custom supplier field element. Such a field element contains following attributes</p> <table><tr><th>Json Attribute</th><th>Mandatory</th><th>Example</th><th>Definition</th></tr><tr><td>identifier</td><td>Yes</td><td>Party.Duns</td><td>Service API compatible field identifier.</td></tr></table>				Json Attribute	Mandatory	Example	Definition	identifier	Yes	Party.Duns	Service API compatible field identifier.
Json Attribute	Mandatory	Example	Definition										
identifier	Yes	Party.Duns	Service API compatible field identifier.										

10.10.3 Field Permissions

Field permissions have a fix logic for custom user fields and for supplier fields permissions are managed and retrieved within PIM Core. For portal administrator users, the corresponding field permission for the logged in users are respected. For supplier users, the permissions for the configured REST user are used. That means, that all supplier users will share the same field permissions.

10.10.3.1 User Fields

- Visible: Always visible
- Editable: Portal admins can always edit such fields, supplier admins can edit data of users of the same organization and other users may only edit their own data.

10.10.3.2 Supplier Fields

- Visible and Editable: A supplier field is only visible or editable when the corresponding user with which the Service API call is made has the appropriate permission (managed in the field rights section in PIM Core).

11 Tooling

- [Run-time Tools](#) (see page 416)
 - [Command Inspector](#) (see page 416)
 - [Eclipse Plug-In Spy](#) (see page 417)
- [Development Tools](#) (see page 421)

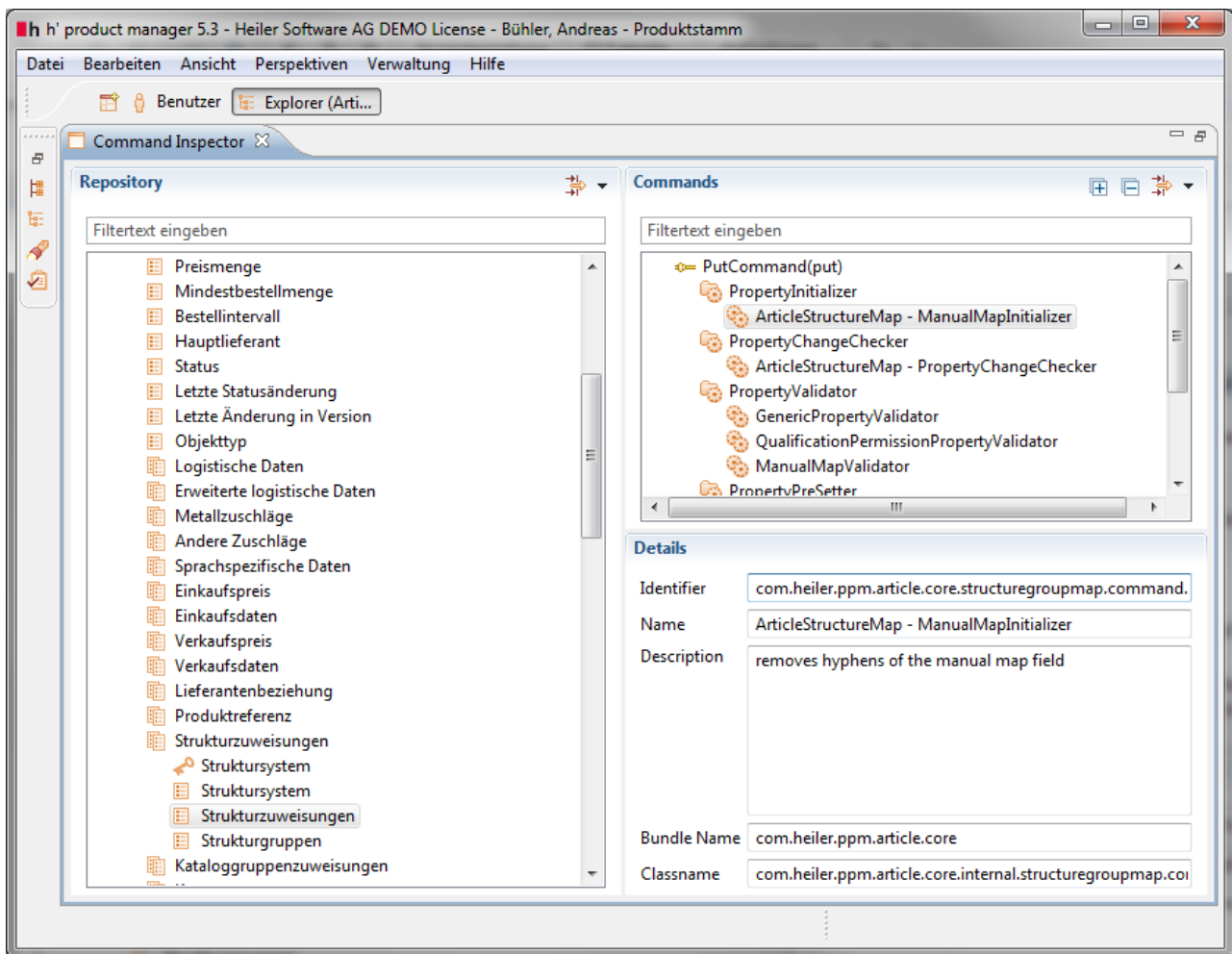
The following tools have been created to help you develop with and for the Heiler Product Manager SDK. They are included in the SDK distribution beginning with version 6.0.

In general we have two kinds of tools. Runtime tools and development tools. The run time tooling is being executed either on the Product Manager client or server, the development tools are executed only from within your development environment.

11.1 Run-time Tools

11.1.1 Command Inspector

The command inspector gives you the ability to check what command operators are contributed for a specific field or entity and in which order they are executed. Additional to that you will get knowledge which class implements the logic and (if given in the command contribution) the description of the operator implementation. This way you can easily see what kind of business logic is being executed when you modify a field or entity.



5 image2012-3-15 11:8:42.png

11.1.1.1 Installation and Execution

The Command Inspector must be executed with the Product Manager Client. It's contained in the SDK tooling feature which can be launched together with the client from your development environment.

Starting by adding new view "Command Inspector" to you client.

11.1.2 Eclipse Plug-In Spy

In the default package of Eclipse for RCP and RAP Developers you can find a plugin called "Plugin-in Spy" which provides two very useful functions.

- Plugin-in Menu Spy (call with the shortcut ALT+RSHIFT + F2)

Plugin-in Selection Spy

Provides information about the active view and the active selection. Call with the shortcut ALT + R-SHIFT + F1.

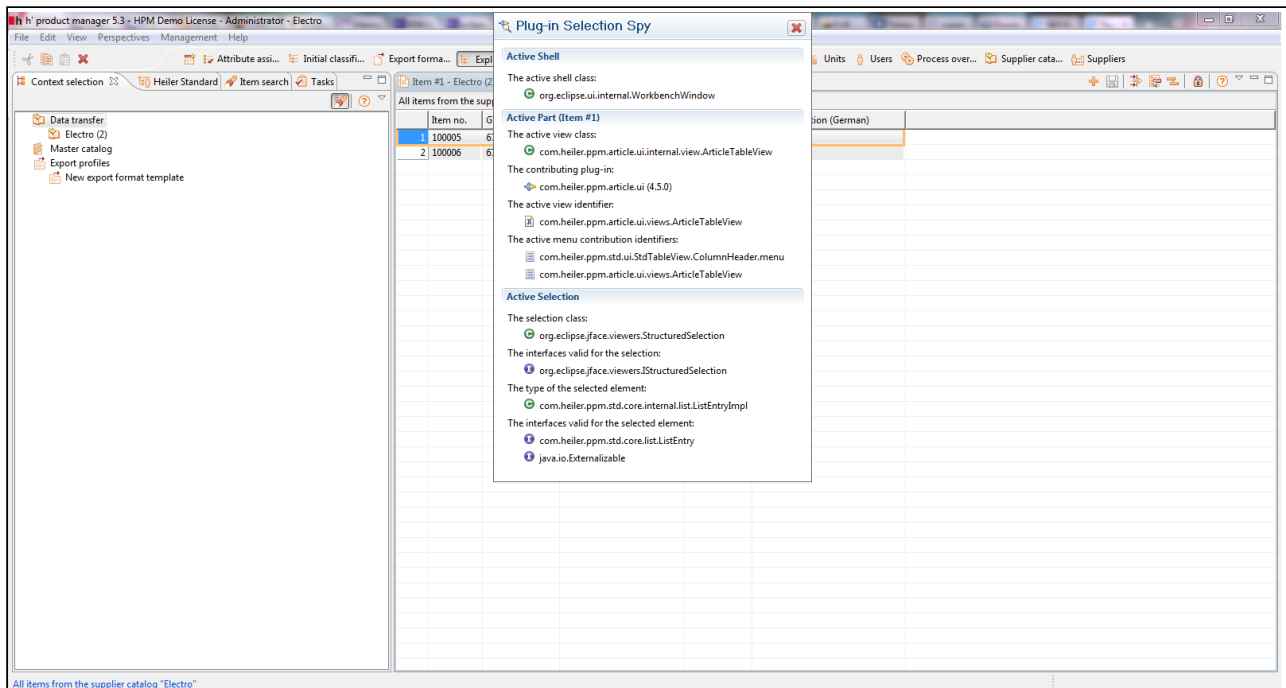
Plugin-in Menu Spy

Provides information about a button or a menu entry (only active). Call with the shortcut ALT + R-SHIFT + F1.

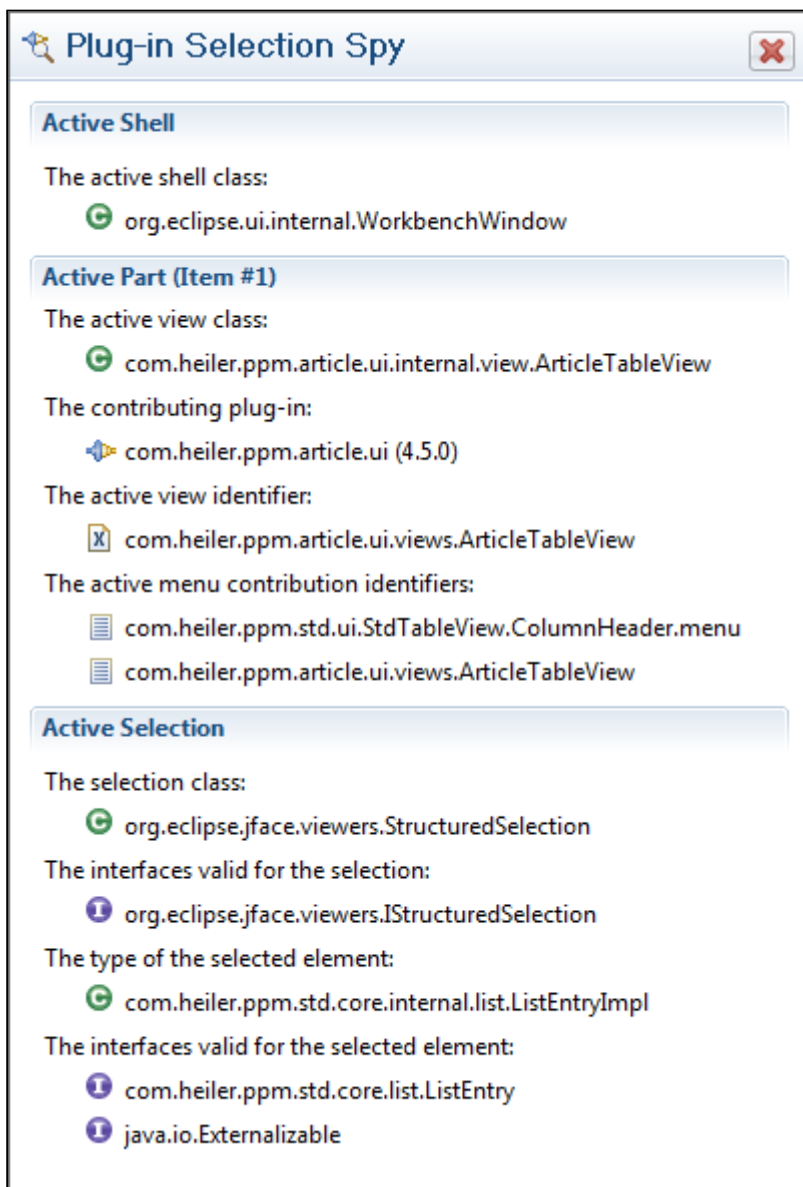
11.1.2.1 Installation and Execution

For your convenience we included the necessary bundles in the client runtime of the SDK since HPM 6.0 (it's not available in the stand alone client!).

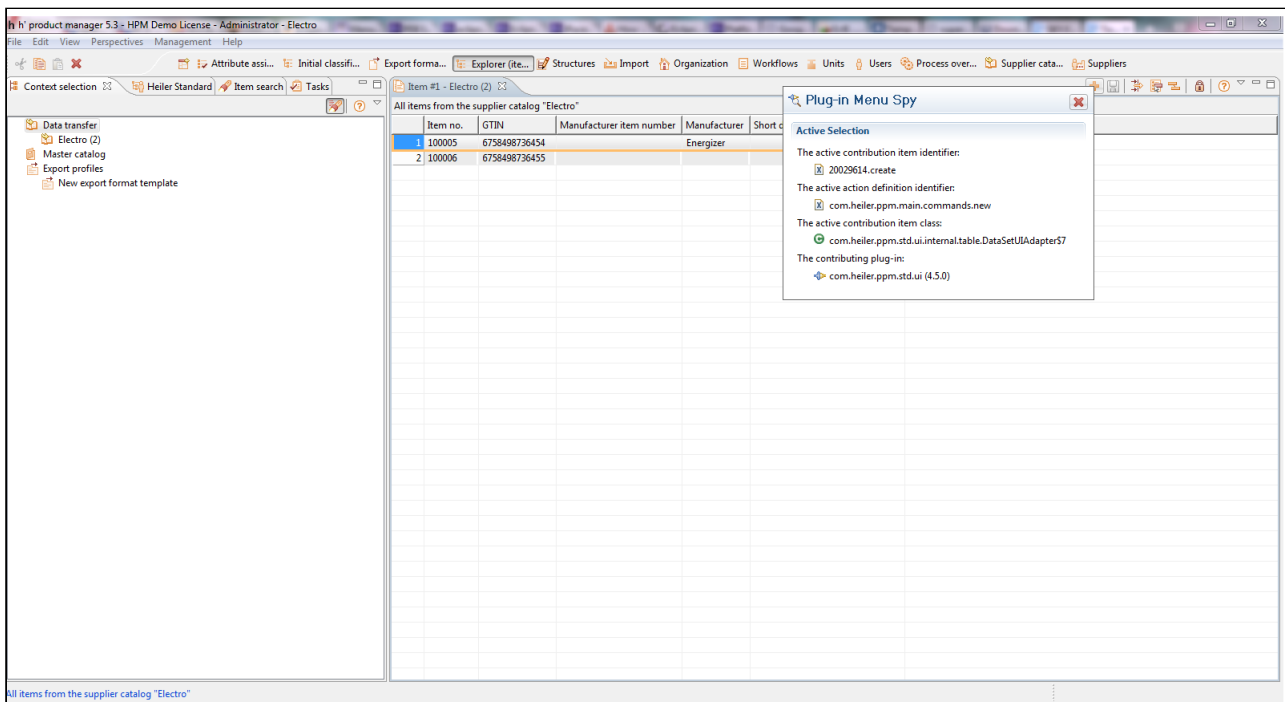
11.1.2.2 Screenshots



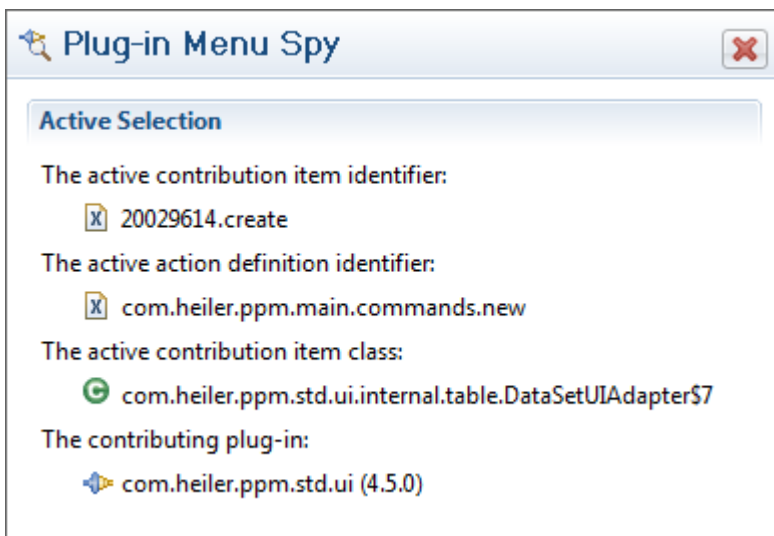
6 Selection Spy Workbench.png



7 Selection Spy Popup.png



8 Menu Spy Workbench.png



9 Menu Spy Popup.png

11.2 Development Tools

12 Database

12.1 Contribute custom database scripts

With the new Database Setup for v8.0 we added a new extension point

(`com.heiler.ppm.dbsetup.core.sqlScript`) where it is possible to contribute customized update scripts. With a flexible declaration it is possible to define the order of these scripts so they can run before or after a specific standard script. Furthermore it is possible to create whole new schemas by defining them in the extension point.

12.2 Custom Update Scripts

To add a new custom update script create a new Plugin and add a dependency to the plugin

`com.heiler.ppm.dbsetup.core` and add the extension point

`com.heiler.ppm.dbsetup.core.sqlScript`.

The extension point is constructed as follow: DB type -> schema -> script type -> script

In the screen shot below you can see how the structure looks like. You will only have to declare the schemas and types you want to contribute to. So if you want to add an update script for the main schema, just add MSSQL -> main -> update -> your script.

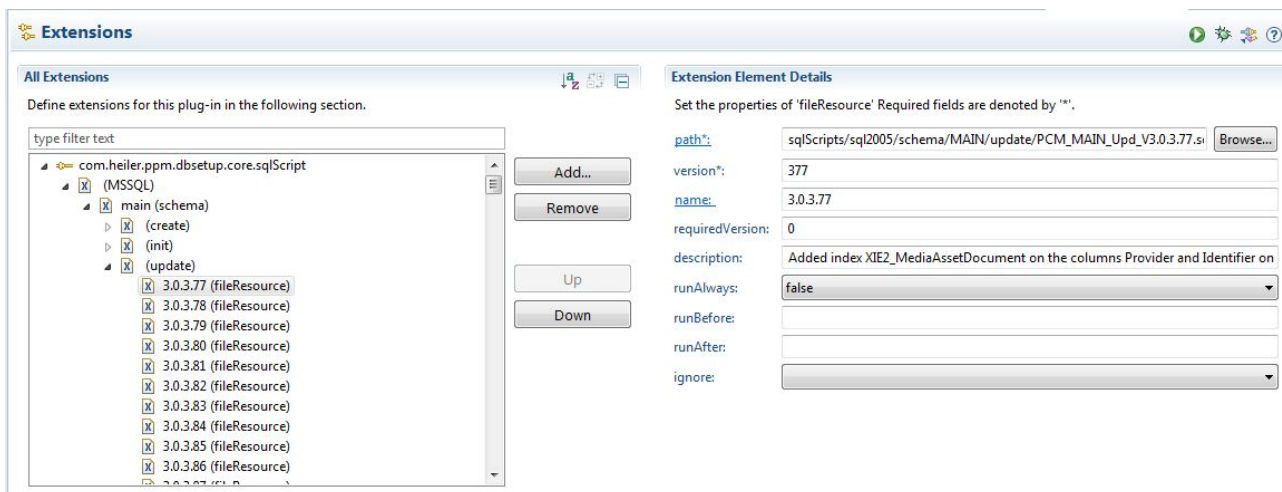
To select your update script it has to be located inside of your plugin.



Your script file has to be encoded with UTF-8. If not, you may get an encoding exception during execution.

Since `version` has to be unique and is used as the order of execution, you will have to define a version that is not taken by our standard scripts. As a convention use only versions above 1000. To execute a script before or after a script from the standard, for example to drop an index, you can define the corresponding script in the `runBefore` or `runAfter` field. Just enter the version of that script. The version consists of the last 3 digits of the update script name (without the dot). If left empty, the defined version of that script will be used as the order of execution.

If `runAlways` is set to false, which is the default, then the version is written into the database and it won't be executed on the next run. If set to true you have to make sure this script can be executed multiple times without any dependencies to the data structure.



13 Integration

The integration manual provides an overview on the different possibilities to integration Product 360 with 3rd party applications. The same technologies are used to integrate with the Informatica BPM system.

13.1 Database Access

Direct access of 3rd party applications to the Informatica MDM Product 360 database schemas is **not** supported. This is not only the case for write access but also for read access. Third party applications reading and/or writing to the tables might interfere with the functionality and performance of the application.

For specific 3rd party applications, like the print integration with Werk II, a read only access to the database has been allowed in the past, knowing that the print generation might reduce the overall performance of the system. However, we strongly recommend to upgrade to the latest Werk II version which provides a Service API integration with Informatica MDM Product 360 in which no direct database access is needed any more.

13.2 File Based Integration

13.2.1 Import

The hotfolder of the import provides an easy way to automatically import files. A detailed documentation about the hot folder can be found here. Besides the hotfolder, an import can also be started by the rest based Service API's [Import API](#) (see page 741).

13.2.2 Export

Available post export steps help to copy exported files to a 3rd party system hot folder. Please see more details on the Export in the User Manual as well as in the Export Knowledge Base.

13.3 Rest Based Integration

13.3.1 Service API

Since Version 7 the Service API is available. The Service API is a set of generic REST based APIs which provide read and write access to nearly all data which is described in the repository. Additionally to that, it also provides management functionality like triggering imports, exports, merges and other features.

We recommend to take a look at the [Service API documentation](#) (see page 428).

13.3.2 Custom REST Services

With version 6.0 we provide the needed [rest infrastructure](#) (see page 423) so anyone can write their own rest based services without the need to care about authentication or deployment of the service. Please take a look at [REST Service Infrastructure](#) (see page 423) for more details and a tutorial which will help you to write your own service.

13.4 Message Queue Based Integration

13.4.1 Message Queue API

The latest addition to our list of integration APIs the Message Queue API provides a resilient and flexible way to integrate with 3rd party systems as well as the Informatica BPM. By using the Apache Message Queue we can guarantee a safe and secure delivery of messages and responses as well as a sophisticated back pressure mechanism in cases of high load - so an overload of the system and it's modules is hardly possible. Our trigger framework provides the configuration UI for the admin user to define for which event in the system a message should be sent to which message queue.

13.5 REST Service Infrastructure



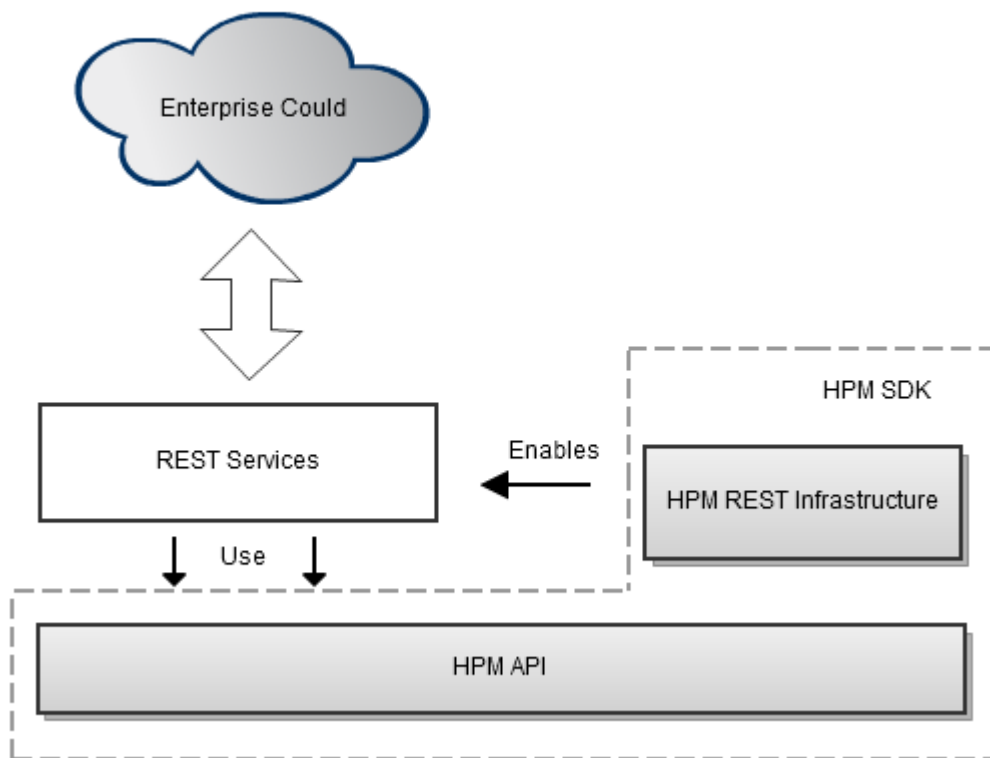
This article applies to the Product Manager Version 6 and following.

- [Overview](#) (see page 424)
- [Security](#) (see page 425)
- [Create your own REST service](#) (see page 425)
 - [REST Resource Example](#) (see page 426)

13.5.1 Overview

HPM server provides a standard way to contribute REST services based on the JAX-RS (JSR 311) standard. JAX-RS is a JEE standard and API to implement restfull services in Java. Thanks to standardization this is a perfect approach to system integration especially in enterprise landscapes. It allows to expose HPM data to the external systems using a well defined technology and thus provides a way to access HPM data by developers who are not familiar with HPM SDK and API.

HPM platform in Version 6.0 does not provide predefined REST services, rather it provides an infrastructure to develop and contribute services into HPM context. Future versions of the platform will contain a set of rest services to access and modify data in HPM.



The HPM platform provides REST infrastructure with the following components:

- JAX-RS implementation called *Jersey*, the reference implementation developed by Oracle. It contains all necessary classes and interfaces to implement rest services.
- Servlet container - Jetty 6. This container is integrated into the OSGi Equinox environment. Jetty properties like host, port and others can be configured in the `server.properties` file of HPM.
- HTTP and HTTPS connectors (provided by Jetty)
- Integration with HPM security and authentication.
- Extension point in HPM platform to which REST services implementations can be contributed. This architecture brings clear separation between HPM API and JAX-RS API and makes REST services development easy and clear for developers.

13.5.2 Security

The security of REST-based access to Product Manager is based on HTTP Basic Authentication combined with Product Manager's own authorization mechanisms. It is required to create a HPM user which has `internal` authentication mode (not external or mixed mode). For security reasons external authentication (Active Directory authentication) must not be used from outside. We also strongly recommend to activate HTTPS protocol in production environment because HTTP Basic Authentication does not encrypt user password which is sent over HTTP.

13.5.3 Create your own REST service

In order to create a REST services in HPM you need to do following steps

- Implement JAX-RS complaint service. Please consult Jersey documentation and other documentation how building JAX-RS services.
- Contribute JAX-RS resources to **com.heiler.ppm.rest.server** extension point which is exported by **com.heiler.ppm.rest.server** bundle

Extension point **com.heiler.ppm.rest.server** is very simple and accepts implementations of `ResourceProvider` interface. Implementation should have only one method `getResources()` which return s a list of services implementation (resources in terms of JAX-RS api).

Extension point contribution example:

Resource Provider contribution

```

1 <extension point="com.heiler.ppm.rest">
2   <Resource
3     class="com.heiler.ppm.custom.rest.CustomResourceProvider">
4   </Resource>
5 </extension>
```

Resource provider implementation

Supplier resource provider implementation

```

1 public class CustomResourceProvider implements ResourceProvider
2 {
3   @Override
4   public Collection< Class< ? >> getResources()
5   {
6     List< Class< ? >> result = new LinkedList< Class< ? >>();
7     result.add( ArticleResource.class );
8     return result;
9   }
```

```
10 }

```

13.5.3.1 REST Resource Example



You can find the complete source code of this example in the HPM 6.0 SDK. The project is called `com.heiler.ppm.custom.rest`

Find articles by EAN in a given catalog:

Article REST service example

```

1  @Path( "/catalogs/{catalogId: [0-9]+}/articles" )
2  public class ArticleResource
3  {
4      @GET
5      @Produces( MediaType.APPLICATION_XML )
6      public Articles findAll( @PathParam( "catalogId" ) Long catalogId,
7                              @QueryParam( "ean" ) String ean )
8                              throws CoreException, InterruptedException
9      {
10         Articles result = new Articles();
11         result.setArticles( findArticlesByEan( catalogId, ean ) );
12         return result;
13     }
14     private List< Article > findArticlesByEan( Long catalogId, String ean )
15     throws CoreException, InterruptedException
16     {
17         ReportResult report = findAsReport( catalogId, ean );
18         return loadByReport( report );
19     }
20     private List< Article > loadByReport( ReportResult report ) throws
21     CoreException, InterruptedException
22     {
23         FieldPath articleIdentifier = new FieldPath( "ArticleType.SupplierAid"
24 );
25         FieldPath shortDescriptionEN = new FieldPath(
26 "ArticleLangType.DescriptionShort" );
27         shortDescriptionEN.getEntityPath()
28             .setLogicalKeyValue( "ArticleLangType.LK.Language",
29 new Long( 7 ) );
30         ListModel listModel = loadArticleList( report, articleIdentifier,
31 shortDescriptionEN );
32
33         List< Article > articles = new ArrayList( listModel.size() );
34         for ( ListEntry listEntry : listModel )
35         {
36             Article article = new Article();

```

```

31         article.setId( listEntry.getEntityProxy()
32                        .getId() );
33         article.setIdentifier( ( String )
listEntry.getFieldValue( listModel.getColumn( articleIdentifier )
34
35         .getIndex() ) );
36         article.setShortDescription( ( String )
listEntry.getFieldValue( listModel.getColumn( shortDescriptionEN )
37
38         .getIndex() ) );
39         articles.add( article );
40     }
41     return articles;
42 }

```

Article data container:

Article data containers

```

1  @XmlElement
2  public class Articles
3  {
4      private List< Article > articles;
5      public Articles()
6      {
7          this.articles = new LinkedList< Article >();
8      }
9      public List< Article > getArticles()
10     {
11         return this.articles;
12     }
13     public void setArticles( List< Article > articles )
14     {
15         this.articles = articles;
16     }
17 }
18
19 ...
20
21 @XmlElement
22 public class Article
23 {
24     private long id;
25     private String identifier;
26     private String shortDescription;
27     public long getId()
28     {
29         return this.id;
30     }

```

```

31     public void setId( long id )
32     {
33         this.id = id;
34     }
35     public String getIdentifier()
36     {
37         return this.identifier;
38     }
39     public void setIdentifier( String identifier )
40     {
41         this.identifier = identifier;
42     }
43     public String getShortDescription()
44     {
45         return this.shortDescription;
46     }
47     public void setShortDescription( String shortDescription )
48     {
49         this.shortDescription = shortDescription;
50     }
51 }
52

```



Please, be careful when designing REST web services. Wrong design or incorrect use of rest-full principals may result a unmanageable solution which is difficult to use and integrate with other systems, which has poor performance and is not clear to other developers. Please remember the following principals of restfull services:

- REST services are stateless. It means that service implementations should not remember any kind of session between requests.
- In REST design everything is a resource. Every data structure received from the server represents an identifiable resource. Every request can be seen as a CRUD on a resource.
- REST services can read and modify resources and usually should not look like remote procedure call (RPC).

You can find a lot of information about designing REST service in a book "Restful Web Services" and "Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services".

13.6 REST Service API

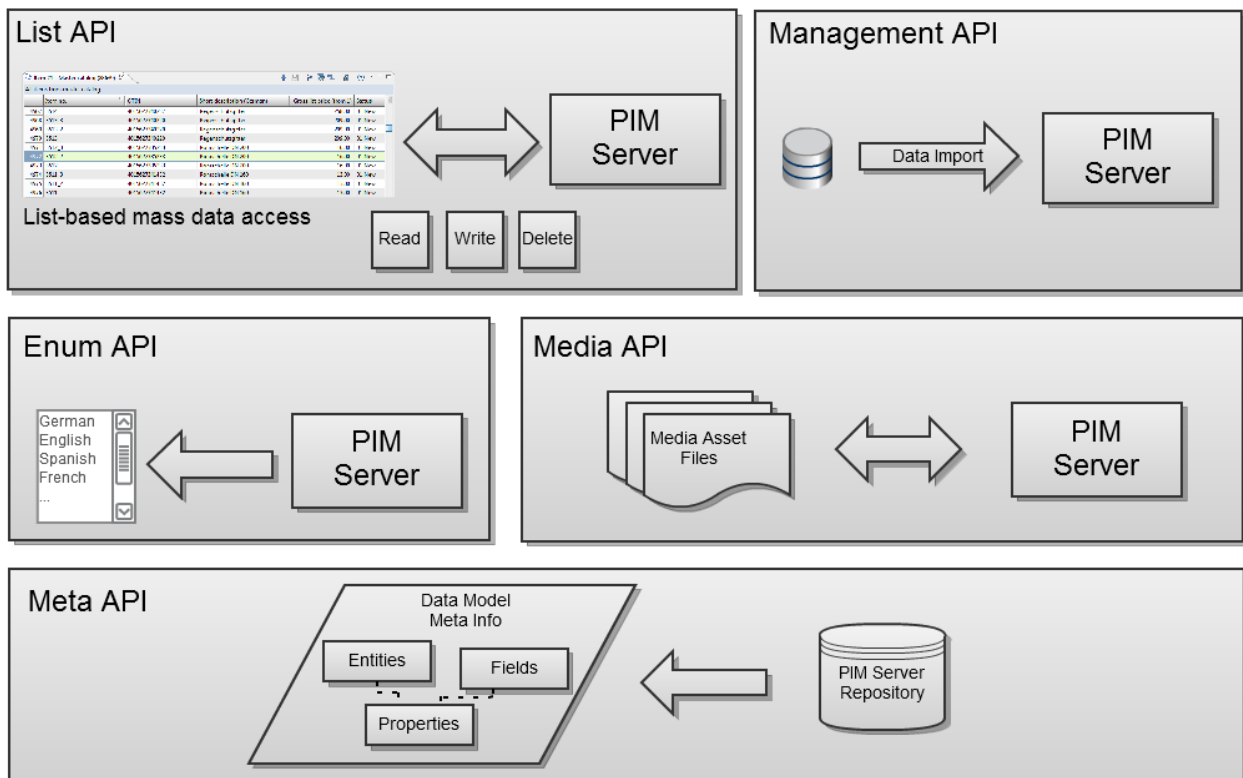
Based on the [REST Service Infrastructure](#) (see page 423) introduced in Version 6 we created a set of services which allows you to read and write almost any data which is described by the repository. In general we provide services to create, read, update, and delete lists of entity items with a configurable set of field values.

If you would like to know more about REST in general, start with the Dare Obasanjo's blog post, [Explaining REST to Damien Katz](#).

- [Available APIs](#) (see page 429)
- [List API](#) (see page 429)

- [Management API](#) (see page 430)
- [Enum API](#) (see page 430)
- [Meta API](#) (see page 430)
- [Media API](#) (see page 430)
- [Security API](#) (see page 430)
- [Object API](#) (see page 430)
- [Getting Started](#) (see page 430)
 - [Authentication](#) (see page 430)
- [General Information](#) (see page 430)
 - [Data Representation](#) (see page 430)
 - [Entity Item Reference](#) (see page 430)
 - [Field Qualification](#) (see page 430)
 - [Search Query Language](#) (see page 431)
 - [Rest Java Client](#) (see page 431)
 - [Rest API Versions](#) (see page 431)

13.6.1 Available APIs



13.6.1.1 List API

The [REST List API](#) (see page 463) provides searching, reporting, navigation and finally result listing of nearly all objects in the system. Based on a query a homogeneous list of objects of a certain type is returned. The

queries are parameterized. The resulting list is a two dimensional table with a given set of columns to control the informational character of the result.

13.6.1.2 Management API

The [REST Management API \(see page 703\)](#) provides access to control the server side management processes like Import, Merge, Export, and others. Typically these processes will be able to be started or scheduled, the process status can be evaluated and the process results can be requested and accessed.

13.6.1.3 Enum API

The Enumeration API provides read-only access to all enumerations in Product 360. The enum entry keys are used in field values as well as for the qualification of fields.

13.6.1.4 Meta API

The [REST Meta API \(see page 870\)](#) provides meta-information on all available Product Manager objects. It provides access not only to the objects, but also to their field including names, data types, constraints, etc. Using this API you can build powerful generic clients which may work with different installations of the Product Manager with different Repository configurations.

13.6.1.5 Media API

The REST Media API provides access to all kinds of media assets in the Product Manager.

13.6.1.6 Security API

The REST Security API provides access to securities of the Product Manager.

13.6.1.7 Object API

With Version 10.5 we introduce an optimized version of the [Read Object API \(see page 632\)](#) and a new [Write Object API \(see page 674\)](#). The usage of [Version 1 of the Read Object Api \(see page 595\)](#) is supported, but discouraged.

With this all typical CRUD (Create, Read, Update and Delete) operations can be performed for all repository based entities which support the object API.

13.6.2 Getting Started

Product Manager's REST APIs provide access to resources (data entities) via URI paths. To use a REST API, your application will make an HTTP/HTTPS request and parse the response. The Product Manager REST API uses JSON as its communication format, and the standard HTTP methods like GET, PUT, POST and DELETE. URIs for Product Manager's REST API resource have the following structure:

```
http://<host>:<port>/rest/<api-version>/<api>/<entity-identifier>;
```

- `<host>` : The Product Manager application server host.
- `<port>` : The http port which is configured in the [server.properties](#) (see page 428) file (see property `http.port` there), usually it's 1501.
- `<api-version>` : The version of the API. The current API version is V2.0.
- `<api>` : The specific api to call. See below for a list of the currently available APIs.
- `<entity-identifier>` : The unique identifier of the Product Manager's resource (also known as entity).

Most API's have a dynamic character since the actual resources which are available for each of those API's depend on the actual repository configuration of the Product Manager installation. To make life easier for everyone we implemented some dynamic documentation lookup which provides all the needed parts. The documentation is provided as rich html, or as machine readable JSON depending on the requested format (html/json). The first "level" of documentation is the list of available api's which can be obtained by `http://<host>:<port>/rest/V1.0/info`

13.6.2.1 Authentication

Every request to the REST API must be authenticated, anonymous access to the Product Managers resources is not possible. Additionally to that, the user which is used for the authentication must have the Service Logon action right assigned. The currently available authentication method is HTTP Basic. Most client software provides a simple mechanism for supplying a user name and password and will build the required authentication headers automatically. More details can be found at [REST Service Authentication](#) (see page 434).

13.6.3 General Information

13.6.3.1 Data Representation

The [REST Service API v10.5](#) (see page 428) transfers all data physically as UTF-8 strings, either as JSON or XML representation. How numbers, dates, times and so on are formatted is described in [REST Datatypes](#) (see page 436).

13.6.3.2 Entity Item Reference

Links to other objects, like other products or structure groups, are represented as entity item references. An entity item reference is a structured object containing the ID and the label.

13.6.3.3 Field Qualification

Qualified fields can be specified on different levels, for example when reading and writing data. Always the same syntax is used which is described in [REST Field Qualification](#) (see page 441).

13.6.3.4 Search Query Language

We have defined a language for search expressions. The language is currently used in the *bySearch* report. The syntax of the search language is described in [REST Search Query Language](#) (see page 447).

13.6.3.5 Rest Java Client

The Rest Service API of the Heiler Product Manager can be from every client technology which is able to handle HTTP requests. However, we expect that a lot of clients will be built based on some Java technology, and therefore we provide also a [REST Java Client](#) (see page 450) for the Rest Service API. The Java client is part of the PIM 7 distribution. Rest Client Java sources are also available. Therefor also Javadoc can be viewed or created.

13.6.3.6 Rest API Versions

All API endpoints contain the Rest API Version. There are two versions available:

- V1.0 is the first iteration and has been extended with several new endpoint over time
- V2.0 provides the same functionality as V1.0 but comes with different defaults and semantics. Changes have been made based on experiences in the field.

In general, clients should always use the latest available version. Older versions might become deprecated and be removed in the future.

List API changes in V2.0

Change	Description
formatData	<p>In V1.0, setting formatData to false didn't have an impact on fields with enumeration, like Article.CurrentStatus. When formatData is set to false,</p> <ul style="list-style-type: none"> ▪ In V1.0 the enumeration label is returned ▪ In V2.0 the enumeration key is returned.
includeLabels	<ul style="list-style-type: none"> ▪ New default value is false ▪ If set to false, root objects in the result also don't have a label anymore. In V1.0 this only affected fields with object references.
cacheId	<p>New default value in responses is no-cache for the case, the client doesn't request using caching.</p>

Change	Description

13.6.4 REST General Information

13.6.4.1 Monitoring

Monitor all REST requests in the log file

Add the following lines to the log4j.xml:

```
<Logger name="com.heiler.ppm.webservice" level="TRACE" />
```

After Server restart REST requests will be written to the server log file. Please use with care as the log file can grow very fast. Example output:

```
015-06-22 17:49:56,386 DEBUG [qtp1453109885-134] [ListResource] GET: http://
localhost:1501/rest/V1.0/list/Product2G/byCatalog?
fields=Product2G.Ac1Proxy,Product2G.CurrentStatus,Product2G.Id,Product2G.Manufacturer
AID,Product2G.ManufacturerName,Product2GLang.DescriptionLong(eng),Product2GLang.Descr
iptionShort(eng),Product2GPriceValueSales.Amount('Public',net_customer,USD,US,2014-07
-01,1.0),Product2G.ProductNo&orderBy&formatData=false&metaData=true&startIndex=0&page
Size=100&includeLabels=true
2015-06-22 17:49:56,386 DEBUG [qtp1453109885-134] [ListResource] Request headers:
2015-06-22 17:49:56,386 DEBUG [qtp1453109885-134] [ListResource] Host:
localhost:1501
2015-06-22 17:49:56,386 DEBUG [qtp1453109885-134] [ListResource] Accept:
application/json
2015-06-22 17:49:56,386 DEBUG [qtp1453109885-134] [ListResource] Authorization:
Basic cmVzdDpoZWlsZXI=
2015-06-22 17:49:56,386 DEBUG [qtp1453109885-134] [ListResource] Accept-
Language: en-US
2015-06-22 17:49:56,386 DEBUG [qtp1453109885-134] [ListResource] User-Agent:
Apache-HttpClient/4.3.3 (java 1.5)
2015-06-22 17:49:56,386 DEBUG [qtp1453109885-134] [ListResource] Connection:
keep-alive
2015-06-22 17:49:56,386 DEBUG [qtp1453109885-134] [ListResource] Accept-
Encoding: gzip,deflate
2015-06-22 17:49:56,386 DEBUG [qtp1453109885-134] [ListResource] Query parameters:
2015-06-22 17:49:56,387 DEBUG [qtp1453109885-134] [ListResource] startIndex: 0
2015-06-22 17:49:56,387 DEBUG [qtp1453109885-134] [ListResource] orderBy:
2015-06-22 17:49:56,387 DEBUG [qtp1453109885-134] [ListResource] formatData:
false
```

```

2015-06-22 17:49:56,387 DEBUG [qtp1453109885-134] [ListResource]      includeLabels:
true
2015-06-22 17:49:56,387 DEBUG [qtp1453109885-134] [ListResource]      pageSize: 100
2015-06-22 17:49:56,387 DEBUG [qtp1453109885-134] [ListResource]      metaData: true
2015-06-22 17:49:56,387 DEBUG [qtp1453109885-134] [ListResource]      fields:
Product2G.AcIProxy,Product2G.CurrentStatus,Product2G.Id,Product2G.ManufacturerAID,Pro
duct2G.ManufacturerName,Product2GLang.DescriptionLong(eng),Product2GLang.DescriptionS
hort(eng),Product2GPriceValueSales.Amount('Public',net_customer,USD,US,2014-07-01,1.0
),Product2G.ProductNo
2015-06-22 17:49:56,387 DEBUG [qtp1453109885-134] [ListResource]      User: rest
2015-06-22 17:49:56,387 DEBUG [qtp1453109885-134] [ListResource]      Locale: en_US
2

```

13.6.4.2 REST Service Authentication

An overview with examples on the various authentication methods supported by the REST Infrastructure of the Product Manager. All authentication methods are implemented as core functionality of the REST/ WebService infrastructure. This means that this authentication will automatically be active for your own REST Services as well as for the [REST Service API](#) (see page 428) which comes out of the box.

- [Basic Authentication](#) (see page 434)
 - [Simple example](#) (see page 434)
 - [Username / Password Basic Authentication](#) (see page 435)
 - [Token Basic Authentication](#) (see page 435)
- [OAuth](#) (see page 436)

Basic Authentication

Product Manager allows REST clients to authenticate themselves with a user name and password using basic authentication .



Basic authentication is unsafe as long as it's not combined with the SSL protocol, since the username and password are transmitted more or less in plain text!

Simple example

Most client software provides a simple mechanism for supplying a user name and password and will build the required authentication headers automatically. For example you can specify the -u argument with curl as follows

```
curl -D- -u myUsername:myPassword -X GET -H "Content-Type: application/json" http://
hpm.heiler.com:1501/rest/V1.0/list/suppliercatalog/all
```

Username / Password Basic Authentication

If you need to you may construct and send basic auth headers yourself. To do this you need to perform the following steps:

1. Build a string of the form username:password
2. Base64 encode the string
3. Supply an "Authorization" header with content "Basic " followed by the encoded string, e.g. "Basic YWRtaW46YWRtaW4="

```
curl -D- -X GET -H "Authorization: Basic bXlvc2VybmFtZTpteVBhc3N3b3Jk" -H "Content-Type: application/json" http://hpm.heiler.com:1501/rest/V1.0/list/suppliercatalog/all
```

Token Basic Authentication

Instead of supplying username/password clients can also use a token to authenticate calls. This token can be retrieved the following ways:

- When running inside the Product 360 Server / Client by using the API: `com.heiler.ppm.security.core.auth.Authenticator.createTokenAsString()`. This token is valid for one hour.
- When already authenticated, a token can also be retrieved via REST by sending a POST to `/rest/V1.0/manage/system/security/token`. For further details, see chapter [REST System API \(see page 777\)](#). This token is valid for one hour.
- When using SAML for authentication (see chapter SAML Configuration), a token can be retrieved for the user contained in a SAML Assertion resp. SAML Response. This is done via REST by sending a POST to `/rest/V1.0/manage/system/security/tokenForSaml`. For further details, see chapter [REST System API \(see page 777\)](#). The token validity period can be configured, and is one day by default.

The token then can be passed base64-encoded as Basic Authentication header. Example:

```
GET /rest/V1.0/list/Article/byCatalog
Authentication: Token
MkZDRDvDMkMyRjEwQTk0OTBFMTAwOEVDNjhFNTREOEeyREMyMUMzQzNGN0I5Q0RGMDVBnkE1MzVBODlFMkFCR
jA1MzFGMzE5Q0E0QkM5OTczRjAyM0QyQzc2RUewQTNGODQ20DA3MTQyMjM5NjdBOTIyM0UyNTlCQzNDOTI1Q0
MwQkM2MEIyRjk5RTg5MTBDRDFfNERERUMzNENGM0MzNUY2OEY3NTM0MTdGMUM5MzU1NzQwRjJFMUVDMkJDNTJ
CMUE5RTg4OUFFMjcZRUm0MDc0NURDNDJCQTMyoERGRjVGRDNGRjY1ODk3RDk2NEVDMkE2MEI5RTcwOTI4ODlG
MERCQTM2MzdFNEIzOTAxMDYzOTQ4MjM4RjkxODcwQjk3RTgwOEZDQzZGMTIwMkIxNkIzQThCRkZERjIwMDkxO
UI5MDZBNzAwRUM3MzM1QjQ1RTY5QkM4NUM3MEEzMDY2N0U4OTkxRTQxQ0YwMUREN0FFOUY5QjJEMjk0NUE2Mj
Q2RjgyN0IyNUUzRUQ2MEQxNjVDN0U1OUQyOERENDcwNUQzRkU1QTMymjlgRUZCQUQzNjRDNTThGNjg2NTQ2QjA
1OEQzQTdEMzE1QkQ4M0QzRjLFQUFCQUVENTM0RDE0RTFGMzdcCNTkxRDQ5REVENTM1NjM2RkQyNTFGQ0NGNDRG
NUY=
```

OAuth

Not yet supported.

13.6.4.3 REST Datatypes

Datatypes

The [REST Datatypes](#) (see page 436) transfers all data physically as UTF-8 strings, either as JSON or XML representation. The following list gives an overview on the data types which are used throughout the whole REST Service API.

STRING

Definition	Unicode Characters
Pattern	
Example	"Hello World!", "Heute ist ein schöner Tag", "世界您好!"
Corresponding Java Datatype	java.lang.String

INTEGER

Definition	Integer Numbers (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
Pattern	[+/-] 0-9*
Example	4711
Corresponding Java Datatype	java.lang.Long

DECIMAL

Definition	<p>Signed decimal number. Please see the properties of the parameter/field for details on the precision and the allowed number of fraction digits.</p> <p>The string representation consists of an optional sign, '+' or '-', followed by a sequence of zero or more decimal digits ("the integer"), optionally followed by a fraction</p> <p>The fraction consists of a decimal point followed by zero or more decimal digits.</p> <p>The string must contain at least one digit in either the integer or the fraction.</p>
Pattern	[+/-] 0-9*[.0-9*]
Example	100.90
Corresponding Java Datatype	java.math.BigDecimal

DATE

Definition	Calendar Date (12/30/1899 to 12/31/9999)
Pattern	yyyy-MM-dd
Example	2012-05-01 (first of May in 2012)
Corresponding Java Datatype	java.sql.Date

TIME

Definition	Time (00:00:00 to 24:00:00)
Pattern	HH:mmZ

Example	12:05 (five minutes after 12 o'clock noon)
Corresponding Java Datatype	java.sql.Time

DATETIME

Definition	Date and Time in one value	
Pattern	Input	Output
	yyyy-MM-dd'T'HH:mm:ss	yyyy-MM-dd'T'HH:mm:ss:SSSZ
	yyyy-MM-dd'T'HH:mm:ssZ	
	yyyy-MM-dd'T'HH:mm:ss:SSS	
	yyyy-MM-dd'T'HH:mm:ss:SSSZ	
Example	2012-05-01T12:00:00 2012-05-01T12:00:00-0700 2012-05-01T12:00:00:000 2012-05-01T12:00:00:000-0700	
Corresponding Java Datatype	java.sql.Timestamp	

BOOLEAN

Definition	Boolean value - either true or false. Note: Fields which are not mandatory might also have an empty string which means no value which should not be interpreted as "false" or "true". It's just, "not defined".
Pattern	true / false
Example	true, false
Corresponding Java Datatype	java.lang.Boolean

ENTITY_ITEM

Definition	<p>A Product Manager object which is described in the Meta API (see page 870). In case entity item's are part of a result object from a service call, they will be serialized as separate object structure. In case you need to provide an entity item as a parameter, a simpler string representation is mostly enough.</p> <p>As part of a result object an entity item is rendered like this:</p> <pre>"object": { "id": "86@1064", "label": "A1" }</pre> <p>If you do not need the human readable label of the entity items, you can disable the label generation by specifying <code>includeLabels</code> with <code>false</code>. See also the REST List API Read for Root Entities (see page 472) page.</p>
-------------------	--

Pattern	<p>Entity Item ID Syntax:</p> <p>Entity Item without a container item:</p> <p>'<external_identifier> <internal_id></p> <p>Entity Item with a container item:</p> <p>'<external_identifier> <internal_id>@'<container_external_identifier> <container_internal_id></p> <p><i>Please note that you need to escape single quotes in case they are part of the external identifier or container external identifier with a backslash</i></p>
Example	<p>'SomeItem'@'SupplierCat1'</p> <p>12345@5</p> <p>'SomeSupplier'</p> <p>42</p> <p>'SomeItem\'WithASingleQuote'@'MASTER'</p>
Corresponding Java Datatype	com.heiler.pim.webservice.client.EntityItemReference

MIME_VALUE

Definition	<p>A reference to a mime file. In case mime values are part of a result object from a service call, they will be serialized as separate object structure.</p> <p>A mime value has a label. The label is shown in the UI in case a preview picture is not used there. The relative file path defines the location of the file inside of the zip archive when you download or upload the mime values</p> <p>The label and mime type are optional in case you want to write a mime value. They will be extracted from the <code>relativeFilePath</code> if omitted.</p> <p>Please note that the relative file path of a mime value will change once it's uploaded and saved as Product 360 makes sure that every mime value is unique in the system. If you retrieve the mime value again after the upload, the <code>relativeFilePath</code> will be different. It should always be treated as a black box.</p>
-------------------	---

Pattern	<i>No Simple string representation, it's always rendered as JSON/XML structure</i>
Example	<pre>{ "label": "examplePicture.jpg", "mimeType": "image/jpg", "relativeFilePath": "dir1\\dir2\\dir3\\examplePicture.[sequenceToMakeFileNameUnique].jpg" }</pre>
Corresponding Java Datatype	<code>com.heiler.pim.webservice.client.MIMEValueReference</code>

ANY

Definition	ANY can be one specific datatype like STRING,DATETIME, DECIMAL, ENTITY_ITEM or BOOLEAN. A field with ANY as datatype must have a specific implementation to figure out which of the specific datatypes mentioned before is actually taking effect.
Pattern	see at the specific datatype
Example	see at the specific datatype
Corresponding Java Datatype	<code>java.lang.Object</code>

13.6.4.4 REST Field Qualification

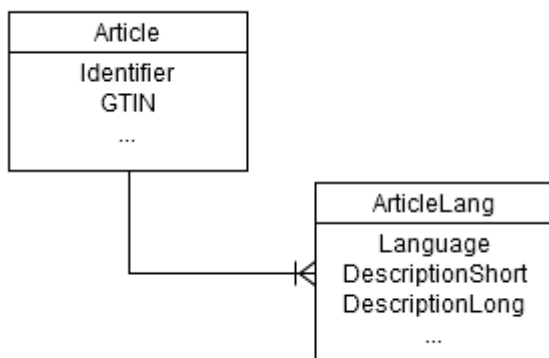
Several services need fully qualified fields in one way or the other. The syntax to qualify fields is documented on this page

- [Syntax](#) (see page 442)
- [Logical Key Value Definition](#) (see page 443)
 - [With Enumeration](#) (see page 443)
 - [Without Enumeration](#) (see page 443)
 - [Using default values](#) (see page 445)

- [Using wildcard matching \(see page 446\)](#)

Product Manager defines many business objects which are described in the Product Manager repository. The [repository \(see page 79\)](#) can be seen as a storage for all meta-data regarding the business models. All objects and all attributes of those objects are described in there. Additionally to that, also the relation between an object and its child objects is described in the repository. You can find a detailed description about the Repository [here \(see page 79\)](#).

An example would be the object "Article" and its child object "ArticleLang". An Article can have multiple "ArticleLang" records, one for each language. Therefore, the language is the attribute which makes the "ArticleLang" records unique for a single item.



To "fully qualify" a field means, that you not only must define which field (by the field's identifier), but also all values for all attributes which identify the children uniquely. In this example, just the language.

Syntax

Fully qualified field: `<Field Identifier>[(<first logical key value>,...,<n-logical key value>)]`

Multiple Field Qualification: `<Fully qualified field>[, <Fully qualified field>]`

Element	Definition
Field Identifier	<p>Basically the unique field identifier as given in the custom section of the repository. These identifiers typically but not necessarily start with the entity identifier separated by a "." from the field's name. This makes the field unique in case of the same field supported by different data entities. Samples are:</p> <ul style="list-style-type: none"> • Article.SupplierAID • Article.EAN • Article.ManufacturerName

Element	Definition
Logical Key Value	<p>The qualification value for the first, second, third and so on logical key. Which logical key is on which position is defined by the hierarchy and the order attribute of the repository logical key type. In case no order attribute is defined in the repository, the order is defined by the relative location in the repository xml file.</p> <p><Parent LK 1, ParentLK 2, Own LK 1, Own LK 2></p>

Logical Key Value Definition

The definition of the logical key value depends on the datatype of the logical key (or it's respective field) and whether the logical key has an enumeration or not.

With Enumeration

You can define everything which can be interpreted by the enumeration as a synonym to one of it's keys. This can be done either as a constant value, or as a string (constants are not allowed to have spaces in it, use strings in this case!)

For example: `ArticleLang.DescriptionShort(en)` , `ArticleLang.DescriptionShort(eng)` , `ArticleLang.DescriptionShort(English)` would be just the same.

Furthermore the key of an enumeration can be used if it's data type is not an entity item, like buyers, suppliers, units etc.

For example: `ArticleLang.DescriptionShort(9)`

A special situation would be a logical key which has an enumeration and it's data type is entity item. In this case you can either specify the actual entity item with the entity item syntax, as well as

a string or constant which is then parsed with the enumerations synonyms.



Some enumerations are case sensitive regarding their synonyms. Which one those are is defined in the enumeration itself. You're always on the safe side if you treat all of them as case-sensitive.

Without Enumeration

Logical keys without enumeration must be defined depending on their datatype. Please see the [REST Datatypes](#) (see page 436) overview for details.

Datatype	Format	Example
Date (without time!)	YYYY-MM-DD	2012-06-22 which is the 22nd of June in 2012
Time (without date!)	HH:mm:ss	13:15:05 which is quarter past one, pm and five seconds
Date and Time	YYYY-MM-DD HH:mm:ss	2012-06-22 13:15:05
Integer	+## or -##	+1247 or -1247
Decimal	+##.## or -##.##	+1.45 or -1.45 (no thousand separator, use dot as comma separator!)
Boolean	true or false	true or false
String	"any string" or "any \"important\" string"	"My Attribute Name" escape the quotation marks by a backslash! E.g. "Size in \" (meaning: size in inch)
Constant	any string a-Z followed by zero or more 0-9	eng, de, public

Datatype	Format	Example
ENTITY_ITEM	<p>Entity Item without a container item: '<external_identifier> <internal_id></p> <p>Entity Item with a container item: '<external_identifier> <internal_id>@' <container_external_id> <container_internal_id></p> <p>Please don't forget use single quotes when you want to specify the external identifier of the entity item!</p>	<p>'SomeItem'@'SupplierCat1' 12345@5</p> <p>'SomeSupplier' 42</p>
Dynamic Value	<p>A dynamic value is a placeholder which will be resolved while parsing the field qualification</p> <p>Syntax: <code>\${Identifier}</code></p> <p>Note: additional named values can be contributed to the PIM server using the extension point: <code>com.heiler.ppm.value.core.namedValues</code></p>	<p>Currently available named values:</p> <ul style="list-style-type: none"> • <code>\${ CurrentDate}</code> = the current date (without time) • <code>\${CurrentTime}</code> = the current time (without date) • <code>\${CurrentTimestamp}</code> = the current date and time

Examples

The short description of an article in english: ArticleLang.DescriptionShort(eng)

The public customer price value, in Euro for Germany, Valid on the 06/22/2012, when I order one: ArticlePriceValueSales.Amount(public,net_customer,eur,de,2012-06-22,1.00)

Using default values

If the repository defines a default value for the logical key then the default value can be referenced in the field qualification using the following notation:

`${Default}`

Note that the resulting fully qualified field may change when the definitions in the repository change.

Using wildcard matching

If no value can be defined as qualification, it is possible to use the

`${Empty}`

qualifier.

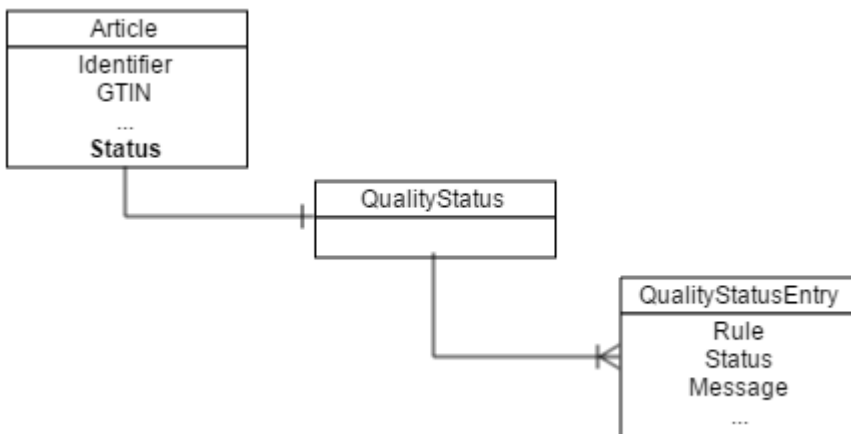
13.6.4.5 REST Transition Fields

The [REST List Read API](#) (see page 470) supports so-called transition fields. The syntax for such transition fields is documented on this page

- [Syntax](#) (see page 446)
- [Examples](#) (see page 447)


The [PIM repository](#) (see page 79) supports proxy fields which reference other repository entities. When fields of such referenced entities are accessed in the context of the referencing entity, they are called *transition fields*. Transition fields have a source and a target field, which can in turn be transition fields. Therefore it is possible to traverse recursively to a target field. All fields along this transition path need to be [fully qualified](#) (see page 441).

An example would be the object "Article" and it's field "Article.Status". Each item has a quality status, whose content resides in the "QualityStatus" object. If we want to obtain the quality status of a certain item for a specific data quality rule, we need to use a transition field with target "QualityStatusEntry.Status" and proper qualification for the "rule" logical key.



Syntax

Transition field: `<Fully qualified field> -> ... -> <n-Fully qualified field>`

Element	Definition
Fully qualified field	<p>A fully qualified field as described in REST Field Qualification (see page 441). The field on the left of the arrow (->) must be a proxy transition field pointing to another repository object, as e.g.</p> <ul style="list-style-type: none"> Article.Status ArticleStructureMap.StructureGroup ArticleLogistic.PackageWeightUnitID <div style="border: 1px solid #f9e79f; padding: 10px; margin-top: 10px;"> <p> Make sure to have a space before and after the arrow, otherwise it will be interpreted as part of the field name, which most likely will not be correct.</p> </div>

Examples

- The quality status of an item of rule "CheckValidGTIN":
Article.Status -> QualityStatusEntry.Status(CheckValidGTIN)
- The quality status of a structure group of rule "CheckValidName" where an item is assigned to:
ArticleStructureGroupMap.StructureGroupProxy('ETIM3.0', 'shoes'@'ETIM3.0') -> StructureGroup.Status -> QualityStatusEntry.Status(CheckValidName)

13.6.4.6 REST Search Query Language

The HSQ syntax directly reflects the search expression API explained in [Entity Search](#) (see [page 151](#)) - so the same expressions are possible as if you build complex search queries the programmatic way using the corresponding API. It's main design goal was to provide an easy to understand and easy to use query syntax which can also be used by end users of the Product Manager.

Syntax
<pre> HSQ Query: [<CONDITION>]+ HSQ Condition: <TERM> [AND OR <CONDITION>]* [NOT]? (<CONDITION>) HSQ Term: [NOT]? <FIELD QUALIFICATION CHARACTERISTIC> <OPERATOR> <VALUE LITERAL> [NOT]? <FIELD QUALIFICATION CHARACTERISTIC> IN (<VALUE LITERAL> [, <VALUE LITERAL>]*) </pre>

Conditions/terms can be connected with AND / OR, whereas the AND has higher priority and thus is evaluated first. Evaluation priority can be influenced by building complex conditions by means of parenthesis (possible around conditions as well as around terms). NOT is used for negating terms as well as complex conditions (where parenthesis are needed, of course).

See again example from code block above:

```
Article.EAN in ("123456789", "987654321", "abcdefghij") AND not (Article.Identifier = "abc" OR Article.Identifier equals "xyz")
```

```
characteristic(9342) = "44"
```

```
characteristic(9347) = "22"&catalog=1011
```

Field Qualification

The HSQ expression is built with a fully qualified field. Please see the [REST Field Qualification \(see page 441\)](#) documentation on the used syntax here. Note, only a single field is allowed in the context of the search query.

Characteristic

Please see the [REST Characteristic \(see page 452\)](#) documentation chapter on the used syntax here. Note, only a single characteristic is allowed in the context of the search query.

Operators

Operators are interpreted case insensitive, thus " equals " and " EQUALS " and " eQuAlS " are, equal 😊

Operator	Allowed Datatypes	Description
equals	String	Value must be exactly the same as the corresponding field content
equalsIC	String	Value must be the same as the corresponding field content, upper and lower case differences will be ignored
contains	String	Field content must contain the given string in the same case
containsIC	String	Field content must contain the given string, but the case of the content is ignored
startsWith	String	Field content must start with the given string in the same case

Operator	Allowed Datatypes	Description
startsWithIC	String	Field content must start with the given string but the case of the content is ignored
wildcard	String	Value can consist wildcards like '%' (one or more characters) and '_' (single character)
wildcardIC	String	Value can consist wildcards but the case of the content is ignored
=	all	Value must be the same as the corresponding field content
~	String	Value can consist wildcards like '%' (one or more characters) and '_' (single character)
!~	String	Value can consist wildcards like '%' (one or more characters) and '_' (single character)
!= / <>	all	Field content must be unequal to the provided value
>	String, Number, Datetime	The field content must be greater then the given value according to the "natural order" of the value
>=	String, Number, Datetime	The field content must be greater or equal as the given value according to the "natural order" of the value
<	String, Number, Datetime	The field content must be less then the given value according to the "natural order" of the value

Operator	Allowed Datatypes	Description
<=	String, Number, Datetime	The field content must be less or equal to the given value
in	all	Field content must be equal to one of the provided values in parenthesis
is empty	all	Field content must be null or empty string



While using wildcard/ ~ operators make sure to escape any wildcard character using '\\'. For eg. "\\%" or "_"

Value Literal

The textual representation of the value is described in the [REST Datatypes](#) (see page 436). Multiple values (comma separated in parenthesis) can only be used for the "in" operator.

In general, the value literal's type must be compatible to the fields's data type where the field qualification points to. Implicit conversion is currently only performed when field destination type is of DECIMAL and provided value literal is of INTEGER - in all other cases the types have to be the same.

Dynamic Value

A value literal can also be a dynamic literal.

A dynamic value is a placeholder which will be resolved while parsing the field qualification

Syntax: `${Identifier}`



Additional named values can be contributed to the PIM server using the extension point: `com.heiler.ppm.value.core.namedValues`

Currently available named values:

- `${CurrentDate}` = the current date (without time)
- `${CurrentTime}` = the current time (without date)
- `${CurrentTimestamp}` = the current date and time

13.6.4.7 REST Java Client

i The Rest Service API of the Heiler Product Manager can be from every client technology which is able to handle HTTP requests. However, we expect that a lot of clients will be built based on some Java technology, and therefore we provide also a [REST Java Client](#) (see page 450) for the Rest Service API. The Java client is part of the PIM 7 distribution. Rest Client Java sources are also available. Therefor also Javadoc can be viewed or created.

Example

Below you can find a short example on how to use the client. The `RestClientExample` class is also part of the rest client API package.

Usage example for the client API

```

1  /**
2   * Three arguments needed in the following order
3   * <p>
4   * - Basic Url, e.g.: "http://pim.heiler.com/rest"<br/>
5   * - Username, e.g.: MyUserName <br/>
6   * - Password, e.g.: MyPassword <br/>
7   * </p>
8   * <code> RestClientExample.main http://pim.heiler.com/rest MyUserName
9   * MyPassword </code>
10  */
11  public static void main( String[] args )
12  {
13      String basicUrl = extractBasicUrl( args );
14      String userName = extractUserName( args );
15      String password = extractPassword( args );
16      try
17      {
18          //Create and login to the client...
19          RestClient restClient = new RestClient();
20          restClient.loginWithBasicAuth( basicUrl, userName, password,
21          Locale.ENGLISH );
22          //Create and parameterize the report query you want to execute...
23          EntityItemReference masterCatalog =
24          EntityItemReferenceFactory.createByIdentifier( "MASTER" ); //$NON-NLS-1$
25          ReportQuery reportQuery = new ReportQuery( "byCatalog" ).addParamete
26          rValue( "catalog", masterCatalog ); //$NON-NLS-1$ //$NON-NLS-2$
27          //Create and parameterize the list request
28          ListReadRequest listReadRequest =
29          restClient.createListReadRequest();
30          EntityItemTable resultTable = listReadRequest.setFields(
31          "Article.SupplierAID" ) //$NON-NLS-1$
32          .setPageSize( 100 )
33          .setStartIndex( 0 )

```

```

28                                     .getRootItems( "Article
29   ", reportQuery ); //$NON-NLS-1$
30   for ( EntityItemTableRow row : resultTable )
31   {
32       List< Object > values = row.getValues();
33       String currentSupplierAid = ( String ) values.get( 0 );
34       System.out.println( currentSupplierAid );
35   }
36   catch ( RestClientException e )
37   {
38       e.printStackTrace();
39   }
40 }

```

13.6.4.8 REST Characteristic

- [Characteristic Operand](#) (see page 452)
 - [Syntax](#) (see page 452)
 - [Usage examples](#) (see page 453)
- [DESCENDANT Operator](#) (see page 458)
 - [Syntax](#) (see page 459)
 - [Limitations](#) (see page 459)
 - [Usage Examples](#) (see page 460)
- [Limitation](#) (see page 463)

In this chapter describes how to use characteristics in the Service API. It introduces the characteristic operand and the DESCENDANT operator.

Characteristic Operand

Syntax

When searching for items using characteristic record values the term `characteristic` with parenthesis has to be used. Both qualifications for the characteristic are **optional**.

`characteristic(<Entity Item>,<Language>)`

Element	Definition
Entity Item	<p>Characteristic entity item without a container item:</p> <p>'<external_identifier> <internal_id></p> <p>Please don't forget to use single quotes when you want to specify the external identifier of the characteristic!</p>

Element	Definition
Language	<p>The value for the language qualification can either be the key or a synonym as defined in the Enum.Language of the repository.</p> <p>Example:</p> <p>To qualify the German language the value 7 or "DE" can be used equally.</p>

So valid examples for the characteristic term are:

```
characteristic()
```

```
characteristic('Ingredients')
```

```
characteristic(156335)
```

```
characteristic(156335, "DE")
```

```
characteristic(156335, 7)
```



It is not possible to search for any characteristic with a specific language.

```
characteristic("DE")
```

Usage examples

Searching for items which have a specific characteristic and a specific value

This call will return all items which have the string value "Bean" for the characteristic `Ingredient`. Note that the characteristic has the datatype `TEXT` and therefore the characteristic record values are of type `String`.

```
http://localhost:1512/rest/V1.0/list/Article/bySearch?
query=characteristic('Ingredient') = "Bean"
```

We have these items with characteristic records:

ItemIdentifier	Characteristic	Characteristic record value
Item1	Ingredient	"Salt"
Item2	Ingredient	"Bean"
Item3	Ingredient	"Egg"
Item4	BakingMaterials	"Flour"

The query above will only return **Item2**.

When searching for string values it is also possible to use specific string operators like : `equalsIC` , `equals` , `contains` , `startsWith` , `contains` ,...etc.

All possible operators can be found in the chapter [REST Search Query Language](#) (see page 447)

```
http://localhost:1512/rest/V1.0/list/Article/bySearch?
query=characteristic('Ingredient') startsWith "Bean"
```

We have these items with characteristic records:

ItemIdentifier	Characteristic	Characteristic record value
Item1	Ingredient	"Salt"
Item2	Ingredient	"Beans"
Item3	Ingredient	"Egg"
Item4	BakingMaterials	"Flour"

The query above will only return **Item2**.

When searching for all items which have a characteristic record for the characteristic `Ingredient` .

```
http://localhost:1512/rest/V1.0/list/Article/bySearch?query=not
characteristic('Ingredient') is empty
```

We have these items with characteristic records:

ItemIdentifier	Characteristic	Characteristic record value
Item1	Ingredient	"Salt"
Item2	Ingredient	"Beans"
Item3	Ingredient	"Egg"
Item4	BakingMaterials	"Flour"

The query above will return **Item1,Item2,Item3**.

Searching for items which have specific characteristics and specific values

This call will return all items which have the value 15678 as a characteristic record for the characteristic "Numeric" and also have the value "2017-12-31" for the characteristic "TerminationDate"

<http://localhost:1512/rest/V1.0/list/Article/bySearch?>

`query=characteristic('Numeric') = 15678 and characteristic('TerminationDate') = "2017-12-31"`

We have these items with characteristic records:

ItemIdentifier	Characteristic	Characteristic record value		Characteristic	Characteristic record value
Item1	Numeric	12345		Termination Date	2017-12-31
Item2	Numeric	15678		Termination Date	2017-12-31
Item3	Numeric	15678		Termination Date	2017-11-20

ItemIdentifier	Characteristic	Characteristic record value		Characteristic	Characteristic record value
Item4	Numeric	1335		Termination Date	2018-12-31

The query above will only return **Item2**.

Searching for items in a specific language

This call will return all items which contain the value "soup recipe" for the characteristic "Description" in English

```
http://localhost:1512/rest/V1.0/list/Article/bySearch?
query=characteristic('Description',9) contains "soup recipe"
```

```
http://localhost:1512/rest/V1.0/list/Article/bySearch?
query=characteristic('Description',en) contains "soup recipe"
```

Both statements are equally valid.

We have these items with characteristic records:

ItemIdentifier	Characteristic	Language	Characteristic record value
Item1	Description	English	"soup without recipe"
Item2	Description	English	"this is a soup recipe"
Item3	Description	German	"Das ist ein Rezept für Suppe"
Item4	BakingMaterials	French	"soup recipe"

The query above will only return **Item2**.

Searching for items with lookup values

This call will return all items where the lookup value "Mushroom" of the lookup "Ingredients" is used regardless of the characteristic.

When trying to search by lookup values, it is required to use the entity proxy

format: 'EXTERNAL_IDENTIFIER_OF_LOOKUP_VALUE'@'EXTERNAL_IDENTIFIER_OF_LOOKUP'

```
http://localhost:1512/rest/V1.0/list/Article/bySearch?query=characteristic() =
'Mushroom'@'Ingredients'
```

We have these items with characteristic records:

ItemIdentifier	Characteristic	Lookup of Characteristic	Characteristic record value
Item1	CookingIngredients	Ingredients	Salt
Item2	CookingIngredients	Ingredients	Mushroom
Item3	CookingIngredients	Ingredients	Egg
Item4	ToxicIngredients	Ingredients	Mushroom

The query above will return **Item2** and **Item 4**.

When specifying a characteristic, it is possible to use the lookup value identifier without the lookup identifier:

```
http://localhost:1512/rest/V1.0/list/Article/bySearch?
query=characteristic('CookingIngredients') = 'Mushroom'
```

The query above will only return **Item2**.

All other operators can be used as well just like working with the fieldpath:

```
http://localhost:1512/rest/V1.0/list/Article/bySearch?
query=characteristic('Ingredient') in
( 'Mushroom'@'Ingredients', 'Egg'@'Ingredients' ) and not
characteristic('TerminationDate') > 2017-12-31T15:00:00
```

We have these items with characteristic records:

ItemIdentifier	Characteristic	Lookup of Characteristic	Characteristic record value		Characteristic	Characteristic record value
Item1	Ingredient	Ingredients	Mushroom		Termination Date	2017-12-28 17:30:00
Item2	Ingredient	Ingredients	Mushroom		Termination Date	2020-12-12 15:00:00
Item3	Ingredient	Ingredients	Egg		Termination Date	2017-12-05 15:00:00
Item4	BakingMaterials	Ingredients	Flour		Termination Date	2017-12-31 13:37:00

The query above will only return **Item1** and **Item3**.

DESCENDANT Operator

Sometimes it is important to find items with certain values. For example to find all items containing sugar won in Brazil. You don't want items containing sugar from India or meat from Brazil in your result set. In order to maintain such item data, you probably modeled a characteristic "Ingredient" containing the values "Sugar" or "Meat" and a characteristic "Country of origin" which is a descendant of the characteristic "Ingredient" in the characteristic hierarchy. Using the characteristic operand from above you would find items containing the ingredient sugar and items with Brazil as country of origin. In order to tell the system that you only want items having the two values in a direct hierarchy, you can use the DESCENDANT operator.

Syntax

Operator	Allowed datatypes	Description
DESCENDANT	all datatypes except MIME	This operator searches for characteristics having a hierarchical relationship.
>>		<p>Operands may be all binary expressions containing a characteristic except connectors like AND and OR.</p> <p>The left operand must be an expression containing a characteristic which is higher in the characteristic hierarchy than the characteristic contained by the right operand.</p> <p>Note: The left operand must not necessarily contain the root characteristic, but it must be higher in the characteristic hierarchy than the characteristic contained by the right operand.</p>

So valid examples for the DESCENDANT operator are:

```
Operand1 DESCENDANT Operand2
```

```
Operand1 DESCENDANT Operand2 DESCENDANT Operand3
```

```
Operand1 >> Operand2
```

```
Operand1 >> Operand2 >> Operand3
```

```
Operand1 >> NOT(Operand2) >> Operand3
```

```
Operand1 >> Operand2 DESCENDANT Operand 3
```

The following examples are **not** valid:

```
Operand1 >> Operand2 AND Operand3 >> Operand 4
```

```
Operand1 >> Operand2 OR Operand3 >> Operand 4
```

Limitations

It is **not** possible to use the same characteristic more than once in one DESCENDANT expression. So the following example is **not** valid:

```
characteristic( 'ABC' ) equals otherValue DESCENDANT characteristic( 'ABC' )
equals otherValue DESCENDANT characteristic( 'ABC' ) equals otherValue
```

The descendant operator **cannot** be negated. So the following example is **not** valid:

```
NOT (characteristic( 'ABC' ) equals value DESCENDANT characteristic( 'ABCD' )
equals otherValue)
```

Usage Examples

Rest call

```
GET http://localhost:1512/rest/V1.0/list/Article/bySearch?
query=characteristic( 'Ingredient' ) equals 'SUGAR'@'Ingredients' DESCENDANT
characteristic( 'CountryOfOrigin' ) equals 'BRAZIL'@'OriginCountries'
```

Rest call - Alternative syntax with >>

```
GET http://localhost:1512/rest/V1.0/list/Article/bySearch?
query=characteristic( 'Ingredient' ) equals 'SUGAR'@'Ingredients' >>
characteristic( 'CountryOfOrigin' ) equals 'BRAZIL'@'OriginCountries'
```

We have these items with characteristic records:

Item Identifier	Characteristic	Lookup of Characteristic	Characteristic record value
Item1	Ingredient	Ingredients	Sugar
Item1	CountryOfOrigin	OriginCountries	Thailand
Item2	Ingredient	Ingredients	Sugar
Item2	CountryOfOrigin	OriginCountries	India
Item2	Ingredient	Ingredients	Meat
Item2	CountryOfOrigin	OriginCountries	Brazil

Item Identifier	Characteristic	Lookup of Characteristic	Characteristic record value
Item3	Ingredient	Ingredients	Sugar
Item3	CountryOfOrigin	OriginCountries	Brazil

The query above will return **Item3**.

Rest call

```
GET http://localhost:1512/rest/V1.0/list/Article/bySearch?
query=(characteristic( 'Ingredient' ) equals 'SUGAR'@'Ingredients' or
characteristic( 'Ingredient' ) equals 'SALT'@'Ingredients') >>
characteristic( 'CountryOfOrigin' ) equals 'BRAZIL'@'OriginCountries'
```

We have these items with characteristic records:

Item Identifier	Characteristic	Lookup of Characteristic	Characteristic record value
Item1	Ingredient	Ingredients	Sugar
Item1	CountryOfOrigin	OriginCountries	Thailand
Item2	Ingredient	Ingredients	Sugar
Item2	CountryOfOrigin	OriginCountries	India
Item2	Ingredient	Ingredients	Meat
Item2	CountryOfOrigin	OriginCountries	Brazil
Item3	Ingredient	Ingredients	Sugar

Item Identifier	Characteristic	Lookup of Characteristic	Characteristic record value
Item3	CountryOfOrigin	OriginCountries	Brazil
Item4	Ingredient	Ingredients	Salt
Item4	CountryOfOrigin	OriginCountries	Brazil

The query above will return **Item3** and **Item4**.

Rest call

```
GET http://localhost:1512/rest/V1.0/list/Article/bySearch?
query=(characteristic('Ingredient') = 'WOOL' >> characteristic('SerialNumber') = 55)
AND ( characteristic('Ingredient') = 'LEATHER' >> characteristic('CountryOfOrigin') =
'ES')
```

We have these items with characteristic records:

Item Identifier	Characteristic	Lookup of Characteristic	Characteristic record value
Item1	Ingredient	Ingredients	Wool
Item1	CountryOfOrigin	OriginCountries	Spain
Item1	SerialNumber	-	55
Item2	Ingredient	Ingredients	Wool
Item2	CountryOfOrigin	OriginCountries	India
Item2	SerialNumber	-	55

Item Identifier	Characteristic	Lookup of Characteristic	Characteristic record value
Item2	Ingredient	Ingredients	Leather
Item2	CountryOfOrigin	OriginCountries	Spain
Item3	Ingredient	Ingredients	Wool
Item3	CountryOfOrigin	OriginCountries	Brazil
Item4	Ingredient	Ingredients	Leather
Item4	CountryOfOrigin	OriginCountries	Spain

The query above will return **Item2**.

Limitation

Currently it is not possible to search for the datatype MIME in search expressions.

13.6.5 REST List API

The [REST List API](#) (see page 463) provides searching, reporting, navigation and finally result listing of nearly all objects in the system. Based on a query a homogeneous list of objects of a certain type is returned. The queries are parameterized. The resulting list is a two dimensional table with a given set of columns to control the informational character of the result.

- [Overview](#) (see page 464)
- [List-Based Mass Data Access](#) (see page 464)
- [Root Entity vs. Sub Entity Access](#) (see page 464)
- [Report vs. Search](#) (see page 464)
- [Report Services](#) (see page 464)
- [Search Services](#) (see page 465)
- [Examples](#) (see page 465)

13.6.5.1 Overview

The List API provides list-based read, write and delete functionality optimized for fast transmission of large amounts of data. It also contains a meta API in order to access meta information dynamically, e.g. for listing all available root entities in the Product Manager's repository.

13.6.5.2 List-Based Mass Data Access

The REST List API is based on the idea of ListModels, which is one central aspect of the Product Manager's data access layer as described in [Data Models \(see page 133\)](#). List-based data access enables to transfer only the data that is needed for processing the current request, omitting unnecessary chunks. Data can be accessed in fast manner, since virtual list models in turn support paging of the result table.

13.6.5.3 Root Entity vs. Sub Entity Access

As described in [Domain Model \(Repository\) \(see page 79\)](#), the PIM Core data model consists of entities, which in turn consist of sub entities. Since sub entity instances are distinguished by their dimensions, logical keys need to be provided when navigating on sub entity level - e.g. we need to fully qualify the path to a sub entity field (provide all logical keys) when accessing the English item short description.

13.6.5.4 Report vs. Search

A Report is a predefined query with a fixed set of parameters and a fixed semantic. Usually the report has a name which reflects its main purpose, like "byCatalog" or "byStatus". The actual report implementation might be implemented by a Stored Procedure or by some other logic which is totally transparent for the REST API user. Take a look at the [Entity Reporting \(see page 137\)](#) page to learn more about Reports and how to contribute your own reports to the Heiler Product Manager.

A Search has no predefined query and only a limited number of fixed parameters. Its semantic is defined by the search query the user must provide for the execution of the search. Heiler Product Manager provides a generic search module which can be used to build simple and complex queries. Take a look at the [Entity Search \(see page 151\)](#) page to learn more about the possibilities of the generic search engine and also its limitations.

13.6.5.5 Report Services

Such a query can represent a **predefined report**, provided to the Product Manager by an [entity report \(see page 137\)](#) contribution which can simply be executed by providing usually one or two parameters.

Because it is possible to extend the set of entity reports also through customizing, we cannot list the total number of entity reports which are available here. On different installations there might be a different set of entity reports. However, the rest API provides info resources which will list the available reports of a specific Product Manager instance. Using these info resources you are even able to create a generic UI for the end user in which he can pick the report he likes.

Samples:

- Show all items/products of the catalog(catalog, assortment)
- Show items/products classified by(structure system)
- Show items/products not classified by(structure system)
- Show items/products by status reached(status)
- Show items/products by status not reached(status)

13.6.5.6 Search Services

The search services are provided in the same way than the report services. The only specialty is the search query parameter which must follow the [HSQ Syntax](#) (see page 0).

13.6.5.7 Examples

Additionally to the examples provided in this documentation, you can also use the ListAPI Postman collection: ListAPI.postman_collection.json

13.6.5.8 REST List API Info

The REST List API provides the possibility to get meta information about available reports and about the parameters of a specific report.

- [List all entities with reports](#) (see page 465)
 - [Result](#) (see page 466)
- [List all available reports for an entity](#) (see page 466)
 - [Result](#) (see page 467)
- [List report specific parameters](#) (see page 467)
 - [Result](#) (see page 468)
- [Examples](#) (see page 468)
 - [List all entities with reports](#) (see page 468)
 - [Retrieving all reports for the entity Article](#) (see page 469)
 - [Retrieving all report specific parameters for the report byCatalog](#) (see page 470)

List all entities with reports

Returns a list of all entities which have contributed reports

URL Pattern	<code>/list/info</code>
Method	<code>GET</code>
Parameters	No parameters are available

Media types	text/html, application/json, application/xml
Result	In case of media types <i>application/json</i> and <i>application/xml</i> a list of all entities which have contributed reports is returned. In case of media type <i>text/html</i> additionally some general information about the <i>REST List API</i> is returned.

Result

A list of entities is returned.

Properties of an entities element			
	Field	Data type	Description
	identifier	String	Entity identifier
	name	String	Language specific name
	description	String	Language specific description

List all available reports for an entity

Returns the list of all available reports for the specified entity.

URL Pattern	/list/{entity-identifier}/info
Method	GET
Parameters	No parameters are available
Media types	text/html, application/json, application/xml

Result	A list of all available reports for the specified entity is returned.
--------	---

Result

A list of reports is returned. For each report the identifier, the name and the description is provided.

Properties of an reports element			
	Field	Data type	Description
	identifier	String	Report identifier
	name	String	Language specific name of the report
	description	String	Language specific description of the report

List report specific parameters

Returns a list of all report specific parameters.

URL Pattern	<code>/list/{entity-identifier}/{report-name}/info</code>
Method	<code>GET</code>
Parameters	No parameters are available
Media types	<code>text/html, application/json, application/xml</code>
Result	Returns a list of report specific parameters. If <code>text/html</code> is request, also the general, report independent parameters are returned.

Result

Properties of an parameters element			
	Field	Data type	Description
	identifier	String	Parameter identifier
	name	String	Language specific name of the parameter
	description	String	Language specific description of the parameter
	dataType	String	
	entity	Object	Proxy representing a repository entity. The proxy contains the properties identifier, name and description. Only set if data type is ENTITY_ITEM and an enumeration is not specified.
	enumeration	Object	Proxy representing an enumeration. The proxy contains the properties identifier and name.

Examples

List all entities with reports

Rest Call
<pre>curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/list/info</pre>

The following JSON object is returned:

1	{"entities": [
---	----------------

```

2      {
3          "identifier": "Article",
4          "name": "Item",
5          "description": ""
6      },
7      {
8          "identifier": "Structure",
9          "name": "Structure",
10         "description": ""
11     },
12     ...
13 }

```

Rest Client Java Code

```

ListInfoRequest listInfoRequest = restClient.createListInfoRequest();
ReportInfos reportInfoList = listInfoRequest .getAllEntitiesWithReports();

```

Retrieving all reports for the entity Article

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/
V1.0/list/Article/info

```

The following JSON object is returned:

```

1  {"reports": [
2      {
3          "identifier": "byAssortment",
4          "name": "Items in assortment",
5          "description": "Determines all items in an assortment"
6      },
7      {
8          "identifier": "byCatalog",
9          "name": "Items from supplier catalog",
10         "description": "Determines all items from a supplier catalog"
11     },
12     ...
13 ]}

```


Rest Client Java Code

```
ListInfoRequest listInfoRequest = restClient.createListInfoRequest();
ReportInfos reportInfoList = listInfoRequest.getAllReports( "Article" );
```

Retrieving all report specific parameters for the report *byCatalog*

The report identifier has to be added (here "byCatalog" to request the items of a certain catalog):

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/
V1.0/list/Article/byCatalog/info
```

The following JSON object is returned:

```

1  {
2      "parameters": [
3          {
4              "identifier": "catalog",
5              "name": "Catalog",
6              "description": "The catalog for which the items are to be
determined",
7              "mandatory": "false",
8              "dataType": "ENTITY_ITEM",
9              "entity": {},
10             "enumeration": {
11                 "name": "Supplier catalogs and master catalog",
12                 "identifier": "Enum.SupplierCatalogsWithMaster"
13             }
14         }
15     ]
16 }
```

Rest Client Java Code

```
ListInfoRequest listInfoRequest = restClient.createListInfoRequest();
ReportInfos reportInfoList = listInfoRequest.getReportParameters( "Article",
"byCatalog" );
```

13.6.5.9 REST List API Read

The [List API](#) (see page 463) provides table based read access on objects via reports contributed to the system. List API read access has to be distinguished between reading root and sub entities in terms of the URL pattern and the query parameters being passed for the request as well as the resulting table being returned.

- [Caching](#) (see page 470)
 - [Use Caching with Paging](#) (see page 470)
 - [Disable Caching](#) (see page 470)

Caching

Table requests can be cached inside of the application server. This feature has been designed to improve the performance in situations in which you want to fetch not all entity items of a certain report/search at once, but in a paged manner. Without the caching the application server would re-execute the report/search every time you request the next page - which is a tremendous overhead.

Use Caching with Paging

When you intend to use the paging functionality with the `startIndex` and `pageSize` parameters, it's a good idea to also use the table caching functionality. Otherwise, the full table will be requested on server side every time you request the next page. This would lead to a big overhead and will slow down the client application, and, probably also the Product 360 server.

To use the caching you need to provide a cache id. This is an arbitrary alphanumeric string which you can define however you like. It's main purpose is to distinguish the request for a specific table from one client to the other. If you have multiple client connections which are able to share the same table, you can also define the same `cacheId`. It's up to the configuration of the caching in the application server how many tables and for how long they are stored in memory.

From a service API standpoint, it doesn't matter. The caching functionality is transparent to the caller. This means, in case the table has been purged from the cache in between two calls from the client, the table will be reloaded transparently.

Disable Caching

If you specify `cacheId= no-cache`, the caching functionality is disabled for this call, and the table will not be cached. It is strongly recommended to provide `no-cache` every time you do not intend to reuse the table!



Please note that in version 1 of the Service API the default behavior of the `cacheId` parameter is not optimal. In fact, if you omit the `cacheId`, the table **will** be added to the cache and an arbitrary `cacheId` will be returned from the server.

In future versions of the Service API this behavior will be changed so that omitting the `cacheId` will not add the table to the cache.

Therefor we strongly recommend to always use `cacheId=no-cache` unless you really need the table cache. Otherwise a serious memory issue might be imposed on the application server, depending on how the list model cache is actually configured. (By default it's set to 1000 list models, eternal in memory lifetime!)

REST List API Read for Root Entities

In general, most common list-based read access will be utilized on root entity level, e.g. Products or StructureGroups, whereas fields can be requested on demand and need to be fully qualified (see [REST Field Qualification](#) (see page 441)) in case they belong to a sub entity of the requested root entity.

- [Executing a specific report](#) (see page 472)
 - [Parameters](#) (see page 473)
 - [Result](#) (see page 477)
 - [Examples](#) (see page 482)
 - [Executing the report byItems with POST](#) (see page 482)
 - [Executing the report byCatalog for the catalog TOOLS](#) (see page 483)
 - [Executing the report byCatalog for the catalog TOOLS and specifying a list of fields \(parameter formatData is false\)](#) (see page 484)
 - [Executing the report byCatalog for the catalog TOOLS and specifying a list of fields \(parameter formatData is true\)](#) (see page 485)
 - [Searching for root entity objects](#) (see page 487)
- [Execute a specific report for root entities for MIME values](#) (see page 488)
 - [Examples for MIME values](#) (see page 489)
 - [Rest Java Client Example for MIME values](#) (see page 489)

Executing a specific report

URL Pattern	<code>/list/{entity-identifier}/{report-name}</code>
Method	GET or Post
Parameters	Common list parameters are supported. Additional parameters of an entity report are report specific and can be obtained from the report service itself.

Accept Header	<code>application/json, application/xml</code>
Content-Type (for POST only)	<code>application/x-www-form-urlencoded</code> All parameters besides the ones from the URL must be provided as form parameters
Result	A REST List API Read for Root Entities (see page 472) which is optimized for fast transmitting and big item counts

Parameters

Beneath the query dependent parameters there are general parameters which can be passed to control the number of rows or the the specific columns which should be included in the query's result

General list query parameters				
Parameter	Required	Default	Datatype	Parameter description
<code>startIndex</code>	no	0	Integer	0-based index within the total list of results. The result list provided for that request will start from that index (paging the results). An integer value in the range between 0 and <i>totalSize</i> -1.

General list query parameters				
pageSize	no	100	Integer	<p>Number of entries which should be provided with that request (paging the results). The list starts at the given <i>start index</i> (see above) and leads to the end index (<i>startIndex + pageSize</i>). If the results <i>totalSize</i> is less than the requested <i>pageSize</i> or if <i>totalSize - startIndex</i> is less than <i>pageSize</i> the number of entries provided is less than the requested <i>pageSize</i>.</p> <p>You can use <code>-1</code> to obtain all items in one request - please note that this might cause serious performance problems on the client side, since the number of items can be quite large.</p>
fields (see page 441)	no		String	<p>Comma-separated list of fields to be provided as result. Please see the REST Field Qualification (see page 441) as well as REST Transition Fields (see page 446) for details on the syntax of this parameter.</p>
metaData	no	false	Boolean	<p>A Boolean parameter. If set to <i>true</i> the meta data of the requested fields (given in the parameter <i>fields</i> see above) are provided.</p> <p>As part of the meta data the language dependent name of the data fields is provided. The locale used by the HTTP request (see the <i>Accept-Language</i> http request-header field) is used to identify the language requested. If the locale can not be resolved or no language dependent name is available for the given locale the field's unique name is provided at this place.</p>

General list query parameters				
<code>formatData</code>	no	false	Boolean	<p>A Boolean parameter. If set to <i>true</i> the values having a locale/language dependent formatting (like date/time, numeric, etc.) returned are formatted corresponding the given locale used by the HTTP request (see the <i>Accept-Language</i> http request-header field). Also fields with an enumeration will be returned with the label of the enumeration's entry instead of it's key.</p> <p>In case the parameter is omitted or set to false, the data will be formatted in an easy to parse, locale neutral format. See the data type representation (see page 0) section for details.</p>
<code>includeLabels</code> (see page 438)	no	true	Boolean	<p>A boolean parameter. If set to true, columns of the data type ENTITY_ITEM also contain the human readable label of the entity item. For compatibility reasons this parameter is by default <code>true</code> for performance reasons clients should set it to <code>false</code>.</p>

General list query parameters				
orderBy	no	0-ASC	String	<p>The field(s) used to sort the result set can be specified by the this parameter. A 0-based index referring to the list of fields as given in the parameter <i>fields</i> has to be provided plus the ordering direction given by <i>ASC</i> (order ascending) or <i>DESC</i> (order descending). The syntax is as follows:</p> <p><code>n-ASC/DESC</code> where <i>n</i> is the 0-based column index and either <i>ASC</i> or <i>DESC</i> has to be provided ("1-ASC" defines to sort ascending by the second field provided in the parameter <i>fields</i>). Multiple columns for ordering can be provided as a comma separated list ("1-ASC,0-DESC" sort by the second column ascending , then descending by the first column).</p> <p>If not necessary for the client to process the rows in a sorted way, he should not provide this parameter. If not given the order of the rows is not defined - even if they look sorted. This might just be a coincidence because of internal processing algorithms. You can only rely on a sort order if the orderBy parameter is provided.</p>
assortmentFilter	no	none	EntityItem	<p>An entity item representation of an assortment. The assortment's item entity must match with the entity of the items the query returns. Additionally to that, also the item parent must match. For example, it's no allowed to query all items of a Catalog "Henri" and specifying an assortment which is bound to the Mastercatalog.</p> <p>You will receive an appropriate error message in case the assortment does not fit.</p>

General list query parameters				
updateAssortment	no	false	Boolean	This parameter works only in combination with the assortment parameter. If set to true the given assortment will be updated before it is used in the query. Please note that updating an assortment might be a performance intensive task, so take care when using this parameter and think about if you really need to have the assortment evaluated every time.
cacheId	no	-	String	See Caching (see page 470) for details
softDeleteMode	no	ALIVE_ONLY	String	<p>The mode for loading soft deleted objects. Possible values:</p> <p>ALIVE_ONLY - Only include non-deleted objects in the result list. This is the default mode.</p> <p>DEAD_OR_ALIVE - Include both non-deleted and soft deleted objects into the result list. In terms of sub entity records, always the "newest" value is provided, meaning that if a non-deleted object for the current qualification exists, this one will be returned, and if only deleted objects exist, the one with the latest deletion time stamp will be returned, respectively.</p>
revision	no	-	ENTITY_ITEM	The revision entity item for which the data should be retrieved

Result

The result of all List API service is always a list model. A list model is an object type independent data structure representing a list of results of a query. Basically it contains a list of list entries (**rows**). The list header provides general information on the number and type of list entries. Optionally the requested data fields' meta information can be provided (**columns**).

Each list entry consists of identifying fields which allow to identify the object behind the list entry uniquely - the navigable. Additionally each list entry might contain a list of object type specific data fields. These fields can be requested flexibly by passing a list of unique field identifiers. Optionally the requested fields' meta information can be provided.

```
{
  "cacheId": "20130403_164441_0",
  "entityIdentifier": "Article",
  "totalSize": 88640,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 100,
  "columnCount": 2,
  "columns": [
    ...
  ],
  "rows": [
    ...
  ]
}
```

Header

The list header always contains the information about the **total number** of results. The result list is provided paged. The **start index** and the **number of entries** within the provided list model page are given in the list header information. The start index and the number of entries to be provided can be controlled by request parameters of the query (**startIndex** and **pageSize**). The actual number of returned list entries and columns within one request are stated with **rowCount** and **columnCount**. Paging of a result table in the web front-end or a virtual table functionality can be realized by this. The **entity type** of the entries of the result list is given as well generally in the list's header information.

List header	
Field	Description
cacheId	The id of the used cache.
entityIdentifier	The entity of the HPM repository the list model contains entries of.
totalSize	The total number of results of the query.

List header	
startIndex	The 0-based index of first entry of the currently provided list model page within the whole result set.
pageSize	The number of requested entries per list model page.
rowCount	Number of entries contained in the currently provided list model page (can be smaller than requested page size).
columnCount	Number of columns contained in the returned list model page (as requested by the fields query parameter).

Columns

The data fields' meta information for client side presentation and validation can be provided with the search result optionally. The meta data ("columns") is added to the resulting list model in case the query passes the parameter **metaData** with **true**.

The language dependent name as defined in the repository server side is provided. The language requested is identified by the locale provided with the HTTP request-header field *Accept-Language*. In case the requested language is not available a fallback the the repositories default language is implemented.

The columns are provided in the same sequence as requested by the parameter "fields" (see above) and identified by the unique field identifier the column stands for. Additionally the field's data type is provided. In future further information might be added for enhanced client side validations of the fields' content (typical information could be the minimum/maximum string length, minimum/maximum value of a numeric field, etc.).

```
"columns": [
{
  "identifier": "Article.SupplierAID",
  "dataType": "STRING",
  "name": "Item No."
},
...
]
```

Column	
Field	Description
identifier	The unique field identifiers including qualification.
datatype	The data type of the field.
name	The language dependent display name of the field. The language used is controlled by the HTTP request-header field <i>Accept-Language</i> when the parameter <i>formatData</i> is provided with <i>true</i> .

Rows

Each record in the resulting list is represented by a **list row**. Each row consists of a set of identifying fields, the so called navigable, which allow to identify the object behind the entry uniquely.

Additionally each entry contains a list of **values** provided for the data fields requested by the parameter "fields". The sequence of the values corresponds to the sequence the data fields in the parameter "fields" were requested.

The number of rows provided as a result for the certain query is controlled by the request parameter **pageSize**. The index within the whole result set of the first entry in the list provided is controlled by the request parameter **startIndex**.

```
"entries": [
{
  "object":
    {
      "id": "3264@1",
      "label": "IT-12345"
    },
  "values": [
    "IT-12345",
    "Hübscher brauner Schuh",
    "Nice brown shoe",
    "Atlas",
    "1.00"
  ]
},
...
]
```

Entry

Field	Description
object	Entity Item Reference to the corresponding object within the PIM system.

Entry														
Field	Description													
values	<p>A list of values provided per entry for the data fields requested by the parameter "fields" (a comma separated list of unique field identifiers including their qualifications). The sequence of the values corresponds to the sequence of the data fields in the parameter "fields" were requested. If no fields were specified the values element is an empty array.</p> <p>The value is usually represented as a String except for fields with data type <i>ENTITY_ITEM</i>. In this case, the value is a JSON object with the properties <code>id</code> and <code>label</code> if the field has an upper-bound of 0 or 1. If the upper bound is greater than 1, the value is a JSON array of JSON objects with the properties <code>id</code> and <code>label</code>.</p> <table><tr><th>Data type</th><th>Upper bound</th><th>JSON type</th></tr><tr><td>ENTITY_ITEM</td><td>>1</td><td>Array of ProxyObjects (properties: <code>id</code> and <code>label</code>)</td></tr><tr><td>ENTITY_ITEM</td><td><=1</td><td>ProxyObject (properties: <code>id</code> and <code>label</code>)</td></tr><tr><td>not ENTITY_ITEM</td><td></td><td>String (delimiter for multiple values is semi-colon)</td></tr></table>		Data type	Upper bound	JSON type	ENTITY_ITEM	>1	Array of ProxyObjects (properties: <code>id</code> and <code>label</code>)	ENTITY_ITEM	<=1	ProxyObject (properties: <code>id</code> and <code>label</code>)	not ENTITY_ITEM		String (delimiter for multiple values is semi-colon)
Data type	Upper bound	JSON type												
ENTITY_ITEM	>1	Array of ProxyObjects (properties: <code>id</code> and <code>label</code>)												
ENTITY_ITEM	<=1	ProxyObject (properties: <code>id</code> and <code>label</code>)												
not ENTITY_ITEM		String (delimiter for multiple values is semi-colon)												

Examples

Executing the report byItems with POST

The entity report "byItems" uses the ENTITY_ITEM Syntax of the service api. It's single mandatory parameter is "items". In case of this report, items means "entity items", not item as in product/item/variant. So this entity report is actually available for all entities.

```
curl --location --request POST 'http://localhost:1512/rest/V2.0/list/Article/byItems' \
```

```
--header 'Accept: application/json' \
--header 'Authorization: Basic cmVzdDpoZWlsZXI=' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'items=11259@1,1257@1' \
--data-urlencode 'fields=Article.SupplierAID'
```

This example will return two rows, one for the Article entity item with the internal id 11259 of the master catalog (internal id=1), and one for the item with the internalId 1257, also from the master. The byItems entity report can handle thousands of entity items and is backed by a cache to resolve those items. You can also use the alternative entity item syntax with the identifiers. Like: 'MyItem'@'MASTER'.

Hint: Always use the byItems entity report instead of the bySearch in case you would just search for the identifier of the entity item. BySearch is NOT backed by an in memory cache, byItems is!

Executing the report *byCatalog* for the catalog *TOOLS*

The parameters of the report have to be added to the URL as query parameters:

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/list/Article/byCatalog?catalog=TOOLS
```

The following JSON object is returned:

```
{
  "cacheId": "20130403_164441_0",
  "entityIdentifier": "Article",
  "totalSize": 88640,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 100,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "86@1064",
        "label": "A1"
      },
      "values": []
    },
    ...
  ]
}
```

Rest Client Java Code

```
EntityItemReference catalog = EntityItemReferenceFactory.createByIdentifier( "TOOLS"
);

ReportQuery reportQuery = new ReportQuery( "byCatalog" );
reportQuery.addParameterValue( "catalog", catalog );

EntityItemTable resultList = restClient.createListReadRequest().getRootItems(
"Article", reportQuery );
```

Executing the report *byCatalog* for the catalog *TOOLS* and specifying a list of fields (parameter *formatData* is false)

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/
V1.0/list/Article/byCatalog?
catalog=TOOLS&fields=Article.SupplierAID,Article.ManufacturerName,Article.MainSupplie
r,Article.QuantityMin
&formatData=false&metaData=true&startIndex=0&pageSize=50&orderBy=0-ASC
```

The following JSON object is returned:

```
{
  "cacheId": "20130403_164441_0",
  "entityIdentifier": "Article",
  "totalSize": 88640,
  "startIndex": 0,
  "pageSize": 50,
  "rowCount": 50,
  "columnCount": 4,
  "columns": [
    {
      "identifier": "Article.SupplierAID",
      "dataType": "STRING",
      "name": "Item no."
    },
    {
      "identifier": "Article.ManufacturerName",
      "dataType": "STRING",
      "name": "Manufacturer"
    },
    {
      "identifier": "Article.MainSupplier",
      "dataType": "ENTITY_ITEM",
      "name": "Main supplier"
    },
    {
      "identifier": "Article.QuantityMin",
```

```

        "dataType": "DECIMAL",
        "name": "Minimum order quantity"
    }
],
"rows": [
    {
        "object": {
            "id": "1304@1",
            "label": "A1"
        },
        "values": [
            "A1",
            "Test Supplier 1",
            {
                "id": "3",
                "label": "Heiler Product Manager"
            },
            "1"
        ]
    },
    ...
]
}

```

Rest Client Java Code

```

EntityItemReference catalog = EntityItemReferenceFactory.createByIdentifier( "TOOLS"
);

ReportQuery reportQuery = new ReportQuery( "byCatalog" );
reportQuery.addParameterValue( "catalog", catalog );
EntityItemTable resultList = restClient.createListReadRequest()
    .setFields( "Article.SupplierAID",
"Article.ManufacturerName", "Article.MainSupplier", "Article.QuantityMin" )
    .setOrderBy( "0-DESC" )
    .setFormatData( false )
    .setMetaData( true )
    .setPageSize(50)
    .getRootItems( "Article", reportQuery );

```

Executing the report *byCatalog* for the catalog *TOOLS* and specifying a list of fields (parameter *formatData* is true)

Finally the additionally required parameters to control the result have to be added:

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/
V1.0/list/Article/byCatalog?

```



```
catalog=TOOLS&fields=Article.SupplierAID,Article.ManufacturerName,Article.MainSupplier,Article.QuantityMin
&formatData=true&metaData=true&startIndex=0&pageSize=50&orderBy=0-ASC
```

The following JSON object is returned:

```
{
  "cacheId": "20130403_175512_0",
  "entityIdentifier": "Article",
  "totalSize": 88640,
  "startIndex": 0,
  "pageSize": 50,
  "rowCount": 50,
  "columnCount": 4,
  "columns": [
    {
      "identifier": "Article.SupplierAID",
      "dataType": "STRING",
      "name": "Item no."
    },
    {
      "identifier": "Article.ManufacturerName",
      "dataType": "STRING",
      "name": "Manufacturer"
    },
    {
      "identifier": "Article.MainSupplier",
      "dataType": "ENTITY_ITEM",
      "name": "Main supplier"
    },
    {
      "identifier": "Article.QuantityMin",
      "dataType": "DECIMAL",
      "name": "Minimum order quantity"
    }
  ],
  "entries": [
    {
      "object": {
        "id": "1304@1",
        "label": "A2"
      },
      "values": [
        "A2",
        "Test Supplier 1",
        {
          "id": "3",
          "label": "Heiler Product Manager"
        },
        "1.0000"
      ]
    }
  ]
}
```

```
]
}
```

Searching for root entity objects

In order to search for objects matching a specific criteria, a generic *bySearch* report is available for all root entities. The search criteria can be specified by providing a search expression in the *query* parameter. The search expression language to be used is documented in [REST Search Query Language](#) (see page 447). The *catalog* parameter can be used to specify in which catalog to look for (default is Master catalog).

Search for all items of the master catalog which contain "4711" in their GTIN number

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/list/Article/bySearch?catalog=MASTER&query=Article.EAN contains "4711"
```

Rest Client Java Code

```
EntityItemReference testCatalogRef = EntityItemReferenceFactory.createByIdentifier(
    "TOOLS" );

ReportQuery reportQuery = new ReportQuery( "bySearch" );
reportQuery.addParameterValue( "query", "Article.EAN contains \"4711\"");

EntityItemTable resultList = restClient.createListReadRequest()
    .getRootItems( "Article", reportQuery );
```

Search for all items of the master catalog which are ordered in boxes, with German gross weight between 10 and 30 kg

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/list/Article/bySearch?query=Article.OrderUnit equals "Box" AND
ArticleLogistic.GrossWeight(Germany) >= 10.00 AND
ArticleLogistic.GrossWeight(Germany) <= 30.00
```

Search for tasks assigned to specific supplier

To request the list of tasks which are assigned to a supplier use following request:

Tasks bySupplier

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/list/Task/bySupplier?supplier=SUPPLIER_FOR_TASKS
```

NOTE: This report returns single (standard) tasks and workflow tasks. No task groups will be returned in the result. Only workflow tasks which have items (count > 0) will be returned

Parameters				
Parameter	Required	Default	Datatype	Parameter description
supplier	yes		EntityItem	Entity item representation of supplier (supplier proxy) for which the tasks should be provided. NOTE: supplier proxy should be represented using an internal ID or using supplier name (not identifier)
includeCompleted	no	false	Boolean	Indicates if completed tasks should be also provided in the result or not. Default value is false => completed tasks will be not provided in the result as default
catalogs	no		List of EntityItems	List with entity item presentations of catalogs (catalog proxies) which will restrict the list of provided in the result tasks. Optional parameter. If given, then only tasks which have content from requested supplier catalogs or empty tasks will be provided in the result. Useful e.g. for the Broker user.

Execute a specific report for root entities for MIME values

URL Pattern

/list/{entity-identifier}/mimes/{report-name}

Method	GET and POST
Parameters	The parameters <code>pageSize</code> , <code>startIndex</code> , <code>fields</code> (see page 441) and <code>qualificationFilter</code> are supported. Other List Read for sub entities parameters are ignored. Additional parameters of an entity report are report specific and can be obtained from the report service itself.
Media types	<code>application/octet-stream</code>
Result	A zip stream with MIME files including the folder structure from target system.

Examples for MIME values

Execute the report *byLookup* and retrieve the MIME files referenced by lookup "Apps" using GET

Rest Call

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET http://localhost:1512/rest/V1.0/list/LookupValue/byLookup?lookup=Apps&fields=LookupValue.MIMEValue
```

Execute the report *byLookup* and retrieve the MIME files referenced by LookupValue by specifying the field using POST

Rest Call

```
curl -u rest:heiler -H "Accept: application/octet-stream" --data "lookup=Apps&fields=LookupValue.MIMEValue" http://localhost:1512/rest/V1.0/list/LookupValue/byLookup
```

Rest Java Client Example for MIME values

Following code retrieves first 100 MIME values including the empty ones from the lookup "Apps".

Usage example for the client API

```
//Create and login to the client...
RestClient restClient = new RestClient();

//Create and parameterize the report query you want to execute...
ReportQuery reportQuery = new ReportQuery( "byLookup" ).addParameterValue( "lookup",
"Apps" );

//Create and parameterize the list request
ListReadRequest mimeZipRequest = restClient.createListReadRequest();
InputStream mimeZipStream = mimeZipRequest.setPageSize( 100 )
                                           .setStartIndex( 0 )
                                           .setFields( "LookupValue.MIMEValue" )
                                           .getRootMIMES( "LookupValue",
reportQuery );
```

REST List API Read for Sub Entities

Provides the ability to retrieve field content of a sub entity without providing a full qualification. An example is a list of all attribute names and values for a set of objects retrieved by a report.

- [Execute a specific report for Read](#) (see page 490)
 - [Examples](#) (see page 495)
- [Execute a specific report of sub entities for MIME values](#) (see page 504)
 - [Parameters](#) (see page 505)
 - [Examples](#) (see page 505)
 - [Rest Java Client Example](#) (see page 507)

Execute a specific report for Read

URL Pattern	/list/{entity-identifier}/{sub-entity-identifier}/{report-name}
Method	GET or POST

Parameters	Common list parameters are supported. Additional parameters of an entity report are report specific and can be obtained from the report service itself. At least one parameter has to be specified, otherwise the list of parameters for this report is returned.
Accept	<code>application/json</code> , <code>application/xml</code>
Content Type (only for POST)	<code>x-www-form-urlencoded</code>
Result	A list model which is optimized for fast transmitting and big item counts

Parameters

Beneath the query dependent parameters there are general parameters which have to be passed to control how the query's result has to be provided. These parameters are mostly the same as for root entities and therefore linked to the previous page.

General list query parameters					
	Parameter	Required	Default	Datatype	Parameter description
	<code>startIndex</code>	no	0	Integer	see REST List API Read for Root Entities (see page 472)
	<code>pageSize</code>	no	100	Integer	see REST List API Read for Root Entities (see page 472)

General list query parameters

fields (see page 441)	no		String	see REST List API Read for Root Entities (see page 472)
metadata	no	false	Boolean	see REST List API Read for Root Entities (see page 472)
formatData	no	false	Boolean	see REST List API Read for Root Entities (see page 472)
orderBy	no	0-ASC	String	<p>The field(s) used to sort the result set can be specified by the this parameter. A 0-based index referring to the list of fields as given in the parameter <i>fields</i> has to be provided plus the ordering direction given by <i>ASC</i> (order ascending) or <i>DESC</i> (order descending). The syntax is as follows:</p> <p><code>n-ASC/DESC</code> where <i>n</i> is the 0-based column index and either <i>ASC</i> or <i>DESC</i> has to be provided ("1-ASC" defines to sort ascending by the second field provided in the parameter fields). Multiple columns for ordering can be provided as a comma separated list ("1-ASC,0-DESC" sort by the second column ascending, then descending by the first column).</p> <p>In case of a sub-entities, rows which belong to the same root item are always returned in a group of rows. There will be all rows for ItemA, followed by all rows for ItemD, followed by all rows for ItemB. The order of those groups is not defined, but it is guaranteed that rows belonging to the same root item are together. So when you loop over this table, you can be sure that this item is not coming again as soon as the entity item reference of the row differs to the one of the previous row (aka if the group changes).</p> <p>If an orderBy parameter is given, this basic principle for sub-entities does not change! The sorting will only be applied WITHIN the groups of rows for the same entity item!</p> <p>If not necessary for the client to process the rows in those groups in a sorted way, he should not provide this parameter.</p>

General list query parameters

	qualificationFilter	no		String	<p>This parameter allows to restrict the output to certain qualifications. An example would a filter so that only Euro and Dollar prices for the customer Heiler are returned.</p> <p>The filter string is a comma-separated list of qualification settings. A qualification setting is a <i>qualification name</i> followed by a comma-separated list of values which is put into parentheses . The qualification name is defined in the repository and is included in the Meta-API.</p> <p>Example for prices in Euro and US-Dollar and customer Heiler:</p> <pre>qualificationFilter=currency(EUR,USD),customer(Heiler)</pre>
--	---------------------	----	--	--------	--

General list query parameters

field Filter r (since 10.1. 0.02)	no		String	<p>This parameter allows to restrict the output based on certain field filters. An example would be a filter so that only ArticleLang records are returned which have a certain keyword, or no keyword or any keyword.</p> <p>The filter string is a comma-separated list of filter settings. A filter setting is a <i>field identifier</i> followed by a comma-separated list of values which is put into parentheses.</p> <p>All REST Datatypes (see page 436) are supported for the parameters, except MIME_VALUE.</p> <p>Syntax: <code>fieldFilter= [not] <FieldIdentifier1>([Parameter1], ..., [ParameterN]),[not] <FieldIdentifierN>([Parameter1], ..., [ParameterN])</code></p> <p>Multiple values for a single filter are interpreted as OR, so the row is returned if any of the given values equals the field value of the row. In case multiple field filters are defined, all of them must be met for the row to be returned. Multiple field filters are treated as AND. The value of the fieldFilter must match to the field's datatype. Fields with enumerations can also use any of the synonym strings of the enumeration's values.</p> <p>Example for ArticleLang with "MyMarker" as keyword: <code>fieldFilter=ArticleLang.Keyword("MyMarker")</code></p> <p>Example for ArticleLang which has no "MyMarker as keyword <code>fieldFilter=not ArticleLang.Keyword("MyMarker")</code></p> <p>Example for ArticleLang which has no keyword <code>fieldFilter=ArticleLang.Keyword()</code></p> <p>Example for ArticleLang which has any keyword <code>fieldFilter=not ArticleLang.Keyword()</code></p> <p>Please also find more examples on this parameter in the ListAPI postman collection. See the Examples section of REST List API (see page 463).</p>
--	----	--	--------	--

General list query parameters

	revis ion	no	-	ENTI TY_IT EM	The revision entity item for which the data should be retrieved
--	--------------	----	---	---------------------	---

Examples

Executing the report *byCatalog* and retrieve the values for short description and long description in all languages

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/list/Article/ArticleLang/byCatalog?fields=ArticleLang.DescriptionShort,ArticleLang.DescriptionLong
```

The following JSON object is returned:

```
{
  "cacheId": "20130404_115132_0",
  "entityIdentifier": "ArticleLang",
  "totalSize": 83,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 83,
  "columnCount": 2,
  "columns": [
    {
      "identifier": "ArticleLang.DescriptionShort",
      "dataType": "STRING",
      "name": "Short description (Language selectable)"
    },
    {
      "identifier": "ArticleLang.DescriptionLong",
      "dataType": "STRING",
      "name": "Long description (Language selectable)"
    }
  ],
  "rows": [
    {
      "object": {
        "id": "1304@1",
        "label": "A1"
      }
    }
  ]
}
```

```

    "qualification": {
      "language": "German"
    },
    "values": [
      "Kurzbeschreibung A1",
      "Detailbeschreibung A1"
    ]
  },
  {
    "object": {
      "id": "1304@1",
      "label": "A1"
    },
    "qualification": {
      "language": "English"
    },
    "values": [
      "Short description A1",
      "Long description A1"
    ]
  },
  {
    "object": {
      "id": "1315@1",
      "label": "A2"
    },
    "qualification": {
      "language": "German"
    },
    "values": [
      "Kurzbeschreibung A2",
      "Detailbeschreibung A2"
    ]
  },
  {
    "object": {
      "id": "1315@1",
      "label": "A2"
    },
    "qualification": {
      "language": "English"
    },
    "values": [
      "Short description A2",
      "Long description A2"
    ]
  },
  ...
]
}

```

Rest Client Java Code

```
ReportQuery reportQuery = new ReportQuery( "byCatalog" );
EntityItemTable resultList = restClient.createListReadRequest()
    .setFields( "ArticleLang.DescriptionShort", "ArticleLang.DescriptionLong" )
    .getSubItems( "Article", "ArticleLang",
reportQuery );
```

Executing the report *byCatalog* for the *MASTER* catalog and retrieve the values for attribute name, attribute datatype and attribute value

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/list/Article/ArticleAttribute/byCatalog?
fields=ArticleAttributeLang.Name,ArticleAttribute.Datatype,ArticleAttributeValue.Value&metaData=true
```

The following JSON object is returned:

```
{
  "cacheId": "20130404_134526_0",
  "entityIdentifier": "ArticleAttribute",
  "totalSize": 88640,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 700,
  "columnCount": 3,
  "columns": [
    {
      "identifier": "ArticleAttributeLang.Name",
      "dataType": "STRING",
      "name": "Attribute name (Name selectable, Language selectable)"
    },
    {
      "identifier": "ArticleAttribute.Datatype",
      "dataType": "INTEGER",
      "name": "Attribute data type (Name selectable)"
    },
    {
      "identifier": "ArticleAttributeValue.Value",
      "dataType": "STRING",
      "name": "Attribute value (Name selectable, Language selectable, Identifier selectable)"
    }
  ]
}
```

```

    }
  ],
  "rows": [
    {
      "object": {
        "id": "1143@1",
        "label": "A1"
      },
      "qualification": {
        "name": "Farbe",
        "language": "German"
      },
      "values": [
        "Farbe",
        "String",
        "rot"
      ]
    },
    {
      "object": {
        "id": "1143@1",
        "label": "A1"
      },
      "qualification": {
        "name": "Größe",
        "language": "German"
      },
      "values": [
        "Größe",
        "Number",
        "43"
      ]
    },
    ...
  ]
}

```

Rest Client Java Code

```

ReportQuery reportQuery = new ReportQuery( "byCatalog" );

EntityItemTable resultList = restClient.createListReadRequest()
    .setFields( "ArticleAttributeLang.Name",
"ArticleAttribute.Datatype", "ArticleAttributeValue.Value" )
    .setMetaData( true )
    .getSubItems( "Article",
"ArticleAttribute", reportQuery );

```

Executing the report byItems with POST

The entity report "byItems" uses the ENTITY_ITEM Syntax of the service api. It's single mandatory parameter is "items". In case of this report, items means "entity items", not item as in product/item/variant. So this entity report is actually available for all entities. The byItems entity report can handle thousands of entity items and is backed by a cache to resolve those items. You can also use the alternative entity item syntax with the identifiers. Like: 'MyItem'@'MASTER'.

Hint: Always use the byItems entity report instead of the bySearch in case you would just search for the identifier of the entity item. BySearch is NOT backed by an in memory cache, byItems is!

```
curl --location --request POST 'http://localhost:1512/rest/V2.0/list/Article/ArticleLang/byItems' \
--header 'Accept: application/json' \
--header 'Authorization: Basic cmVzdDpoZWlsZXI=' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'items=11259@1,1257@1' \
--data-urlencode 'fields=ArticleLang.DescriptionShort'
```

This will return the short description field in all languages for the items with the internal id's 11259 and 1257 from the master catalog (= 1).

```
{
  "cacheId": "no-cache",
  "entityIdentifier": "ArticleLang",
  "totalSize": 2,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 2,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "1257@1",
        "entityId": 1000
      },
      "qualification": {
        "language": "9"
      },
      "values": [
        "hummm"
      ]
    },
    {
      "object": {
        "id": "11259@1",
        "entityId": 1000
      },
      "qualification": {
```

```

        "language": "9"
      },
      "values": [
        "dummy"
      ]
    }
  ]
}

```

Executing the report *byItems* for an item in SampleCatalog and retrieve the values for all characteristics

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/list/Article/ArticleCharacteristicValue/byItems?
items='AIW_6382437688'@'SampleCatalog'&fields=ArticleCharacteristicValue.Characterist
ic

```

The following JSON object is returned:

```

{
  "cacheId": "20181109_113416_0",
  "entityIdentifier": "ArticleCharacteristicValue",
  "totalSize": 1,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 7,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "214@1200",
        "label": "AIW_6382437682",
        "entityId": 1000
      },
      "qualification": {
        "recordKey": "72f315fa2e73d20c:3964b945:16050a4147b:-7af4",
        "rootCharacteristic": {
          "id": "78",
          "label": "Sustainability Initiative [Sustainability]",
          "entityId": 8000
        },
        "parentRecordKey": "root",

```

```

        "characteristic": {
            "id": "78",
            "label": "Sustainability Initiative [Sustainability]",
            "entityId": 8000
        }
    },
    "values": [
        {
            "id": "78",
            "label": "Sustainability Initiative [Sustainability]",
            "entityId": 8000
        }
    ]
},
{
    "object": {
        "id": "214@1200",
        "label": "AIW_6382437682",
        "entityId": 1000
    },
    "qualification": {
        "recordKey": "72f315fa2e73d20c:3964b945:16050a4147b:-7a03",
        "rootCharacteristic": {
            "id": "78",
            "label": "Sustainability Initiative [Sustainability]",
            "entityId": 8000
        },
        "parentRecordKey": "72f315fa2e73d20c:3964b945:16050a4147b:-7af4",
        "characteristic": {
            "id": "79",
            "label": "Eco Footprint [Sustainability.EcoFootprint]",
            "entityId": 8000
        }
    },
    "values": [
        {
            "id": "79",
            "label": "Eco Footprint [Sustainability.EcoFootprint]",
            "entityId": 8000
        }
    ]
}
...
]
}

```

Executing the report *by/items* for an item in *SampleCatalog* and retrieve the values for Order

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/list/Article/ArticleCharacteristicValue/byItems?
items='AIW_6382437688'@'SampleCatalog'&fields=ArticleCharacteristicValue.Order
```

The following JSON object is returned:

```
{
  "cacheId": "20181109_113931_0",
  "entityIdentifier": "ArticleCharacteristicValue",
  "totalSize": 1,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 7,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "214@1200",
        "label": "AIW_6382437682",
        "entityId": 1000
      },
      "qualification": {
        "recordKey": "72f315fa2e73d20c:3964b945:16050a4147b:-7af4",
        "rootCharacteristic": {
          "id": "78",
          "label": "Sustainability Initiative [Sustainability]",
          "entityId": 8000
        },
        "parentRecordKey": "root",
        "characteristic": {
          "id": "78",
          "label": "Sustainability Initiative [Sustainability]",
          "entityId": 8000
        }
      },
      "values": [
        "-32767"
      ]
    },
    {
      "object": {
        "id": "214@1200",
        "label": "AIW_6382437682",
        "entityId": 1000
      },
    },
  ]
}
```

```

    "qualification": {
      "recordKey": "72f315fa2e73d20c:3964b945:16050a4147b:-7a03",
      "rootCharacteristic": {
        "id": "78",
        "label": "Sustainability Initiative [Sustainability]",
        "entityId": 8000
      },
      "parentRecordKey": "72f315fa2e73d20c:3964b945:16050a4147b:-7af4",
      "characteristic": {
        "id": "79",
        "label": "Eco Footprint [Sustainability.EcoFootprint]",
        "entityId": 8000
      }
    },
    "values": [
      "-32756"
    ]
  }
  ...
]
}

```

Executing the report *byItems* for an item in *SampleCatalog* and retrieve the values for characteristics language specific data

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/list/Article/ArticleCharacteristicValue/byItems?
items='AIW_6382437688'@'SampleCatalog'&fields=ArticleCharacteristicValueLang.Language

```

The following JSON object is returned:

```

{
  "cacheId": "20181030_121849_0",
  "entityIdentifier": "ArticleCharacteristicValue",
  "totalSize": 1,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 14,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "211@1200",
        "label": "AIW_6382437688",
        "entityId": 1000
      }
    }
  ]
}

```

```

    },
    "qualification": {
      "recordKey": "6bfe197e8d5135c4:-1dd34fd6:1604f0fd364:-7ef1",
      "rootCharacteristic": {
        "id": "88",
        "label": "Return Policy [ReturnPolicy]",
        "entityId": 8000
      },
      "parentRecordKey": "root",
      "language": "Language independent",
      "characteristic": {
        "id": "88",
        "label": "Return Policy [ReturnPolicy]",
        "entityId": 8000
      }
    },
    "values": [
      "Language independent"
    ]
  },
  ...
]
}

```

Example with Field Filter

This example requests all ArticleLang sub entity records of all items of the "MySupplierCatalog" catalog and returns the Short Description and the Keyword fields. Only records which have "communicativeness" as a keyword will be returned.

```

curl --location --request GET 'http://localhost:1512/rest/V1.0/list/Article/
ArticleLang/byCatalog?
fields=ArticleLang.DescriptionShort,ArticleLang.Keyword&fieldFilter=ArticleLang.Keywo
rd(%22communicativeness%22)&catalog=%27MySupplierCatalog%27' \
--header 'Accept: application/json' \
--header 'Authorization: Basic cmVzdDpoZWlsZXI='

```

Please find additional examples for the field filters in the ListApi postman collection.

Execute a specific report of sub entities for MIME values

URL Pattern

```

/list/{entity-identifier}/{sub-entity-identifier}/
mimes/{report-name}

```

Method	GET and POST
Parameters	The parameters <code>pageSize</code> , <code>startIndex</code> , <code>fields</code> (see page 441) and <code>qualificationFilter</code> are supported. Other List Read for sub entities parameters are ignored. Additional parameters of an entity report are report specific and can be obtained from the report service itself.
Media types	<code>application/octet-stream</code>
Result	A zip stream with MIME files including the folder structure from target system.

Parameters

The parameters used here are same as that of root entities.

Parameters valid for Article → ArticleCharacteristicValue entities only				
includeUnavailable	no	false	Boolean	Allows to return MIME files for available AND unavailable characteristics. Default value is false and means that only available characteristics will be considered. Read more about Characteristic values in KnowledgeBase → Characteristics - Dynamic data model → Characteristic values

Note: Parameter `fields` (see page 441) is not used for Article → ArticleCharacteristicValue.

Examples

Execute the report *byCatalog* and retrieve the MIME files referenced by Article characteristic values for items of MASTER catalog using GET

Rest Call

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET http://
localhost:1512/rest/V2.0/list/Article/ArticleCharacteristicValue/mimes/byCatalog
```

Execute the report *bySearch* and retrieve the MIME files referenced by Article characteristic values for specific item using GET

Rest Call

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET
http://localhost:1512/rest/V2.0/list/Article/ArticleCharacteristicValue/mimes/
bySearch?query=Article.SupplierAID = "ITEM_WITH_CHARACTERISTIC"
```

Execute the report *byStructureGroup* and retrieve the MIME files referenced by Article characteristic values qualified by characteristic record key using GET

Rest Call

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET
http://localhost:1512/rest/V2.0/list/Article/ArticleCharacteristicValue/mimes/
byStructureGroup?
structureGroup='GROUP_1_2'@'HeilerStandard'&qualificationFilter=recordKey("c003c0da0c
a388f7:-2c5e83f8:1614649c74c:-7e9d")
```

Execute the report *byLookup* and retrieve the MIME files referenced by LookupValueLang by specifying the field using POST

Rest Call

```
curl -u rest:heiler -H "Accept: application/octet-stream" --data
"lookup=Apps&fields=LookupValueLang.MIMEValue" http://localhost:1512/rest/V1.0/list/
LookupValue/LookupValueLang/byLookup
```

Rest Java Client Example

You can find a short example on how to use the client below. The full example classes can be viewed as part of the SDK package in the folder `examples\integration`.

Example 1

Following code block is used for retrieving first 100 MIME values from *masterCatalog* for the Characteristics with identifiers "CharacteristicIdentifier1" and "CharacteristicIdentifier2".

Usage example for the client API

```
//Create and login to the client...
RestClient restClient = new RestClient();

//Create and parameterize the report query you want to execute...
EntityItemReference masterCatalog = EntityItemReferenceFactory.createByIdentifier(
"MASTER" );
ReportQuery reportQuery = new ReportQuery( "byCatalog" ).addParameterValue( "catalog",
masterCatalog );

//Create and parameterize the list request
ListReadRequest mimeZipRequest = restClient.createListReadRequest();
InputStream mimeZipStream = mimeZipRequest.setPageSize( 100 )
                                           .setStartIndex( 0 )
                                           .setQualificationFilters( "characteristic('
CharacteristicIdentifier1')",
"characteristic('CharacteristicIdentifier2')" )
                                           .getSubMIMES( "Article",
"ArticleCharacteristicValue", reportQuery );
```

Example 2

Following code block retrieves the first 100 MIME values from the subentity LookupValueLang.MIME.

Usage example for the client API

```
//Create and login to the client...
RestClient restClient = new RestClient();
```

```
//Create and parameterize the report query you want to execute...
ReportQuery reportQuery = new ReportQuery( "byLookup" ).addParameterValue( "lookup",
"Apps" );

//Create and parameterize the list request
ListReadRequest mimeZipRequest = restClient.createListReadRequest();
InputStream mimeZipStream = mimeZipRequest.setPageSize( 100 )
                                           .setStartIndex( 0 )
                                           .setFields( "LookupValueLang.MIMEValue" )
                                           .getSubMIMES( "LookupValue",
"LookupValueLang", reportQuery ))
```

REST List API Read MIME files of Root and Sub Entities

The [REST List API Read MIME files of Sub Entities](#) (see page 508) provides the ability to determine MIME files referenced by a sub entity and provide them as a zip stream. An example is a zip stream containing MIME files for a set of objects retrieved by a report.

- [Root Entity](#) (see page 508)
 - [Execute a specific report for root entities](#) (see page 508)
 - [Parameters](#) (see page 509)
 - [Examples](#) (see page 510)
 - [Rest Java Client Example](#) (see page 511)
- [Sub Entity](#) (see page 511)
 - [Execute a specific report for sub root entities](#) (see page 511)
 - [Parameters](#) (see page 512)
 - [Additional parameters](#) (see page 512)
 - [Examples](#) (see page 512)
 - [Rest Java Client Example](#) (see page 514)

Root Entity

Execute a specific report for root entities

URL Pattern	<code>/list/{entity-identifier}/mimes/{report-name}</code>
Method	GET and POST

Parameters	The parameters <code>pageSize</code> , <code>startIndex</code> , <code>fields</code> (see page 441) and <code>qualificationFilter</code> are supported. Other List Read for sub entities parameters are ignored. Additional parameters of an entity report are report specific and can be obtained from the report service itself.
Media types	<code>application/octet-stream</code>
Result	A zip stream with MIME files including the folder structure from target system.

Parameters

Beneath the query dependent parameters below are the general parameters which can be passed to control the query's result:

General list query parameters				
Parameter	Required	Default	Datatype	Parameter description
<code>startIndex</code>	no	0	Integer	see REST List API Read for Root Entities (see page 472)
<code>pageSize</code>	no	100	Integer	see REST List API Read for Root Entities (see page 472)

General list query parameters				
qualificationFilter	no		String	<p>This parameter allows to restrict the output to certain qualifications. An example would be a filter so that only Euro and Dollar prices for the customer Heiler are returned.</p> <p>The filter string is a comma-separated list of qualification settings. A qualification setting is a <i>qualification name</i> followed by a comma-separated list of values which is put into parentheses. The qualification name is defined in the repository and is included in the Meta-API.</p> <p>Example for prices in Euro and US-Dollar and customer Heiler:</p> <pre>qualificationFilter=currency(EUR,USD),customer(Heiler)</pre>
fields (see page 441)	no		String	<p>Comma-separated list of fields to be provided as result. Please see the REST Field Qualification (see page 441) as well as REST Transition Fields (see page 446) for details on the syntax of this parameter. It is usually used for specifying which field contains the mime value.</p>

Examples

Execute the report *byLookup* and retrieve the MIME files referenced by lookup "Apps" using GET

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET http://localhost:1512/rest/V1.0/list/LookupValue/byLookup?lookup=Apps&fields=LookupValue.MIMEValue
```

Execute the report *byLookup* and retrieve the MIME files referenced by LookupValue by specifying the field using POST

```
curl -u rest:heiler -H "Accept: application/octet-stream" --data
"lookup=Apps&fields=LookupValue.MIMEValue" http://localhost:1512/rest/V1.0/list/
LookupValue/byLookup
```

Rest Java Client Example

Following code retrieves first 100 MIME values including the empty ones from the lookup "Apps".

Usage example for the client API

```
//Create and login to the client...
RestClient restClient = new RestClient();

//Create and parameterize the report query you want to execute...
ReportQuery reportQuery = new ReportQuery( "byLookup" ).addParameterValue( "lookup",
"Apps" );

//Create and parameterize the list request
ListReadRequest mimeZipRequest = restClient.createListReadRequest();
InputStream mimeZipStream = mimeZipRequest.setPageSize( 100 )
                                           .setStartIndex( 0 )
                                           .setFields( "LookupValue.MIMEValue" )
                                           .getRootMIMES( "LookupValue",
reportQuery );
```

Sub Entity

Execute a specific report for sub root entities

URL Pattern	/list/{entity-identifier}/{sub-entity-identifier}/ mimes/{report-name}
Method	GET and POST

Parameters	The parameters <code>pageSize</code> , <code>startIndex</code> , <code>fields</code> (see page 441) and <code>qualificationFilter</code> are supported. Other List Read for sub entities parameters are ignored. Additional parameters of an entity report are report specific and can be obtained from the report service itself.
Media types	<code>application/octet-stream</code>
Result	A zip stream with MIME files including the folder structure from target system.

Parameters

The parameters used here are same as that of root entities described above.

Additional parameters

There is an additional optional parameter to use in the calls for Characteristic MIME values.

Additional parameters				
<code>includeUnavailable</code>	<code>no</code>	<code>false</code>	Boolean	Allows to return MIME files for available AND unavailable characteristics. Default value is false and means that only available characteristics will be considered.

 Parameter `fields` (see page 441) is ignored in the calls for Characteristic MIME values.

Read more about Characteristic values in KnowledgeBase → Characteristics - Dynamic data model → Characteristic values.

Examples

Execute the report *byCatalog* and retrieve the MIME files referenced by Article characteristic values for items of MASTER catalog using GET

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET http://localhost:1512/rest/V2.0/list/Article/ArticleCharacteristicValue/mimes/byCatalog
```

Execute the report *bySearch* and retrieve the MIME files referenced by Article characteristic values for specific item using GET

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET http://localhost:1512/rest/V2.0/list/Article/ArticleCharacteristicValue/mimes/bySearch?query=Article.SupplierAID = "ITEM_WITH_CHARACTERISTIC"
```

Execute the report *bySearch* and retrieve the MIME files referenced by Variant characteristic values for specific variant using GET

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET http://localhost:1512/rest/V2.0/list/Variant/VariantCharacteristicValue/mimes/bySearch?query=Variant.VariantNo equals "VARIANT_WITH_CHARACTERISTIC"
```

Execute the report *byStructureGroup* and retrieve the MIME files referenced by Article characteristic values qualified by characteristic record key using GET

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET http://localhost:1512/rest/V2.0/list/Article/ArticleCharacteristicValue/mimes/byStructureGroup?structureGroup='GROUP_1_2'@'HeilerStandard'&qualificationFilter=recordKey("c003c0da0ca388f7:-2c5e83f8:1614649c74c:-7e9d")
```

Execute the report *byStructureGroup* and retrieve the MIME files referenced by Product characteristic values qualified by root characteristic using GET

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET http://localhost:1512/rest/V2.0/list/Product2G/Product2GCharacteristicValue/mimes/byStructureGroup?structureGroup='GROUP_2_1'@'MIME_Structure'&qualificationFilter=rootCharacteristic('Root1')
```

Execute the report *byLookup* and retrieve the MIME files referenced by LookupValueLang by specifying the field using POST

```
curl -u rest:heiler -H "Accept: application/octet-stream" --data "lookup=Apps&fields=LookupValueLang.MIMEValue" http://localhost:1512/rest/V1.0/list/LookupValue/LookupValueLang/byLookup
```

Rest Java Client Example

You can find a short example on how to use the client below. The full example classes can be viewed as part of the SDK package in the folder `examples\integration`.

Example 1

Following code block is used for retrieving first 100 MIME values from *masterCatalog* for the Characteristics with identifiers "CharacteristicIdentifier1" and "CharacteristicIdentifier2".

Usage example for the client API

```
//Create and login to the client...
RestClient restClient = new RestClient();

//Create and parameterize the report query you want to execute...
EntityItemReference masterCatalog = EntityItemReferenceFactory.createByIdentifier(
"MASTER" );
ReportQuery reportQuery = new ReportQuery( "byCatalog" ).addParameterValue( "catalog",
masterCatalog );

//Create and parameterize the list request
ListReadRequest mimeZipRequest = restClient.createListReadRequest();
InputStream mimeZipStream = mimeZipRequest.setPageSize( 100 )
                                           .setStartIndex( 0 )
                                           .setQualificationFilters( "characteristic('
CharacteristicIdentifier1')",
"characteristic('CharacteristicIdentifier2')" )
                                           .getSubMIMES( "Article",
"ArticleCharacteristicValue", reportQuery );
```

Example 2

Following code block retrieves the first 100 MIME values from the subentity `LookupValueLang.MIME`.

Usage example for the client API

```
//Create and login to the client...
RestClient restClient = new RestClient();

//Create and parameterize the report query you want to execute...
ReportQuery reportQuery = new ReportQuery( "byLookup" ).addParameterValue( "lookup",
"Apps" );

//Create and parameterize the list request
ListReadRequest mimeZipRequest = restClient.createListReadRequest();
InputStream mimeZipStream = mimeZipRequest.setPageSize( 100 )
                                           .setStartIndex( 0 )
                                           .setFields( "LookupValueLang.MIMEValue" )
                                           .getSubMIMES( "LookupValue",
"LookupValueLang", reportQuery ))
```

13.6.5.10 REST List API Write

- [REST List API Write for Root Entities \(see page 515\)](#)
- [REST List API Write for Sub Entities \(see page 528\)](#)

REST List API Write for Root Entities

The REST List API provides the possibility to set the values of several qualified fields for a list of given objects. The request returns a protocol containing the updated resp. created objects and a list of errors and warnings. It is possible to create new objects by providing external identifiers instead of internal IDs. Processing is distributed over multiple threads to speed up processing.

- [Write values into qualified fields \(see page 515\)](#)
 - [Parameters \(see page 516\)](#)
 - [Content \(see page 517\)](#)
 - [Result \(see page 517\)](#)
- [Examples \(see page 520\)](#)
 - [Setting non-formatted values \(see page 520\)](#)
 - [Setting formatted values \(see page 523\)](#)
 - [Setting MIMEValues for entity \(see page 524\)](#)

Write values into qualified fields

Writes the given values into qualified fields for one or more specified objects. All objects have to be of the same type.

URL Pattern	/list/{entity-identifier}
Method	POST
Parameters	formatData
Content types	application/json
Media type	application/json
Result	A protocol containing some statistical information like the number of errors and warnings, a list of all errors and warnings and the created resp. updated objects.

Parameters

Available parameters					
	Parameter	Required	Default	Datatype	Parameter description
	formatData	no	false	Boolean	If set to true, language specific formatted values are expected.
	includeObjectsInProtocol	no	true	Boolean	If set to true, a list of created and updated objects are included in the protocol. To improve performance, this value should be set to false if the list of objects is not needed. This option is available since HPM 7.0.04.

Available parameters

	recreate	no	false	Boolean	If set to true, a sub entity item is always newly created (currently, only QualityStatusEntry objects are supported)
--	----------	----	-------	---------	--

Content

The expected content has the same format as the result of a read request (see also [REST List API Read for Root Entities](#) (see page 477)) but not all properties have to be filled. Not required properties are ignored. It is possible to use the output of a read request, modify some values and use the modified read output as content for a write request.

Required properties are:

- `columns` containing
 - an array of objects. Each object must contain the property `identifier` which contains the qualified field identifier
- `rows` containing
 - the object id. If an external identifier is provided and a corresponding object does not exist, a new object is created.
 - a list of new values for each object.
- `mimeValueArchives` containing
 - File Reference id and originalFileName which is generated while uploading File using File upload API.
 - Multiple FileReference id and originalFileName can be passed.
 - This property is introduced to MIMEValue field only.

Formatting of values

- The values have to be provided as strings and formatted as described in [REST Datatypes](#) (see page 436).
- If an empty string is specified as value, the content of the field is cleared. If `null` is specified as value, the content of the field is left unchanged.
- In case a field has the type `ENTITY_ITEM` either a string or an JSON object can be provided. In case a string is provided, the string is treated as label if the field has an enumeration, otherwise as external identifier. In case a JSON object is provided, only the property `id` is evaluated.

Result

A protocol is returned as result. The protocol consists of a list of counters (number of errors, warnings etc.), a list of errors and warnings (members of the field `entries`) and a list of created and updated objects

(members of the field `objects`). The list of created and updated objects is only included if the option `includeObjectsInProtocol` is set to true (default).

Properties of a <code>counters</code> element			
	Field	Data type	Description
	<code>errors</code>	Integer	Total number of errors
	<code>warnings</code>	Integer	Total number of warnings
	<code>createdObjects</code>	Integer	Number of created objects
	<code>updatedObjects</code>	Integer	Number of updated objects
	<code>objectsWithErrors</code>	Integer	Number of objects with errors
	<code>objectsWithWarnings</code>	Integer	Number of objects with warnings
Properties of an <code>entries</code> element			
	Field	Data type	Description
	<code>row</code>	Integer	The index of the row causing this error or warning. The first row has the index 0.
	<code>objectType</code>	String	The type of the object (language dependent label is used). Might be empty.
	<code>object</code>	EntityItem	The object the error or warning occurred in. Might be empty.

	severity	String	ERROR, WARNING or INFO
	category	String	Category of the error. Possible values are: SYSTEM, UNIQUENESS, DATATYPE, CARDINALITY, RANGE, CONSISTENCY, SECURITY, NOTE, SUMMARY
	propertyLabel	String	The field causing this error or warning (language dependent label is used). Might be empty.
	message	String	The message describing the error or warning.
	logDate	Date	Date when the error or warning occurred.
	logTime	Time	Time when the error or warning occurred.

Properties of an objects element

	Field	Data type	Description
	row	Integer	The index of the row which is responsible for creating or updating this object. The first row has the index 0.
	object	EntityItem	A reference to the updated or created object. Contains the id and the label of the object.
	status	List of Strings	The status of the object. Possible values are: CREATED, UPDATED, ERROR, WARNING

Examples

Setting non-formatted values

Sets values in the fields *Article.ManufacturerName*, *Article.OrderUnit* and *ArticlePriceValueSales.Amount*

```
curl -u rest:heiler -H "Accept: application/json" -X POST http://localhost:1501/rest/V1.0/list/Article
```

The following JSON object is provided as content:

```

1  {
2    "columns": [
3      {
4        "identifier": "Article.ManufacturerName"
5      },
6      {
7        "identifier": "Article.OrderUnit"
8      },
9      {
10       "identifier":
11       "ArticlePriceValueSales.Amount(1,nrp,USD,US,2010-08-06,1.5)"
12     }
13   ],
14   "rows": [
15     {
16       "object":
17       {
18         "id": "'A1'@'MASTER'"
19       },
20       "values": [
21         "Heiler Software AG",
22         {
23           "id": "39"
24         },
25         "12.34"
26       ]
27     },
28     {
29       "object":
30       {
31         "id": "'A2'@'MASTER'"
32       },
33       "values": [
34         "Heiler Software AG",
35         {
36           "label": "fifteen kg drum"
37         },
38         "NotANumber"
39       ]
40     }
41   ]
42 }
```

```

38     ]
39   },
40   {
41     "object":
42     {
43       "id": "'A3'@'MASTER'"
44     },
45     "values": [
46       "",
47       { },
48       ""
49     ]
50   },
51   {
52     "object":
53     {
54       "id": "'A4'@'MASTER'"
55     },
56     "values": [
57       null,
58       null,
59       null
60     ]
61   }
62 ]
63 }
64
65

```

Remarks:

- If the items A1, A2, A3 and A4 exists already, they are updated. Otherwise the items are newly created.
- In case of item A2, an error occurs while setting the value. An error description is provided in the protocol.
- In case of item A3, the content of the fields is cleared .
- In case of item A4, the content of the fields is left unchanged .

The following protocol is returned :

```

1  {
2    "counters": {
3      "errors": 1,
4      "warnings": 0,
5      "createdObjects": 4,
6      "updatedObjects": 0,
7      "objectsWithErrors": 1,
8      "objectsWithWarnings": 0
9    },
10   "entries": [
11     {
12       "row": 1,
13       "objectType": "Item",
14       "object": {

```

```

15         "id": "120015@1",
16         "label": "A2"
17     },
18     "severity": "ERROR",
19     "category": "DATATYPE",
20     "propertyLabel": "Price (from 1.0000)",
21     "message": "Value 'NotANumber' could not be converted into
type java.math.BigDecimal.",
22     "logDate": "2013-04-04",
23     "logTime": "11:59:46"
24 }
25 ],
26 "objects": [
27     {
28         "row": 0,
29         "object": {
30             "id": "120014@1",
31             "label": "A1"
32         },
33         "status": [
34             "CREATED"
35         ]
36     },
37     {
38         "row": 1,
39         "object": {
40             "id": "120015@1",
41             "label": "A2"
42         },
43         "status": [
44             "CREATED",
45             "ERROR"
46         ]
47     },
48     {
49         "row": 2,
50         "object": {
51             "id": "120013@1",
52             "label": "A3"
53         },
54         "status": [
55             "CREATED"
56         ]
57     },
58     {
59         "row": 3,
60         "object": {
61             "id": "120012@1",
62             "label": "A4"
63         },
64         "status": [
65             "CREATED"
66         ]

```

```

67     }
68   ]
69 }

```

Setting formatted values

Sets the values which are provided in a German format

```

curl -u rest:heiler -H "Accept: application/json" -H "Accept-Language: de-DE" -X POST
http://localhost:1501/rest/V1.0/list/Article?formatData=true

```

The following JSON object is provided as content:

```

1  {
2    "columns": [
3      {
4        "identifier": "Article.ManufacturerName"
5      },
6      {
7        "identifier": "Article.OrderUnit"
8      },
9      {
10       "identifier":
11       "ArticlePriceValueSales.Amount(1,nrp,USD,US,2010-08-06,1.5)"
12     },
13     "rows": [
14       {
15         "object":
16         {
17           "id": "'A1'@'MASTER'"
18         },
19         "values": [
20           "Heiler Software AG",
21           {
22             "label": "15 kg-Fass"
23           },
24           "12,34"
25         ]
26       }
27     ]
28   }

```

The returned protocol looks like:

```

1    "counters": {
2      "errors": 0,
3      "warnings": 0,

```

```

4      "createdObjects": 0,
5      "updatedObjects": 1,
6      "objectsWithErrors": 0,
7      "objectsWithWarnings": 0
8    },
9    "entries": [],
10   "objects": [
11     {
12       "row": 0,
13       "object": {
14         "id": "120033@1",
15         "label": "A1"
16       },
17       "status": [
18         "UPDATED"
19       ]
20     }
21   ]
22 }

```

Setting MIMEValues for entity

To set MIMEValue to particular column, first we have to upload Zip file containing MIMEValue we want to attach using File API. While uploading File using File upload API, user will get id and originalFileName object. After that object has to set to 'mimeValueArchives' as shown in example. User can't reuse Filereference of uploaded file.

```

curl -u rest:heiler -H "Accept: application/json" -H 'Content-Type: application/json'
-X POST http://localhost:1512/rest/V1.0/List/LookupValue

```

The following JSON object is provided as content:

```

1  {
2    "columns": [
3      {
4        "identifier": "LookupValue.MIMEValue"
5      }
6    ],
7    "rows": [
8      {
9        "object": {
10         "id": "31@27",
11         "label": "Amazon",
12         "entityId": 7300
13       },
14       "values":

```

```

15         [
16             {
17                 "relativeFilePath": "17/8/32/fishstick.jpg"
18             }
19         ]
20     },
21     {
22         "object": {
23             "id": "29@27",
24             "label": "Netflix",
25             "entityId": 7300
26         },
27         "values":
28         [
29             {
30                 "label": "img1.jpg",
31                 "mimeType": "image/jpeg",
32                 "relativeFilePath": "18/1/img1.jpg"
33             }
34         ]
35     },
36     {
37         "object": {
38             "id": "32@27",
39             "label": "Skype",
40             "entityId": 7300
41         },
42         "values":
43         [
44             {
45                 "mimeType": "image/jpeg",
46                 "relativeFilePath": "18/1/img1.jpg"
47             }
48         ]
49     }
50 ],
51 "mimeValueArchives": [
52 {
53     "id": "b1f9e95f-f61e-4139-a429-efa16fbf4035",
54     "originalFilename": "uploadMime.zip"
55 }
56 ]
57
58 }

```

The returned protocol looks like:

```

1  {
2      "counters": {
3          "errors": 0,

```



```

4      "warnings": 0,
5      "createdObjects": 0,
6      "updatedObjects": 3,
7      "objectsWithErrors": 0,
8      "objectsWithWarnings": 0
9  },
10  "entries": [],
11  "objects": [
12      {
13          "row": 0,
14          "object": {
15              "id": "31@27",
16              "label": "Amazon",
17              "entityId": 7300
18          },
19          "status": [
20              "UPDATED"
21          ]
22      },
23      {
24          "row": 1,
25          "object": {
26              "id": "29@27",
27              "label": "Netflix",
28              "entityId": 7300
29          },
30          "status": [
31              "UPDATED"
32          ]
33      },
34      {
35          "row": 2,
36          "object": {
37              "id": "32@27",
38              "label": "Skype",
39              "entityId": 7300
40          },
41          "status": [
42              "UPDATED"
43          ]
44      }
45  ]
46  }

```

Remarks:

- If the object Netflix exists already, they are updated. Otherwise the items are newly created with MIMEValues if MIMEValue reference is set in 'mimeValueArchives'.

Example 2: If mimeValueArchives fileReferences is invalid or used earlier then User will get "400: Bad Request" in Response. As shown in bellow:

```

1  {
2    "columns": [
3      {
4        "identifier": "LookupValue.MIMEValue"
5      }
6    ],
7    "rows": [
8      {
9        "object": {
10         "id": "31@27",
11         "label": "Amazon",
12         "entityId": 7300
13       },
14       "values":
15         [
16           {
17             "relativeFilePath": "17/8/32/fishstick.jpg"
18           }
19         ]
20     },
21     {
22       "object": {
23         "id": "29@27",
24         "label": "Netflix",
25         "entityId": 7300
26       },
27       "values":
28         [
29           {
30             "label": "img1.jpg",
31             "mimeType": "image/jpeg",
32             "relativeFilePath": "18/1/img1.jpg"
33           }
34         ]
35     },
36     {
37       "object": {
38         "id": "32@27",
39         "label": "Skype",
40         "entityId": 7300
41       },
42       "values":
43         [
44           {
45             "mimeType": "image/jpeg",
46             "relativeFilePath": "18/1/img1.jpg"
47           }
48         ]
49     }
50 ],

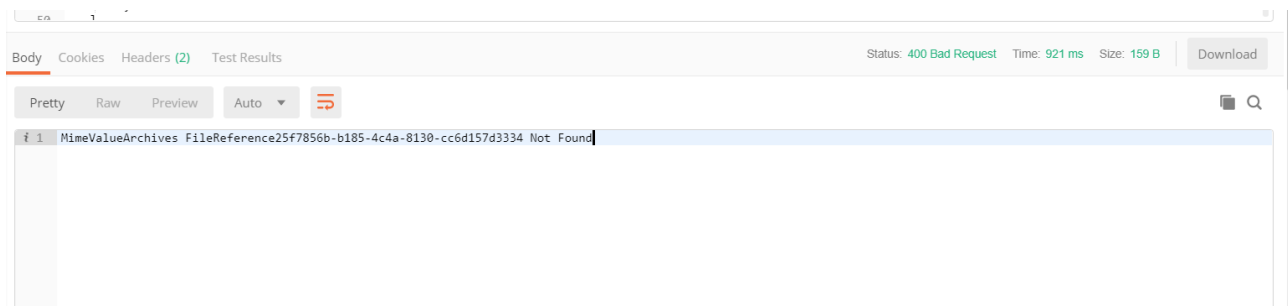
```

```

51     "mimeValueArchives": [
52     {
53         "id": "25f7856b-b185-4c4a-8130-cc6d157d3334",
54         "originalFilename": "uploadMime.zip"
55     }
56     ]
57
58 }

```

The response will look like:



REST List API Write for Sub Entities

The REST List API provides the possibility to set the values of several fields of a specific sub entity for a list of given objects. The request returns a protocol containing the updated resp. created objects and a list of errors and warnings. It is possible to create new objects by providing external identifiers instead of internal IDs. Processing is distributed over multiple threads to speed up processing.

- [Write values into fields of a certain sub entity](#) (see page 528)
 - [Parameters](#) (see page 529)
 - [Content](#) (see page 529)
 - [Result](#) (see page 530)
- [Examples](#) (see page 530)
 - [Setting values in sub entity ArticleLang](#) (see page 530)
 - [Setting values to subentity from root entity](#) (see page 532)
 - [Setting lookup value of an item](#) (see page 534)
 - [Setting MIMEValues for SubEntity](#) (see page 535)

Write values into fields of a certain sub entity

Writes the given values into fields of a one specified sub entity for a list of provided objects. All objects have to be of the same type.

URL Pattern

/list/{entity-identifier}/{sub-entity-identifier}

Method	POST
Parameters	formatData
Content types	application/json
Media type	application/json
Result	A protocol containing some statistical information like the number of errors and warnings, a list of all errors and warnings and the created resp. updated objects.

Parameters

Available parameters					
	Parameter	Required	Default	Datatype	Parameter description
	formatData	no	false	Boolean	If set to true, language specific formatted values are expected.

Content

The expected content has the same format as the result of a read request (see also [REST List API Read for Root Entities \(see page 477\)](#)) but not all properties have to be filled. Not required properties are ignored. It is possible to use the output of a read request, modify some values and use the modified read output as content for a write request.

The expected content has the same format as the result of a report query, but not all properties have to be filled.

Required properties are

- `columns` containing
 - an array of objects. Each object must contain the property `identifier` which contains the qualified field identifier
- `rows` containing

- the object `id`. If an external identifier is provided and a corresponding object does not exist, a new object is created.
- the qualification of the sub entity
- a list of new values for each object

Formatting of values

- The values have to be provided as strings and formatted as described in [REST Datatypes](#) (see page 436).
- If an empty string is specified as value, the content of the field is cleared. If `null` is specified as value, the content of the field is left unchanged.
- In case a field has the type `ENTITY_ITEM` either a string or an JSON object can be provided. In case a string is provided, the string is treated as label if the field has an enumeration, otherwise as external identifier. In case a JSON object is provided, only the property `id` is evaluated.

Result

A protocol is returned as result. The protocol consists of a list of counters (number of errors, warnings etc.), a list of single errors and warnings (in property `entries`) and a list of created and updated objects.

A detailed description can be found at [REST List API Write for Root Entities](#) (see page 517)

Examples

Setting values in sub entity `ArticleLang`

This example set values of the fields `ArticleLang.DescriptionShort` and `ArticleLang.Keyword`

```
curl -u rest:heiler -H "Accept: application/json" -X POST http://localhost:1501/rest/V1.0/list/Article/ArticleLang
```

The following JSON object is provided as content:

```

1  {
2    "columns": [
3      {
4        "identifier": "ArticleLang.DescriptionShort"
5      },
6      {
7        "identifier": "ArticleLang.Keyword"
8      }
9    ],
10   "rows": [
11     {
12       "object":
13       {
14         "id": "'A1'@'MASTER'"

```

```

15     },
16     "qualification" : { "language" : "en" },
17     "values": [
18         "An English short description",
19         [ "EnglishKeyword1", "EnglishKeyword2" ]
20     ]
21 },
22 {
23     "object":
24     {
25         "id": "'A1'@'MASTER'"
26     },
27     "qualification" : { "language" : "de" },
28     "values": [
29         "A German short description",
30         [ "GermanKeyword2", "GermanKeyword1" ]
31     ]
32 },
33 {
34     "object":
35     {
36         "id": "'A2'@'MASTER'"
37     },
38     "qualification" : { "language" : "en" },
39     "values": [
40         "English short description for second item",
41         null
42     ]
43 },
44 {
45     "object":
46     {
47         "id": "'A3'@'MASTER'"
48     },
49     "qualification" : { "language" : "en" },
50     "values": [
51         "English short description for third item",
52         []
53     ]
54 }
55 ]
56 }

```

Remarks:

- In case of item A2, the content of the field *ArticleLang.Keyword* is cleared .
- In case of item A3, the content of the fields *ArticleLang.Keyword* is left unchanged .

The following protocol is returned :

```

1  {
2      "counters": {

```

```

3      "errors": 0,
4      "warnings": 0,
5      "createdObjects": 3,
6      "updatedObjects": 0,
7      "objectsWithErrors": 0,
8      "objectsWithWarnings": 0
9  },
10  "entries": [],
11  "objects": [
12      {
13          "row": 0,
14          "object": {
15              "id": "120044@1",
16              "label": "A1"
17          },
18          "status": [
19              "CREATED"
20          ]
21      },
22      {
23          "row": 2,
24          "object": {
25              "id": "120045@1",
26              "label": "A2"
27          },
28          "status": [
29              "CREATED"
30          ]
31      },
32      {
33          "row": 3,
34          "object": {
35              "id": "120043@1",
36              "label": "A3"
37          },
38          "status": [
39              "CREATED"
40          ]
41      }
42  ]
43  }

```

Setting values to subentity from root entity

This example maps a unit to a specific unit system. In detail, a unit with ID 7001 will be taken and mapped to a unit system with ID 1. Therefore the values of the fields UnitSystemSpecific.Code and UnitSystemSpecific.Lang.Name in english and german will be set for the specified unit system.

```
curl -u rest:heiler -H "Accept: application/json" -X POST http://localhost:1512/rest/
V1.0/list/Unit
```

The following JSON object is provided as content:

```

1  {
2    "columns": [
3      {
4        "identifier": "UnitSystemSpecific.Code(1)"
5      },
6      {
7        "identifier": "UnitSystemSpecificLang.Name(1,9)"
8      },
9      {
10       "identifier": "UnitSystemSpecificLang.Name(1,7)"
11     }
12   ],
13   "rows": [
14     {
15       "object": {
16         "id": "7001"
17       },
18       "values": [
19         "ZzZ",
20         "My unit name for this unit system",
21         "Mein Name für die Einheit in diesem Einheitensystem"
22       ]
23     }
24   ]
25 }
26

```

The following protocol is returned :

```

1  {
2    "counters": {
3      "errors": 0,
4      "warnings": 0,
5      "createdObjects": 0,
6      "updatedObjects": 1,
7      "objectsWithErrors": 0,
8      "objectsWithWarnings": 0
9    },
10   "entries": [],
11   "objects": [
12     {
13       "row": 0,
14       "object": {
15         "id": "7001",
16         "label": "Dose",
17         "entityId": 3100
18       },
19       "status": [
20         "UPDATED"

```



```

21         ]
22     }
23 ]
24 }
```

Setting lookup value of an item

This example set values of the fields *ArticleCharacteristicValue.LookupValue*

```
curl -u rest:heiler -H "Accept: application/json" -X POST http://localhost:1512/rest/
V1.0/list/Article/ArticleCharacteristicValue
```

The following JSON object is provided as content:

```

1  {
2    "columns": [
3      {
4        "identifier": "ArticleCharacteristicValue.LookupValue"
5      }
6    ],
7    "rows": [
8      {
9        "object": {
10           "id": "3@1100"
11         },
12        "qualification": {
13          "recordKey": "72f315fa2e73d20c:3964b945:16050a4147b:-79d4",
14          "rootCharacteristic": {
15            "id": "49"
16          },
17          "parentRecordKey": "root",
18          "characteristic": {
19            "id": "49",
20            "label": "Ingredient [Ingredient]",
21            "entityId": 8000
22          }
23        },
24        "values": [
25          [
26            {
27              "id": "163@111",
28              "label": "Egg",
29              "entityId": 7300
30            }
31          ]
32        ]
33      }
34    ]
35  }
```

```
35 }
}
```

The following protocol is returned :

```
1 {
2   "counters": {
3     "errors": 0,
4     "warnings": 0,
5     "createdObjects": 0,
6     "updatedObjects": 1,
7     "objectsWithErrors": 0,
8     "objectsWithWarnings": 0
9   },
10  "entries": [],
11  "objects": [
12    {
13      "row": 0,
14      "object": {
15        "id": "3@1100",
16        "label": "AIW_6382437684",
17        "entityId": 1000
18      },
19      "status": [
20        "UPDATED"
21      ]
22    }
23  ]
24 }
```

Setting MIMEValues for SubEntity

To set MIMEValue to particular column, first we have to upload Zip file containing MIMEValue we want to attach using File API. While uploading File using File upload API, user will get id and originalFileName object. After that object has to set to 'mimeValueArchives' as shown in example. User can't reuse Filereference of uploaded file.

```
curl -u rest:heiler -H "Accept: application/json" -H 'Content-Type: application/json'
-X POST http://localhost:1512/rest/V1.0/List/LookupValue/LookupValueLang
```

The following JSON object is provided as content:

```
1 {
2   "columns": [
3     {
4       "identifier": "LookupValueLang.MIMEValue"
```

```

5      }
6    ],
7    "rows": [
8      {
9        "object": {
10         "id": "31@27"
11       },
12       "qualification": {
13         "language": "German"
14       },
15       "values":
16       [
17         {
18           "relativeFilePath": "17/8/32/fishstick.jpg"
19         }
20       ]
21     },
22     {
23       "object": {
24         "id": "29@27",
25         "label": "Netflix",
26         "entityId": 7300
27       },
28       "qualification": {
29         "language": "German"
30       },
31       "values":
32       [
33         {
34           "label": "img1.jpg",
35           "mimeType": "image/jpeg",
36           "relativeFilePath": "18/1/img1.jpg"
37         }
38       ]
39     },
40     {
41       "object": {
42         "id": "32@27",
43         "label": "Skype",
44         "entityId": 7300
45       },
46       "qualification": {
47         "language": "German"
48       },
49       "values":
50       [
51         {
52           "mimeType": "image/jpeg",
53           "relativeFilePath": "18/1/img1.jpg"
54         }
55       ]
56     }
57   ]

```

```

58     ],
59     "mimeValueArchives": [
60     {
61         "id": "1f66a149-937f-4401-a3e5-737a05c35a25",
62         "originalFilename": "uploadMime.zip"
63     }
64     ]
65 }

```

The returned protocol looks like:

```

1  {
2      "counters": {
3          "errors": 0,
4          "warnings": 0,
5          "createdObjects": 0,
6          "updatedObjects": 3,
7          "objectsWithErrors": 0,
8          "objectsWithWarnings": 0
9      },
10     "entries": [],
11     "objects": [
12         {
13             "row": 0,
14             "object": {
15                 "id": "31@27",
16                 "label": "Amazon",
17                 "entityId": 7300
18             },
19             "status": [
20                 "UPDATED"
21             ]
22         },
23         {
24             "row": 1,
25             "object": {
26                 "id": "29@27",
27                 "label": "Netflix",
28                 "entityId": 7300
29             },
30             "status": [
31                 "UPDATED"
32             ]
33         },
34         {
35             "row": 2,
36             "object": {
37                 "id": "32@27",
38                 "label": "Skype",
39                 "entityId": 7300

```

```

40         },
41         "status": [
42             "UPDATED"
43         ]
44     }
45 ]
46 }

```

Remarks:

- If the object Netflix exists already, they are updated.

Example 2: If mimeValueArchives fileReferences is invalid or used earlier then User will get "400: Bad Request" in Response. As shown in bellow:

```

1  {
2      "columns": [
3          {
4              "identifier": "LookupValueLang.MIMEValue"
5          }
6      ],
7      "rows": [
8          {
9              "object": {
10                 "id": "31@27"
11             },
12             "qualification": {
13                 "language": "German"
14             },
15             "values":
16                 [
17                     {
18                         "relativeFilePath": "17/8/32/fishstick.jpg"
19                     }
20                 ]
21             },
22             {
23                 "object": {
24                     "id": "29@27",
25                     "label": "Netflix",
26                     "entityId": 7300
27                 },
28                 "qualification": {
29                     "language": "German"
30                 },
31                 "values":
32                     [
33                         {
34                             "label": "img1.jpg",
35

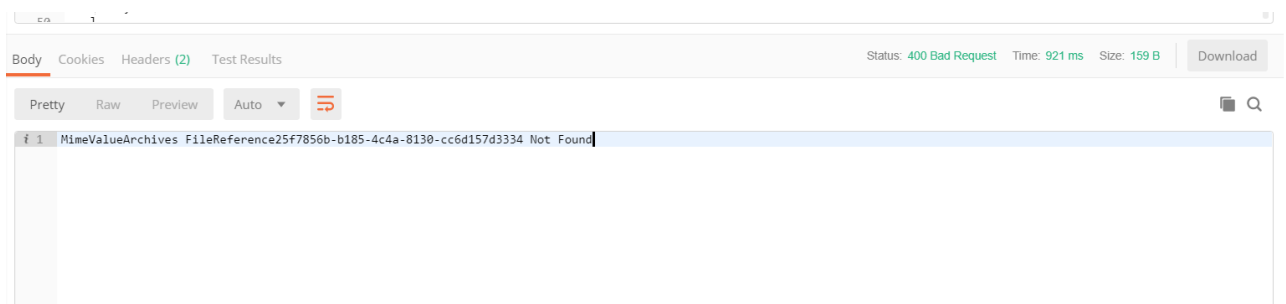
```

```

36         "mimeType": "image/jpeg",
37         "relativeFilePath": "18/1/img1.jpg"
38     }
39 ]
40 },
41 {
42     "object": {
43         "id": "32@27",
44         "label": "Skype",
45         "entityId": 7300
46     },
47     "qualification": {
48         "language": "German"
49     },
50     "values":
51     [
52         {
53             "mimeType": "image/jpeg",
54             "relativeFilePath": "18/1/img1.jpg"
55         }
56     ]
57 },
58 ],
59 "mimeValueArchives": [
60 {
61     "id": "25f7856b-b185-4c4a-8130-cc6d157d3334",
62     "originalFilename": "uploadMime.zip"
63 }
64 ]
65 }

```

The response will look like:



13.6.5.11 REST List API Delete

The REST List API provides the possibility to delete objects using a report. All objects returned by the report are deleted.

- [Delete all objects returned by a report \(see page 540\)](#)

- [Result](#) (see page 540)
- [Delete sub entity records of objects returned by a report](#) (see page 541)
 - [Result](#) (see page 542)
- [Examples](#) (see page 543)
 - [Delete all item in the supplier catalog TOOLS](#) (see page 543)
 - [Delete all sales prices with currency USD in the supplier catalog TOOLS](#) (see page 544)
 - [Delete a Characteristic QUALITY in a revision DAILY_VERSION](#) (see page 545)
 - [Delete all Characteristic Values of an Item in a revision DAILY_VERSION](#) (see page 545)

Delete all objects returned by a report

URL Pattern	<code>/list/{entity-identifier}/{report-name}</code>
Method	DELETE
Parameters	Only the parameters of the specified entity report are available.
Content-Type	<code>application/x-www-form-urlencoded</code>
Media types	<code>application/json</code> , <code>application/xml</code>
Result	A protocol containing information about the number of retrieved and deleted items and a list of errors and warnings.

Result

A protocol is returned as result. The protocol consists of a list of counters (number of errors, warnings and retrieved and deleted objects), a list of errors and warnings (members of the field `entries`) and a list of created and updated objects (members of the field `objects`).

Properties of a <code>counters</code> element			
	Field	Data type	Description
	errors	Integer	Total number of errors

Properties of a counters element

	warnings	Integer	Total number of warnings
	retrievedObjects	Integer	Number of objects retrieved by the report
	deletedObjects	Integer	Number of deleted objects

The properties of an entries element are the same as when writing values and is described at [REST List API Write for Root Entities](#) (see page 517).

Warning

Be aware that when deleting objects via the List API, all objects belonging to the specified report will be deleted at once. So once the deletion request is fired, there will NOT appear any "Are you really sure..?" confirmation dialog 😊!

Limit delete rights of Rest users as much as possible

As a best practise, it is recommended to restrict the delete rights of users accessing the Product Manager via Rest Service API as much as possible.

Delete sub entity records of objects returned by a report

For example "delete all prices for a specific customer from all items of the master catalog"

Since (PIM 8.0)

URL Pattern	<code>/list/{entity-identifier}/{sub-entity-identifier}/{report-name}</code>
Method	<code>DELETE</code>
Content-Type	<code>application/x-www-form-urlencoded</code>

Media types	application/json, application/xml			
Result	A protocol containing information about the number of retrieved and deleted items and a list of errors and warnings.			
Parameters				
Parameter	Required	Default	Datatype	Parameter description
report parameters	no		String	Parameters of the specified entity report.
qualificationFilter	no		String	<p>This parameter allows to restrict the deletion to certain qualifications. An example would a filter so that only Euro and Dollar prices for the customer Heiler are deleted.</p> <p>The filter string is a comma-separated list of qualification settings. A qualification setting is a <i>qualification name</i> followed by a comma-separated list of values which is put into parentheses . The qualification name is defined in the repository and is included in the Meta-API.</p> <p>Example for prices in Euro and US-Dollar and customer Heiler:</p> <pre>qualificationFilter=current cy(EUR,USD),customer(Heile r)</pre>

Result

A protocol is returned as result. The protocol consists of a list of counters (number of errors, warnings and retrieved and deleted objects), a list of errors and warnings (members of the field `entries`) and a list of created and updated objects (members of the field `objects`).

Properties of a counters element

Field	Data type	Description
errors	Integer	Total number of errors
warnings	Integer	Total number of warnings
retrievedObjects	Integer	Number of objects retrieved by the report
deletedObjects	Integer	Number of deleted objects

The properties of an entries element are the same as when writing values and is described at [REST List API Write for Root Entities](#) (see page 517).

Warning

Be aware that when deleting objects via the List API, all objects belonging to the specified report will be deleted at once. So once the deletion request is fired, there will NOT appear any "Are you really sure..?" confirmation dialog!

Examples

Delete all item in the supplier catalog TOOLS

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -H "Content-Type: application/x-www-form-urlencoded" -X DELETE http://localhost:1501/rest/V1.0/list/Article/byCatalog?catalog=TOOLS
```

The returned protocol looks like

```
{
  "counters": {
    "retrievedObjects": 3,
    "warnings": 0,
    "errors": 0,
    "deletedObjects": 3
  }
}
```

```

    },
    "entries": []
  }

```

Rest Client Java Code

```

EntityItemReference toolsCatalogRef = EntityItemReferenceFactory.createByIdentifier(
"TOOLS" );

ReportQuery reportQuery = new ReportQuery( "byCatalog" );
reportQuery.addParameterValue( "catalog", toolsCatalogRef );

ListDeleteRequest deleteRequest = getRestClient().createListDeleteRequest();

DeleteProtocol protocol = deleteRequest.deleteRootItems( "Article", reportQuery );

```

Delete all sales prices with currency USD in the supplier catalog TOOLS

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -H "Content-Type: application/x-
www-form-urlencoded" -X DELETE http://localhost:1501/rest/V1.0/list/Article/
ArticlePriceSales/byCatalog?catalog=TOOLS&qualificationFilter=currency(USD)

```

Rest Client Java Code

```

EntityItemReference toolsCatalogRef = EntityItemReferenceFactory.createByIdentifier(
"TOOLS" );

ReportQuery reportQuery = new ReportQuery( "byCatalog" );
reportQuery.addParameterValue( "catalog", toolsCatalogRef );

ListDeleteRequest deleteRequest = getRestClient().createListDeleteRequest();

DeleteProtocol protocol = deleteRequest.deleteSubItems( "Article",
"ArticlePriceSales", reportQuery, "currency(USD)" );

```

Delete a Characteristic QUALITY in a revision DAILY_VERSION

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -H "Content-Type: application/x-www-form-urlencoded" -X DELETE http://localhost:1512/rest/V1.0/list/Characteristic/bySearch?query=Characteristic.Identifier equals "QUALITY"&revision='DAILY_VERSION'
```

Rest Client Java Code

```
ReportQuery reportQuery = new ReportQuery( "bySearch" );
reportQuery.addParameterValue( "query", "Characteristic.Identifier equals 'QUALITY'" );
;
reportQuery.addParameterValue( "revision", "'DAILY_VERSION'");

ListDeleteRequest deleteRequest = getRestClient().createListDeleteRequest();

DeleteProtocol protocol = deleteRequest.deleteRootItems( "Characteristic",
reportQuery );
```

Delete all Characteristic Values of an Item in a revision DAILY_VERSION

Deletion of Subentity records would be a change of an object in that revision and is not supported.

Warning

Any revision other than the default revision will result in an error message stating that the given revision is not supported!

13.6.5.12 REST List API Errors

General List API Errors

For any list query, the following errors may occur depending on the executed query.

Occasion	Return code	Message
Misspelled / unqualified field	400 (Bad request)	The field list "<Field List>" could not be interpreted: Error 1 (e.g. ArticleLang.DescriptionShort / ArticleLang.DescriptionShort() -> unqualified) Error 2 (e.g. ArticleLang.DescriptionShort(de, en, 123) -> Invalid qualification) Error 3 (e.g. ...)
Invalid entity	404 (Not found)	Entity '<entity identifier>' does not exist
Entity is no root entity	404 (Not found)	Entity '<entity identifier>' is not a root entity.
Report/Search not found	404 (Not found)	Report '<report name>' does not exist
Any other unexpected error	500 (internal server error)	<ul style="list-style-type: none"> The response body will contain the error message of the original exception as plain text without stack trace in this case The stack trace of the exception is logged on the server

Report Specific Errors

If the result of a predefined report is requested, the following errors may occur additionally to the General Errors of the List API.

Occasion	Return code	Message
Missing report parameters	400 (Bad request)	The following mandatory report parameters have not been provided: A, B, C

Occasion	Return code	Message
Invalid values for report parameters	400 (Bad request)	If paramter value could not be converted: "Value of given parameter X has a wrong format" If proxy does not exist: "Given item for parameter X does not exist"
Missing report	404 (Not found)	Report '<ReportName>' does not exist.

Search Specific Errors

If a search query is performed using the search query language, the following errors may happen in addition to the [General Errors](#) (see page 0) of the List API.

Occasion	Return code	Message
Invalid query syntax	400 (Bad request)	Search Query syntax is incorrect Example: Unexpected character found at position 11 a = 1 AND b / 2 ^
No search available	404 (Resource not found)	

13.6.5.13 REST List API Tutorial

Loading Data For Examples

1. Create catalog *RestTutorial*
2. Import the attached zip file RestTutorial.zip (Perspective *Import*, menu *File / Load import project / from zip file*) into the newly created folder *RestTutorial*.

Tools

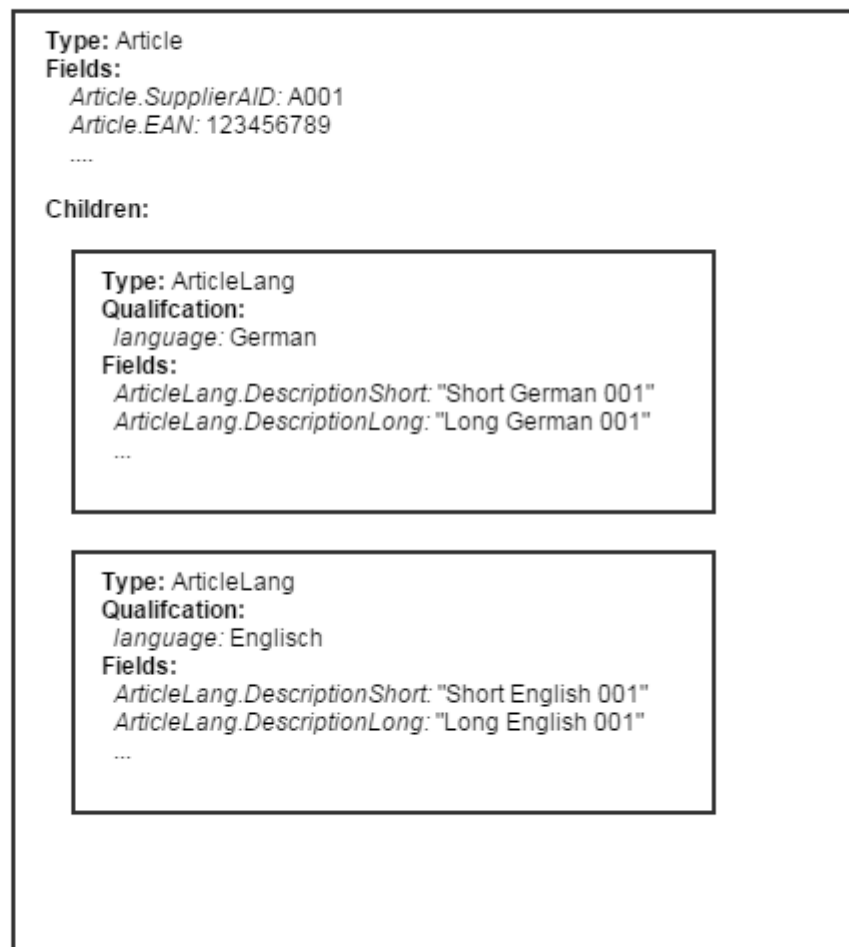
Postman for Google Chrome as REST client (<http://www.getpostman.com/>)

REST API Tutorial - Data Model

Data Model

Objects that can contain embedded/other objects. Embedded objects are uniquely identified by qualifiers (also called logical keys).

Links to other objects are possible.



Explore the data model via HTML Browser

<http://localhost:1501/rest/V1.0/meta/>

Qualifier and fields have types

See [REST Datatypes](#) (see page 436)

Note that references to other objects use a special syntax.

Qualifier and fields can contain enumerations and proposal enumerations

Example for qualifier with enumeration

`http://localhost:1501/rest/V1.0/meta/Article/ArticleLang/ArticleLangType.LK.Language`

Example for values of an enumeration (link is included in qualifier)

`http://localhost:1501/rest/V1.0/enum/Enum.Language`

REST API Tutorial - List Read for Root Entities



The list read for root entities is similar to a root entity table in the Rich Client.

The following information has to be specified for a list read:

1. Which objects should be included (rows) -> Report

List of available reports:

`http://localhost:1501/rest/V1.0/list/info`

Please note that there is a general report called *bySearch* which allows powerful search queries (see also [REST Search Query Language](#) (see page 447)).

General information about reports: [Entity Reporting](#) (see page 137)

2. Which fields are required (columns)

The fields are specified in the URL parameter *fields*

Fields of sub entities have to be qualified, see also [REST Field Qualification](#). (see page 441)

3. Options

Various options are available, like sort order, paging and formatting information. They are described in section *Parameters* in [REST List API Read for Root Entities](#) (see page 472)

Please note that the options *includeLabels* and *cached* have an impact on performance (unfortunately, the default settings are not the fastest settings). If not otherwise required, set *includeLabels* to *false* and *cached* to *no-cache*.

Note that the locale, the request is executed in, has to be set directly in the http header (property *Accept-Language*, the format is *<language>-<country>*, e.g., *en-US* or *de-DE*).

Examples

Root entity fields only

Report:

byCatalog whereby parameter *catalog* is set to *RestTutorial*

Fields:

Article.SupplierAID

Article.ManufacturerName
Article.MainSupplier
Article.QuantityMin

Options:

metaData is set to *true* (shows meta information for the specified fields)
pageSize is set to -1 (return all items)
chached is set to *no-cache* (result of report execution is not cached, paged access is not possible)

GET <http://localhost:1501/rest/V1.0/list/Article/byCatalog?catalog=RestTutorial&fields=Article.SupplierAID,Article.ManufacturerName,Article.MainSupplier,Article.QuantityMin&metaData=true&pageSize=-1&cacheId=no-cache>

Root fields and qualified sub entity fields

Fields:

Article.SupplierAID
ArticleLang.DescriptionShort(German)
ArticleLang.DescriptionShort(English)
ArticleAttributeValue.Value(Length,"Language independent",DEFAULT)
ArticleAttributeValue.Value("Front Color",English,DEFAULT)

GET [http://localhost:1501/rest/V1.0/list/Article/byCatalog?catalog=RestTutorial&fields=Article.SupplierAID,&ArticleLang.DescriptionShort\(German\),ArticleLang.DescriptionShort\(English\),ArticleAttributeValue.Value\(Length,"Language independent",DEFAULT\),ArticleAttributeValue.Value\("Front Color",English,DEFAULT\)&metaData=true](http://localhost:1501/rest/V1.0/list/Article/byCatalog?catalog=RestTutorial&fields=Article.SupplierAID,&ArticleLang.DescriptionShort(German),ArticleLang.DescriptionShort(English),ArticleAttributeValue.Value(Length,\)

Remark:

If a qualification is an enumeration, the label, one of the synonyms or the key can be used.

So instead of ArticleLang.DescriptionShort(German) also the following expressions are possible:

ArticleLang.DescriptionShort(de) (synonym)
ArticleLang.DescriptionShort(7) (key)

Transition field access

Since 7.1.03 it is possible to use transition fields that allow to access values of linked objects directly (currently read access only).

For example to get the acronym of the main supplier from the example above, *Article.MainSupplier->Party.Acronym* can be used.

See also [REST Transition Fields](#) (see page 446).

Fields:

Article.SupplierAID
Article.MainSupplier->Party.Name
Article.MainSupplier->Party.Acronym

```
GET http://localhost:1501/rest/V1.0/list/Article/byCatalog?
catalog=RestTutorial&fields=Article.SupplierAID,Article.MainSupplier-
>Party.Name,Article.MainSupplier->Party.Acronym&metaData=true&pageSize=-1
```

Paged access

Use the parameters *startIndex*, *pageSize* and *cacheId* to perform a page based access.

First call (without parameter *cacheId*):

```
GET http://localhost:1501/rest/V1.0/list/Article/byCatalog?catalog= RestTutorial &
fields=Article.SupplierAID,ArticleLang.DescriptionShort(German)&startIndex=0&pag
eSize=1
```

Second call (with parameter *cacheId*):

```
GET http://localhost:1501/rest/V1.0/list/Article/byCatalog?catalog= RestTutorial &
fields=Article.SupplierAID,ArticleLang.DescriptionShort(German)&startIndex=1&pag
eSize=1&cacheId=<Returned cacheId>
```

REST API Tutorial - List Write for Root Entities



The input of list write and the output of list read uses the same structure. You can perform the list read, modify some values and write them back.

Update

The HTTP method POST instead of GET is used, the body contains the data to be written. The data in the body has the same structure as a read result, but only the values of *columns*, *formatData*, *includeObjectsInProtocol* and *rows* are relevant.

The fields have to be specified in the *columns* section.

Example

The short description and attribute value is updated, new values are *Update short English A001* resp. *light blue* for A001 and *red* as attribute value for A002. The value of the short description of A002 remains unchanged by setting the value to *null*.

The example uses the external identifier instead of the internal id as the internal ID is different on each system and cannot be used for copy and paste example.

External identifier are written in single quotes: 'A001'@'RestTutorial' instead of somethink like 39861@11639 (see also entry *ENTITY_ITEM* in [REST Datatypes](#) (see page 436)).

```
POST http://localhost:1501/rest/V1.0/list/Article
```

Message body

```

1  {
2    "columns": [
3      {
4        "identifier": "ArticleLang.DescriptionShort(English)"
5      },
6      {
7        "identifier": "ArticleAttributeValue.Value(\"Front
Color\",English,DEFAULT)"
8      }
9    ],
10   "rows": [
11     {
12       "object": {
13         "id": "'A001'@'RestTutorial'"
14       },
15       "values": [
16         "Updated Short English A001",
17         "light blue"
18       ]
19     },
20     {
21       "object": {
22         "id": "'A002'@'RestTutorial'"
23       },
24       "values": [
25         null,
26         "red"
27       ]
28     }
29   ]
30 }

```

A protocol is returned containing created resp. updated items and errors resp. warnings.

If the object part of the protocol is not required, set *includeObjectsInProtocol* to *false*. This improves performance significantly.

Example with error

Errors are reported in the returned protocol. This example sets the attribute *Length* to *unknown* for item *A001* which is not a valid decimal value.

Message body

```

1  {
2    "columns": [
3      {
4        "identifier": "ArticleAttributeValue.Value(Length,\"Language
independent\",DEFAULT)"
5      }
6    ],
7    "rows": [
8      {
9        "object": {
10         "id": "'A001'@'RestTutorial'"
11       },
12       "values": [
13         "unknown"
14       ]
15     },
16     {
17       "object": {
18         "id": "'A002'@'RestTutorial'"
19       },
20       "values": [
21         "23"
22       ]
23     }
24   ]
25 }

```

Create

The request is the same POST request as for update, but the *object* entry has to use the external identifier instead of the internal id.

So if you want to create an item with the external identifier (Supplier AID) *A001* in the master catalog, you have to write it as 'A001'@1 or 'A001'@'MASTER'.

See also entry *ENTITY_ITEM* in [REST Datatypes](#) (see page 436).

Example

A new item is created with four fields.

POST <http://localhost:1501/rest/V1.0/list/Article>

Message Body

```

1  {

```

```

2      "columns": [
3        {
4          "identifier": "ArticleLang.DescriptionShort(English)"
5        },
6        {
7          "identifier": "ArticleAttributeValue.Value(\"Front
color\",English,DEFAULT)"
8        }
9      ],
10     "rows": [
11       {
12         "object": {
13           "id": "'A003'@'RestTutorial'"
14         },
15         "values": [
16           "Created Short English A003",
17           "green"
18         ]
19       }
20     ]
21   }

```

REST API Tutorial - List Read for Sub Entities

Motivation

Reads on sub entity level are necessary when

Qualifications are unknown

- Attribute names
- References to other items (the referenced item is part of the qualification)

Too many fields

- Getting values for 100 attributes would require 100 qualified fields

Query

Reading on sub entity level is like reading on entity level, but

- URL contains the identifier of the sub entity additionally to the root entity
- Fields are not qualified
- Results contains the qualification for each row

Remark

Only direct sub entities can be specified, but the queries can contains fields from sub sub entities.

Examples

Language data: Query the values of the field *ArticleLang.DescriptionShort* from the sub entity *ArticleLang* for all languages

```
GET http://localhost:1501/rest/V1.0/list/Article/ArticleLang/byCatalog?
catalog=RestTutorial&fields= ArticleLang.DescriptionShort &metaData=true&pageSize=-1
```

Attributes: Query all values of the fields *ArticleAttribute.Datatype* and *ArticleAttributeValue.Value* (which is a field of a sub sub entity) from the sub entity *ArticleAttribute*

```
GET http://localhost:1501/rest/V1.0/list/Article/ArticleAttribute/byCatalog?
catalog=RestTutorial&fields= ArticleAttribute.Datatype ,ArticleAttributeValue.Value&m
etaData=true&pageSize=-1
```

References: Query all values of the fields *ArticleReference.Type*, *ArticleReference.ReferencedArticle* and *ArticleReference.Quantity* from the sub entity *ArticleReference*

```
GET http://localhost:1501/rest/V1.0/list/Article/ArticleReference/byCatalog?
catalog=RestTutorial&fields= ArticleReference.Type ,ArticleReference.ReferencedArticle,
ArticleReference.Quantity&metaData=true&pageSize=-1
```

Qualification filter

A qualification filter allows to filter the returned values. The filtername can be looked up in qualifier description in the entry *Qualification filter identifier*

Example

In this example only the English short descriptions returned. Therefore the parameter *qualificationFilter* is set to *language(English)*.

The filter name can be found in the entry *Qualification filter identifier* of <http://localhost:1501/rest/V1.0/meta/Article/ArticleLang/ArticleLangType.LK.Language>

```
GET http://localhost:1501/rest/V1.0/list/Article/ArticleLang/byCatalog?
catalog=RestTutorial&fields= ArticleLang.DescriptionShort & qualificationFilter=language(E
nglish)& metaData=true&pageSize=-1
```

REST API Tutorial - List Write for Sub Entities

Writing on sub entity level is similar to writing on root entity level.

Differences:

- URL contains the identifier of the sub entity additionally to the root entity
- Fields are not qualified
- Qualification entry is required for each row

Example

The attributes *Length* and *Front Color* are added to item *A003*

POST <http://localhost:1501/rest/V1.0/list/Article/ArticleAttribute>

Message body

```

1  {
2    "columns": [
3      {
4        "identifier": "ArticleAttribute.Datatype"
5      },
6      {
7        "identifier": "ArticleAttributeValue.Value"
8      }
9    ],
10   "rows": [
11     {
12       "object": {
13         "id": "'A003'@'RestTutorial'"
14       },
15       "qualification": {
16         "name": "Length",
17         "language": "Language independent",
18         "identifier": "DEFAULT"
19       },
20       "values": [
21         "Decimal",
22         "3"
23       ]
24     },
25     {
26       "object": {
27         "id": "'A003'@'RestTutorial'"
28       },
29       "qualification": {
30         "name": "Front Color",
31         "language": "German",
32         "identifier": "DEFAULT"
33       },
34       "values": [
35         "Character string",
36         "rot"
37       ]
38     },
39     {
40       "object": {
41         "id": "'A003'@'RestTutorial'"
42       },

```

```

43     "qualification": {
44         "name": "Front Color",
45         "language": "English",
46         "identifier": "DEFAULT"
47     },
48     "values": [
49         "Character string",
50         "red"
51     ]
52 }
53 ]
54 }

```

13.6.5.14 REST List API Troubleshooting and How To's

A collection of small how to's and hints as how to work with the list api. This page also serves as a troubleshooting page which summarizes solutions we encountered during practical usage of the List API.

User and User Groups

Assigning User Groups to a User

Use the following REST API call to assign user groups to a user:

POST /rest/V1.0/list/User

```

1  {
2      "columns": [{"identifier": "User.UserGroups"}],
3      "rows": [{
4          "object": { "id": "'restuser'" },
5          "values": [ [ { "id" : "'InfaBPM'" }, { "id" :
6              "'Categorization'" } ] ]
7      }]
8  }

```

In this example the user *restuser* gets the user groups *InfaBPM* and *Categorization* assigned.



To add just one user group no list element is needed. For example
 { "id" : "'InfaBPM'" }

would be sufficient to assign group *InfaBPM* to a user.

Characteristics

- When searching for a string value which is in the entity proxy format like:
`characteristic('StringCharacteristic') = "'LookupValue'@'Lookup'",` the value will still be interpreted as a lookup value and not as a string.
- Please make sure when using the search that there is **no leading whitespace** in the URL: `list/Article/bySearch?query= characteristic('23124123') is empty`

Structure Groups

Query the list of items assigned to multiple structure groups of a structure

Use the following REST request to get a list of all items assigned to "SG_1" and "SG_2" of "Structure" structure.

Note: in this request, the "ArticleStructureGroupMap" (read-only) entity is used since there is one entry for each item - structure group mapping.

```
1 http://localhost:1512/rest/V1.0/list/Article/bySearch?query=(not
  (ArticleStructureGroupMap.StructureProxy("Structure",
    "'SG_1'@'Structure'" ) is empty) ) and (not
  (ArticleStructureGroupMap.StructureProxy("Structure",
    "'SG_2'@'Structure'" ) is empty))&fields=Article.SupplierAID
```

Updating Product's Structure Group with Rest API


Use the following REST API call to change the structure group of a product:

POST /rest/V1.0/list/Product2G

```
1 {
2   "entityIdentifier": "Product2G",
3   "rowCount": 1,
4   "columnCount": 1,
5   "columns": [
6     {
7       "identifier": "Product2GStructureMap.ManualMap(\"Brands\")"
8     }
9   ],
10  "rows": [{
11    "object": {
12      "id": "87534@1",
13      "label": "MGRTN00000003381699"
14    },
15    "values": [ "NEW BRAND" ]
16  }]
```

```
17    }
```

In this example the object with the id 87534@1 is assigned to the "NEW BRAND" structure group within the Brands structure system.

 You have to specify the Manual Map fields as a list of Strings, not entity items. You need to set the actual string identifier of the structure groups, not using the entity item syntax with the internal ID's.

You can also use this approach to change structure groups of articles and variants.

Access to structure group fails

The following call fails:

```
GET http://<hostname>:1501/rest/V1.0/list/StructureGroup/byParentGroup?
structureGroup=20153@17969&fields=StructureGroup.Level
```

with error:

```
ERROR [qtp2129920-124] [FragmentManager] execute: Error while initializing resp.
executing fragments
java.lang.IllegalStateException: Couldn't find a column index for column identifier:
com.heiler.ppm.structure.db.model.StructureGroupRevision.structureId
    at
com.heiler.ppm.fragment.server.hql.FragmentHqlContext.getColumnIndex(FragmentHqlConte
xt.java:238)
```

while the following succeeds:

```
GET http://<hostname>:1501/rest/V1.0/list/StructureGroup/byParentGroup?
structureGroup=20153@17969&fields=StructureGroup.Level,StructureGroup.StructureProxy
```

StructureGroup levels

Query *bySearch* on an attribute which is based on an enumeration:

```
GET http://<hostname>:1501/rest/V1.0/list/Article/bySearch?
query=Article.Channelkennzeichen contains "technisat"
```

- A multi-selection should also be possible, e.g. "technisat.de, ekshop.de"
- In general there is also the problem, that the data also has to be written. How could this be achieved? The DB contains comma separated values.

Filtering on structure group attribute

The following call for example returns the structure group attributes:

```
http://<hostname>:1501/rest/V1.0/list/StructureGroup/StructureGroupAttribute/
bySearch?
  query=StructureGroup.Identifier%20=%20%221365663226877%22
  &structure=TechniSat
  &fields=StructureGroupAttributeValue.Value
  &qualificationFilter=language(de)
```

It would be helpful a filter on sub attributes could be added to the query, e.g.

```
StructureGroupAttribute.IsMandatory = false
```

however adding this filter to the query leads to an internal error:

```
java.lang.NullPointerException
  at
  com.heiler.ppm.search.server.internal.hql.exp.logicalkey.DefaultLogicalKeyGenerator.g
enerateHQLCondition(DefaultLogicalKeyGenerator.java:97)
```

Not possible to search by Structure Group Attribute Value

The following call:

```
GET http://localhost:1501/rest/V1.0/list/StructureGroup/bySearch?metaData=true
  &structure=10024
  &query=StructureGroupAttributeValue.Value("Brand ID",en,DEFAULT) equals "TEST"
  &pageSize=1000&orderBy=0-ASC
  &fields=StructureGroup.Identifier
```

leads to the following internal error:

```
ERROR [qtp461258525-162] [CoreExceptionMapper] Error while interpreting object
"StructureGroupAttributeValue.Value("BrandID","en",DEFAULT) equals "TEST":
Error while parsing condition 'StructureGroupAttributeValue.Value("Brand
ID","en",DEFAULT) equals "TEST":
Unknown enum entry provided for enum-based field (StructureGroupAttributeValue.Value)
- Entry for synonym 'TEST' not found!
com.heiler.ppm.std.core.dsl.DslParseException: Error while interpreting object
"StructureGroupAttributeValue.Value("Brand ID","en",DEFAULT) equals "TEST":
    at
com.heiler.ppm.std.core.internal.dsl.ParseUtils.performParse(ParseUtils.java:53)
    at com.heiler.ppm.search.core.hsqa.HSQParser.parseSearchQuery(HSQParser.java:62)
```

See also <https://mysupport.informatica.com/message/89002?et=watches.email.thread>

Media Assets

Get media asset document with attributes fails

The following call:

```
GET http://<hostname>:1501/rest/V1.0/list/MediaAsset/MediaAssetDocument/byItems?
items=14&fields=MediaAssetDocument.Identifier,MediaAssetDocumentAttributes.Resolution
```

leads to an internal error:

```
java.lang.IllegalArgumentException: Sort settings at index = 0 specify a field path
for which the list model doesn't have appropriate column
    at
com.heiler.ppm.std.core.list.util.ListEntryComparator.<init>(ListEntryComparator.java
:269)
```

The following call fails with: Entity 'MediaAssetDocumentAttributes' is not a direct child entity of entity 'MediaAsset'.

```
GET http://localhost:1501/rest/V1.0/list/MediaAsset/MediaAssetDocumentAttributes/
byItems?items=14
```

How to read and write values for characteristics



This document describes how you can use the service api to read and write item, product or variant values for characteristics.

Characteristics are the new alternative to attributes. They allow to store additional data on products, variants

and items without the need to adjust the repository and without the restriction to have the information on each and every product, variant and item. Characteristics are similar to attributes in that you can configure them to have a specific data type and define if they are mandatory or multi value. They are also dynamic in nature and can be changed during runtime. In contrast to attributes, they allow a hierarchy of values and dependencies between them. This allows for a value to only be maintained dependent on the selected value for the parent characteristic.

Please see the Knowledge Base article "Characteristics - Dynamic data model" for more details on the possibilities and the general handling within Product 360.

The Service API for characteristics follows the general List API contract and has the same parameters. Please make yourself familiar with the [REST List API](#) (see page 463) before you read this how-to.

Definitions

Name	
<code>Characteristic</code>	The model definition. Name, Datatype, Lookup Values, etc.
<code>CharacteristicRecord</code>	A characteristic record contains the actual value for the characteristic for a specific entity-item like item, product or variant. Additionally to the value itself it also contains the reference to the characteristic as well as a unique key of the record and the reference to the parent record.
<code>ArticleCharacteristicValue</code> , <code>Product2GCharacteristicValue</code> , , <code>VariantCharacteristicValue</code>	The repository entity for a characteristic record of the corresponding root entity
<code>SimpleArticleCharacteristicValue</code> , <code>SimpleProduct2GCharacteristicValue</code> , <code>SimpleVariantCharacteristicValue</code>	The repository entity for a simple characteristic record of the corresponding root entity. A simple characteristic is a root characteristic without children. Values for simple characteristics can be assigned to items, products or variants by specifying only the simple characteristic and the language.

Qualifications

Qualification	Datatype	
characteristic	ENTITY_ITEM	The characteristic to which this value belongs as ENTITY_ITEM (see page 436)
rootCharacteristic	ENTITY_ITEM	The root characteristic (not necessarily the parent) of the characteristic as ENTITY_ITEM (see page 436). <i>This qualification is only used for read access, it can be omitted for write requests.</i>
recordKey	String	The key of the characteristic record, is unique within the characteristic. The default value for this qualification is 0000.0000.RK The record key is mandatory in case there are multiple records for the same characteristic. We recommend to use the default record key as long as you don't have multiple records for the same characteristic, this will allow you to obtain values for this characteristic also by means of fully qualifying a field.
language	String	The key or code or name of the language in which the value should be interpreted. This defaults to -1 for characteristics which are not defined as language specific. Only Text or MIME characteristics can be language dependent.

Datatypes

Characteristic Datatype	Corresponding Rest Datatype
TEXT	STRING (see page 436)
INTEGER	INTEGER (see page 436)

Characteristic Datatype	Corresponding Rest Datatype
DECIMAL	DECIMAL (see page 437)
BOOLEAN	BOOLEAN (see page 438)
DATE	DATE (see page 437)
DATETIME	DATETIME (see page 438)
MIME	MIME_VALUE (see page 440)
LOOKUP	ENTITY_ITEM (see page 438)
NONE	<p>This datatype is typically used to build logical groups of characteristics with a common parent. For example: Dimensions with the children height, width and depth. Records of characteristics with this datatype do not hold an own value.</p>

Fields

The field which contains the actual value of a characteristic record is

`ArticleCharacteristicValueLang.Value`

This field is always a multi-value field which means it will always be returned as array. Some characteristics are multi-value and others are not, in this case the client can treat them all the same. The datatype with regards to the Service API is ANY, which means any of the datatypes which are supported by the characteristics will be returned - depending on the definition of the record's characteristic.

Special Case: LookupValues

In order to provide the generic Field transition functionality for characteristic values of datatype lookup a specialized field is provided: `ArticleCharacteristicValue.LookupValue`

This field only contains a value if the datatype of the characteristic is lookup value and it can actually be used to build a transition field like:

`ArticleCharacteristicValue.LookupValue → LookupValueLang.Description(de)` which would return the German description of this lookup value.

Read Access

CharacteristicsValues can be read in the same way as any other sub-entity can be read. Either by fully qualifying a field parameter with *all* qualifications or by requesting all records for this sub-entity and specifying *some* of the qualifications as optional filters.

The nature of the characteristics makes it not easy to use in a fully qualified manner, as the `recordKey` can be different for each item/characteristic combination.

So one fully qualified column might not return values for all items in the table as the record key must not match.

Examples based on the root entity

As mentioned above, reading characteristic values with fully qualified fields is difficult in general. But it is quite easy for simple characteristics as the following examples show.

GET Simple characteristic values for Item

```
http://<server>:<port>/rest/V2.0/list/Article/bySearch?query=Article.SupplierAID
startsWith
"ItemRestTest"&fields=SimpleArticleCharacteristicValueLang.Value(COLOR,-1),SimpleArticleCharacteristicValueLang.Value(STATEMENT,en),SimpleArticleCharacteristicValueLang.Value(STATEMENT,de)&includeLabels=true
```

As a result the following structure of characteristic values will be provided

Simple characteristic values for Item

```
1  {
2    "cacheId": "no-cache",
3    "entityIdentifier": "Article",
4    "totalSize": 2,
5    "startIndex": 0,
6    "pageSize": 100,
7    "rowCount": 2,
8    "columnCount": 0,
9    "columns": [],
10   "rows": [
11     {
12       "object": {
13         "id": "158684@1",
```



```

14         "label": "ItemRestTest",
15         "entityId": 1000
16     },
17     "values": [
18         [
19             {
20                 "id": "1455@945",
21                 "label": "green",
22                 "entityId": 7300
23             }
24         ],
25         [
26             "Statement"
27         ],
28         [
29             "Erklärung"
30         ]
31     ]
32 },
33 {
34     "object": {
35         "id": "158685@1",
36         "label": "ItemRestTest2",
37         "entityId": 1000
38     },
39     "values": [
40         [
41             {
42                 "id": "1456@945",
43                 "label": "blue",
44                 "entityId": 7300
45             }
46         ],
47         [
48             "No statement necessary"
49         ],
50         [
51             "Keine Erklärung nötig"
52         ]
53     ]
54 }
55 ]
56 }
```

GET MIME value meta data and MIME values for Product

GET MIME value meta data for Product

```
http://<server>:<port>/rest/V2.0/list/Product2G/bySearch?query=Product2G.ProductNo
startsWith
"RestTestProduct"&fields=SimpleProduct2G.CharacteristicValueLang.Value(LOGO,-1)&includeLabels=true
```

As a result the following structure of characteristic values will be provided

MIME values meta data for Product

```

1  {
2    "cacheId": "no-cache",
3    "entityIdentifier": "Product2G",
4    "totalSize": 2,
5    "startIndex": 0,
6    "pageSize": 100,
7    "rowCount": 2,
8    "columnCount": 0,
9    "columns": [],
10   "rows": [
11     {
12       "object": {
13         "id": "158686@1",
14         "label": "RestTestProduct",
15         "entityId": 1100
16       },
17       "values": [
18         [
19           {
20             "label": "logo.png",
21             "mimeType": "image/png",
22             "relativeFilePath": "29\\27\\19\\
23             \\logo.908f8b16002fbfb2_4955b4f5_170d4d6aaae_-7e4a.png"
24           }
25         ]
26       },
27       {
28         "object": {
29           "id": "158687@1",
30           "label": "RestTestProduct2",
31           "entityId": 1100
32         },
33         "values": [
34           [
35             {
36               "label": "logo_global.png",
37               "mimeType": "image/png",
38               "relativeFilePath": "10\\46\\16\\
39               \\logo_global.908f8b16002fbfb2_4955b4f5_170d4d6aaae_-7e3b.png"

```

```

40         ]
41     ]
42 }
43 ]
44 }
```

GET MIME values for Product

```

http://<server>:<port>/rest/V2.0/list/Product2G/mimes/bySearch?
query=Product2G.ProductNo startsWith
"RestTestProduct"&fields=SimpleProduct2GCharacteristicValueLang.Value(LOGO,-1)&includeLabels=true
```

As result a zip file containing the MIME files will be provided

Examples based on the sub-entity

GET Characteristic values for Item

```

http://<server>:<port>/rest/V2.0/list/Article/ArticleCharacteristicValue/bySearch?
query=Article.SupplierAID equals
"ArticleRestTest"&fields=ArticleCharacteristicValue.RecordKey,ArticleCharacteristicValueLang.Value&includeLabels=true
```

As a result the following structure of characteristic values will be provided

Characteristic values for Items

```

1  {
2      "cacheId": "no-cache",
3      "entityIdentifier": "ArticleCharacteristicValue",
4      "totalSize": 1,
5      "startIndex": 0,
6      "pageSize": 100,
7      "rowCount": 1,
8      "columnCount": 0,
9      "columns": [],
10     "rows": [
11         {
12             "object": {
13                 "id": "10126@1",
14                 "label": "ArticleRestTest",
15                 "entityId": 1000
16             },

```

```

17         "qualification": {
18             "recordKey": "0000.0000.RK",
19             "rootCharacteristic": {
20                 "id": "10769",
21                 "label": "LookupRestArticleTestCharacteristic
[LookupRestArticleTestCharacteristic]",
22                 "entityId": 8000
23             },
24             "parentRecordKey": "root",
25             "language": "",
26             "characteristic": {
27                 "id": "10769",
28                 "label": "LookupRestArticleTestCharacteristic
[LookupRestArticleTestCharacteristic]",
29                 "entityId": 8000
30             }
31         },
32         "values": [
33             "0000.0000.RK",
34             [
35                 {
36                     "id": "10615@3",
37                     "label": "ActualValue",
38                     "entityId": 7300
39                 }
40             ]
41         ]
42     }
43 ]
44 }

```

GET Characteristic values for Product

```

http://<server>:<port>/rest/V2.0/list/Product2G/Product2GCharacteristicValue/
bySearch?query=Product2G.ProductNo equals
"RestTestProductNo"&fields=Product2GCharacteristicValue.RecordKey,Product2GCharacteri
sticValueLang.Value&includeLabels=true

```

As a result the following structure of characteristic values will be provided

Characteristic values for Product

```

1  {
2      "cacheId": "no-cache",
3      "entityIdentifier": "Product2GCharacteristicValue",
4      "totalSize": 1,
5      "startIndex": 0,
6      "pageSize": 100,

```

```

7      "rowCount": 10,
8      "columnCount": 0,
9      "columns": [],
10     "rows": [
11         {
12             "object": {
13                 "id": "10123@1",
14                 "label": "RestTestProductNo",
15                 "entityId": 1100
16             },
17             "qualification": {
18                 "recordKey": "0000.0000.RK",
19                 "rootCharacteristic": {
20                     "id": "10759",
21                     "label": "TextRestTestCharacteristic
[TextRestTestCharacteristic]",
22                     "entityId": 8000
23                 },
24                 "parentRecordKey": "root",
25                 "language": "-1",
26                 "characteristic": {
27                     "id": "10759",
28                     "label": "TextRestTestCharacteristic
[TextRestTestCharacteristic]",
29                     "entityId": 8000
30                 }
31             },
32             "values": [
33                 "0000.0000.RK",
34                 [
35                     "TextRestTestCharacteristicValue"
36                 ]
37             ]
38         },
39     ]
40 }

```

GET Characteristic values for Variant

```

http://<server>:<port>/rest/V2.0/list/Variant/VariantCharacteristicValue/bySearch?
query=Variant.VariantNo equals
"VariantRestTest"&fields=VariantCharacteristicValue.RecordKey,VariantCharacteristicVa
lueLang.Value&includeLabels=true

```

As a result the following structure of characteristic values will be provided

Characteristic values for Variant

```

1  {
2      "cacheId": "no-cache",
3      "entityIdentifier": "VariantCharacteristicValue",
4      "totalSize": 1,
5      "startIndex": 0,
6      "pageSize": 100,
7      "rowCount": 10,
8      "columnCount": 0,
9      "columns": [],
10     "rows": [
11         {
12             "object": {
13                 "id": "10123@1",
14                 "label": "RestTestVariantNo",
15                 "entityId": 1100
16             },
17             "qualification": {
18                 "recordKey": "0000.0000.RK",
19                 "rootCharacteristic": {
20                     "id": "10759",
21                     "label": "TextRestTestCharacteristic
[TextRestTestCharacteristic]",
22                     "entityId": 8000
23                 },
24                 "parentRecordKey": "root",
25                 "language": "-1",
26                 "characteristic": {
27                     "id": "10759",
28                     "label": "TextRestTestCharacteristic
[TextRestTestCharacteristic]",
29                     "entityId": 8000
30                 }
31             },
32             "values": [
33                 "0000.0000.RK",
34                 [
35                     "TextRestTestCharacteristicValue"
36                 ]
37             ]
38         },
39     ]
40 }

```

GET Characteristic values and lookup value codes for Item

```
http://<server>:<port>/rest/V2.0/list/Article/ArticleCharacteristicValue/bySearch?
query=Article.SupplierAID equals
"ItemRestTest"&fields=ArticleCharacteristicValue.RecordKey,ArticleCharacteristicValue
Lang.Value,ArticleCharacteristicValue.LookupValue ->
LookupValue.Code&includeLabels=true
```

As a result the following structure of characteristic values and lookup value codes will be provided

Characteristic values and lookup value codes for Item

```
1  {
2    "cacheId": "no-cache",
3    "entityIdentifier": "ArticleCharacteristicValue",
4    "totalSize": 1,
5    "startIndex": 0,
6    "pageSize": 100,
7    "rowCount": 1,
8    "columnCount": 0,
9    "columns": [],
10   "rows": [
11     {
12       "object": {
13         "id": "158684@1",
14         "label": "ItemRestTest",
15         "entityId": 1000
16       },
17       "qualification": {
18         "recordKey": "0000.0000.RK",
19         "rootCharacteristic": {
20           "id": "3422",
21           "label": "Color [COLOR]",
22           "entityId": 8000
23         },
24         "parentRecordKey": "root",
25         "language": "-1",
26         "characteristic": {
27           "id": "3422",
28           "label": "Color [COLOR]",
29           "entityId": 8000
30         }
31       },
32       "values": [
33         "0000.0000.RK",
34         [
35           {
36             "id": "1455@945",
37             "label": "green",
38             "entityId": 7300
39           }
40         ],

```

```

41         "GREEN"
42     ]
43 }
44 ]
45 }

```

GET Characteristic values for Item for specific languages

```

http://<server>:<port>/rest/V2.0/list/Article/ArticleCharacteristicValue/bySearch?
query=Article.SupplierAID equals
"ItemRestTest"&fields=ArticleCharacteristicValueLang.Value&includeLabels=true&qualifi
cationFilter=language(en,de,fr)

```

As a result the following structure of characteristic values will be provided

Characteristic values for Item for specific languages

```

1  {
2      "cacheId": "no-cache",
3      "entityIdentifier": "ArticleCharacteristicValue",
4      "totalSize": 1,
5      "startIndex": 0,
6      "pageSize": 100,
7      "rowCount": 3,
8      "columnCount": 0,
9      "columns": [],
10     "rows": [
11         {
12             "object": {
13                 "id": "158684@1",
14                 "label": "ItemRestTest",
15                 "entityId": 1000
16             },
17             "qualification": {
18                 "recordKey": "0000.0000.RK",
19                 "rootCharacteristic": {
20                     "id": "4300",
21                     "label": "Statement [STATEMENT]",
22                     "entityId": 8000
23                 },
24                 "parentRecordKey": "root",
25                 "language": "9",
26                 "characteristic": {
27                     "id": "4300",
28                     "label": "Statement [STATEMENT]",
29                     "entityId": 8000
30                 }
31             },

```



```

32         "values": [
33             [
34                 "Statement"
35             ]
36         ]
37     },
38     {
39         "object": {
40             "id": "158684@1",
41             "label": "ItemRestTest",
42             "entityId": 1000
43         },
44         "qualification": {
45             "recordKey": "0000.0000.RK",
46             "rootCharacteristic": {
47                 "id": "4300",
48                 "label": "Statement [STATEMENT]",
49                 "entityId": 8000
50             },
51             "parentRecordKey": "root",
52             "language": "12",
53             "characteristic": {
54                 "id": "4300",
55                 "label": "Statement [STATEMENT]",
56                 "entityId": 8000
57             }
58         },
59         "values": [
60             [
61                 "Déclaration"
62             ]
63         ]
64     },
65     {
66         "object": {
67             "id": "158684@1",
68             "label": "ItemRestTest",
69             "entityId": 1000
70         },
71         "qualification": {
72             "recordKey": "0000.0000.RK",
73             "rootCharacteristic": {
74                 "id": "4300",
75                 "label": "Statement [STATEMENT]",
76                 "entityId": 8000
77             },
78             "parentRecordKey": "root",
79             "language": "7",
80             "characteristic": {
81                 "id": "4300",
82                 "label": "Statement [STATEMENT]",
83                 "entityId": 8000
84             }
85         }
86     }
87 }

```

```

85         },
86         "values": [
87             [
88                 "Erklärung"
89             ]
90         ]
91     }
92 ]
93 }

```

GET versioned Characteristic values for Product

```

http://<server>:<port>/rest/V2.0/list/Product2G/Product2GCharacteristicValue/
bySearch?query=Product2G.ProductNo equals
"P_3"&revision='ProductVersion2'&fields=Product2GCharacteristicValue.RecordKey,Produc
t2GCharacteristicValueLang.Value&includeLabels=true

```

As a result the following structure of characteristic values will be provided

Versioned Characteristic values for Product

```

1  {
2      "cacheId": "no-cache",
3      "entityIdentifier": "Product2GCharacteristicValue",
4      "revision": {
5          "id": "10",
6          "label": "ProductVersion2",
7          "entityId": 5600
8      },
9      "totalSize": 1,
10     "startIndex": 0,
11     "pageSize": 100,
12     "rowCount": 5,
13     "columnCount": 0,
14     "columns": [],
15     "rows": [
16         {
17             "object": {
18                 "id": "17@1",
19                 "label": "P_3",
20                 "entityId": 1100
21             },
22             "qualification": {
23                 "recordKey": "0000.0000.RK",
24                 "rootCharacteristic": {
25                     "id": "24",
26                     "label": "Marketing information
[marketingInformation]",

```

```

27         "entityId": 8000
28     },
29     "parentRecordKey": "0000.0000.RK",
30     "language": "-1",
31     "characteristic": {
32         "id": "26",
33         "label": "Coupon family code [couponFamilyCode]",
34         "entityId": 8000
35     }
36 },
37 "values": [
38     "0000.0000.RK",
39     [
40         "54321"
41     ]
42 ]
43 },
44 {
45     "object": {
46         "id": "17@1",
47         "label": "P_3",
48         "entityId": 1100
49     },
50     "qualification": {
51         "recordKey": "0000.0000.RK",
52         "rootCharacteristic": {
53             "id": "24",
54             "label": "Marketing information
[marketingInformation]",
55             "entityId": 8000
56         },
57         "parentRecordKey": "root",
58         "language": "",
59         "characteristic": {
60             "id": "24",
61             "label": "Marketing information
[marketingInformation]",
62             "entityId": 8000
63         }
64     },
65     "values": [
66         "0000.0000.RK",
67         [
68             ""
69         ]
70     ]
71 },
72 {
73     "object": {
74         "id": "17@1",
75         "label": "P_3",
76         "entityId": 1100
77     },

```

```

78         "qualification": {
79             "recordKey": "0000.0000.RK",
80             "rootCharacteristic": {
81                 "id": "24",
82                 "label": "Marketing information
[marketingInformation]",
83                 "entityId": 8000
84             },
85             "parentRecordKey": "0000.0000.RK",
86             "language": "7",
87             "characteristic": {
88                 "id": "31",
89                 "label": "Included accessories
[tradeItemIncludedAccessories]",
90                 "entityId": 8000
91             }
92         },
93         "values": [
94             "0000.0000.RK",
95             [
96                 "Version ProductVersion2"
97             ]
98         ]
99     },
100     {
101         "object": {
102             "id": "17@1",
103             "label": "P_3",
104             "entityId": 1100
105         },
106         "qualification": {
107             "recordKey": "0000.0000.RK",
108             "rootCharacteristic": {
109                 "id": "24",
110                 "label": "Marketing information
[marketingInformation]",
111                 "entityId": 8000
112             },
113             "parentRecordKey": "0000.0000.RK",
114             "language": "9",
115             "characteristic": {
116                 "id": "31",
117                 "label": "Included accessories
[tradeItemIncludedAccessories]",
118                 "entityId": 8000
119             }
120         },
121         "values": [
122             "0000.0000.RK",
123             [
124                 "Version ProductVersion2"
125             ]
126     ]

```

```

127     },
128     {
129         "object": {
130             "id": "17@1",
131             "label": "P_3",
132             "entityId": 1100
133         },
134         "qualification": {
135             "recordKey": "0000.0000.RK",
136             "rootCharacteristic": {
137                 "id": "24",
138                 "label": "Marketing information
[marketingInformation]",
139                 "entityId": 8000
140             },
141             "parentRecordKey": "0000.0000.RK",
142             "language": "-1",
143             "characteristic": {
144                 "id": "25",
145                 "label": "Build-In product type [buildInProductType]",
146                 "entityId": 8000
147             }
148         },
149         "values": [
150             "0000.0000.RK",
151             [
152                 "Version ProductVersion2"
153             ]
154         ]
155     }
156 ]
157 }

```

GET versioned Characteristic values for Variant

```

http://<server>:<port>/rest/V2.0/list/Variant/VariantCharacteristicValue/bySearch?
query=Variant.VariantNo equals
"V\_3"&revision='ProductVersion2'&fields=VariantCharacteristicValue.RecordKey,VariantC
haracteristicValueLang.Value&includeLabels=true

```

As a result the following structure of characteristic values will be provided

Versioned Characteristic values for Variant

```

1  {
2      "cacheId": "no-cache",
3      "entityIdentifier": "VariantCharacteristicValue",
4      "revision": {

```

```

5      "id": "10",
6      "label": "ProductVersion2",
7      "entityId": 5600
8  },
9  "totalSize": 1,
10 "startIndex": 0,
11 "pageSize": 100,
12 "rowCount": 2,
13 "columnCount": 0,
14 "columns": [],
15 "rows": [
16     {
17         "object": {
18             "id": "94@1",
19             "label": "V_3",
20             "entityId": 1200
21         },
22         "qualification": {
23             "recordKey": "0000.0000.RK",
24             "rootCharacteristic": {
25                 "id": "24",
26                 "label": "Marketing information
[marketingInformation]",
27                 "entityId": 8000
28             },
29             "parentRecordKey": "root",
30             "language": "",
31             "characteristic": {
32                 "id": "24",
33                 "label": "Marketing information
[marketingInformation]",
34                 "entityId": 8000
35             }
36         },
37         "values": [
38             "0000.0000.RK",
39             [
40                 ""
41             ]
42         ]
43     },
44     {
45         "object": {
46             "id": "94@1",
47             "label": "V_3",
48             "entityId": 1200
49         },
50         "qualification": {
51             "recordKey": "0000.0000.RK",
52             "rootCharacteristic": {
53                 "id": "24",
54                 "label": "Marketing information
[marketingInformation]",

```

```

55         "entityId": 8000
56     },
57     "parentRecordKey": "0000.0000.RK",
58     "language": "9",
59     "characteristic": {
60         "id": "39",
61         "label": "Key words [tradeItemKeyWords]",
62         "entityId": 8000
63     }
64 },
65 "values": [
66     "0000.0000.RK",
67     [
68         "Variant versioned with product in ProductVersion2"
69     ]
70 ]
71 }
72 ]
73 }
```

GET versioned Characteristic values for Item

```

http://<server>:<port>/rest/V2.0/list/Article/ArticleCharacteristicValue/bySearch?
query=Article.SupplierAID equals
"I_3"&revision='ProductVersion2'&fields=ArticleCharacteristicValue.RecordKey,ArticleC
haracteristicValueLang.Value&includeLabels=true
```

As a result the following structure of characteristic values will be provided

Versioned Characteristic values for Item

```

1  {
2      "cacheId": "no-cache",
3      "entityIdentifier": "ArticleCharacteristicValue",
4      "revision": {
5          "id": "10",
6          "label": "ProductVersion2",
7          "entityId": 5600
8      },
9      "totalSize": 1,
10     "startIndex": 0,
11     "pageSize": 100,
12     "rowCount": 2,
13     "columnCount": 0,
14     "columns": [],
15     "rows": [
16         {
17             "object": {
```

```

18         "id": "95@1",
19         "label": "I_3",
20         "entityId": 1000
21     },
22     "qualification": {
23         "recordKey": "0000.0000.RK",
24         "rootCharacteristic": {
25             "id": "24",
26             "label": "Marketing information
[marketingInformation]",
27             "entityId": 8000
28         },
29         "parentRecordKey": "0000.0000.RK",
30         "language": "9",
31         "characteristic": {
32             "id": "39",
33             "label": "Key words [tradeItemKeyWords]",
34             "entityId": 8000
35         }
36     },
37     "values": [
38         "0000.0000.RK",
39         [
40             "Item versioned with variant and product in
ProductVersion2"
41         ]
42     ]
43 },
44 {
45     "object": {
46         "id": "95@1",
47         "label": "I_3",
48         "entityId": 1000
49     },
50     "qualification": {
51         "recordKey": "0000.0000.RK",
52         "rootCharacteristic": {
53             "id": "24",
54             "label": "Marketing information
[marketingInformation]",
55             "entityId": 8000
56         },
57         "parentRecordKey": "root",
58         "language": "",
59         "characteristic": {
60             "id": "24",
61             "label": "Marketing information
[marketingInformation]",
62             "entityId": 8000
63         }
64     },
65     "values": [
66         "0000.0000.RK",

```



```

67         [
68             ""
69         ]
70     ]
71 }
72 ]
73 }
```

GET Characteristic values including unavailable records for Item

```

http://<server>:<port>/rest/V2.0/list/Article/ArticleCharacteristicValue/bySearch?
query=Article.SupplierAID equals
"UnavailableCharacteristicTest"&fields=ArticleCharacteristicValue.RecordKey,ArticleCh
aracteristicValueLang.Value
&includeUnavailable=true
```

As a result the following structure of characteristic values will be provided

Characteristic values for Item including unavailable records

```

1  {
2      "cacheId": "no-cache",
3      "entityIdentifier": "ArticleCharacteristicValue",
4      "totalSize": 1,
5      "startIndex": 0,
6      "pageSize": 100,
7      "rowCount": 1,
8      "columnCount": 0,
9      "columns": [],
10     "rows": [
11         {
12             "object": {
13                 "id": "10127@1",
14                 "entityId": 1000
15             },
16             "qualification": {
17                 "recordKey": "0000.0000.RK",
18                 "rootCharacteristic": {
19                     "id": "10770",
20                     "entityId": 8000
21                 },
22                 "parentRecordKey": "root",
23                 "language": "-1",
24                 "characteristic": {
25                     "id": "10770",
26                     "entityId": 8000
27                 }
28             },
29         },
30     ]
31 }
```

```

29         "values": [
30             "0000.0000.RK",
31             [
32                 "Some Unavailable Value"
33             ]
34         ]
35     }
36 ]
37 }

```

GET MIME value metadata and MIME values for Product**GET MIME value metadata for Product**

```

http://<server>:<port>/rest/V2.0/list/Product2G/Product2GCharacteristicValue/
bySearch?query=Product2G.ProductNo equals
"RestTestProductNo"&fields=Product2GCharacteristicValueLang.Value&includeLabels=true

```

MIME value metadata for Product

```

1  {
2      "cacheId": "no-cache",
3      "entityIdentifier": "Product2GCharacteristicValue",
4      "totalSize": 1,
5      "startIndex": 0,
6      "pageSize": 100,
7      "rowCount": 10,
8      "columnCount": 0,
9      "columns": [],
10     "rows": [
11         {
12             "object": {
13                 "id": "10123@1",
14                 "label": "RestTestProductNo",
15                 "entityId": 1100
16             },
17             "qualification": {
18                 "recordKey": "0000.0000.RK",
19                 "rootCharacteristic": {
20                     "id": "10765",
21                     "label": "MimeRestTestCharacteristic
[MimeRestTestCharacteristic]",
22                     "entityId": 8000
23                 },
24                 "parentRecordKey": "root",
25                 "language": "-1",
26                 "characteristic": {

```

```

27         "id": "10765",
28         "label": "MimeRestTestCharacteristic
[MimeRestTestCharacteristic]",
29         "entityId": 8000
30     },
31 },
32 "values": [
33     [
34         {
35             "label":
36             "880x495_cmsv2_298e3b01-877d-57e3-9ce0-0542084c5af4-3217366.jpg",
37             "mimeType": "image/jpeg",
38             "relativeFilePath": "35\\18\\14\\
39             \880x495_cmsv2_298e3b.b245cc1ddda0ddd7_1023f19d_16bad3f75bb_-7bf7.jpg"
40         }
41     ]
42 ]
43 }
```

GET MIME values for Product

```

http://<server>:<port>/rest/V2.0/list/Product2G/Product2GCharacteristicValue/mimes/
bySearch?query=Product2G.ProductNo equals
"RestTestProductNo"&fields=Product2GCharacteristicValueLang.Value
```

As result a zip file containing the MIME files will be provided

Write Access

Examples based on the root entity

As mentioned above, writing characteristic values with fully qualified fields is difficult in general. But it is quite easy for simple characteristics as the following example show.

POST Simple characteristic values for Item

```

http://<server>:<port>/rest/V2.0/list/Article
```

This POST request requires the following request body

Simple characteristic values for Item

```

1  {
2    "columns": [
3      { "identifier":
4        "SimpleArticleCharacteristicValueLang.Value(STATEMENT,en)" },
5      { "identifier":
6        "SimpleArticleCharacteristicValueLang.Value(STATEMENT,de)" },
7      { "identifier":
8        "SimpleArticleCharacteristicValueLang.Value(STATEMENT,fr)" }
9    ],
10   "rows": [
11     {
12       "object": { "id": "'ItemRestTest'@'MASTER'" }
13     },
14     "values": [
15       "Another EN statement",
16       "Another DE statement",
17       "Another FR statement"
18     ]
19   ]
20 }

```

As a result the following answer will be provided

Simple characteristic values for Item Result

```

1  {
2    "counters": {
3      "errors": 0,
4      "warnings": 0,
5      "createdObjects": 0,
6      "updatedObjects": 1,
7      "objectsWithErrors": 0,
8      "objectsWithWarnings": 0
9    },
10   "entries": [],
11   "objects": [
12     {
13       "row": 0,
14       "object": {
15         "id": "2@1",
16         "label": "ItemRestTest",
17         "entityId": 1000
18       },
19       "status": [
20         "UPDATED"
21       ]
22     }
23   ]
24 }

```

```

22     }
23   ]
24 }

```

Examples based on the sub-entity

POST Characteristic values for Item: lookup value

<http://<server>:<port>/rest/V2.0/list/Article/ArticleCharacteristicValue>

This POST request requires the following request body

Characteristic values for Item

```

1  {
2    "columns": [
3      {
4        "identifier": "ArticleCharacteristicValueLang.Value"
5      }
6    ],
7    "rows": [
8      {
9        "object": {
10         "id": "10126@1"
11       },
12       "qualification": {
13         "recordKey": "0000.0000.RK",
14         "rootCharacteristic": {
15           "id": "10769"
16         },
17         "parentRecordKey": "root",
18         "language": "-1",
19         "characteristic": {
20           "id": "10769",
21           "label": "LookupRestArticleTestCharacteristic
22           [LookupRestArticleTestCharacteristic]",
23           "entityId": 8000
24         }
25       },
26       "values": [
27         [
28           {
29             "id": "10617@3",
30             "label": "ActualLookupValueLabel",
31             "entityId": 7300

```

```

31         }
32     ]
33 ]
34 }
35 ]
36 }

```

As a result the following answer will be provided

Characteristic values for Item Result

```

1  {
2      "counters": {
3          "errors": 0,
4          "warnings": 0,
5          "createdObjects": 0,
6          "updatedObjects": 1,
7          "objectsWithErrors": 0,
8          "objectsWithWarnings": 0
9      },
10     "entries": [],
11     "objects": [
12         {
13             "row": 0,
14             "object": {
15                 "id": "10126@1",
16                 "label": "ArticleRestTest",
17                 "entityId": 1100
18             },
19             "status": [
20                 "UPDATED"
21             ]
22         }
23     ]
24 }

```

POST Characteristic values for Product:: language-dependent string values

<http://<server>:<port>/rest/V2.0/list/Product2G/Product2GCharacteristicValue>

This POST request requires the following request body

Characteristic values for Product

```

1  {
2      "columns": [

```

```

3      {
4        "identifier": "Product2GCharacteristicValueLang.Value"
5      }
6    ],
7    "rows": [
8      {
9        "object": {
10          "id": "'RestTestProduct'@'MASTER'"
11        },
12        "qualification": {
13          "recordKey": "0000.0000.RK",
14          "rootCharacteristic": {
15            "id": "4300"
16          },
17          "parentRecordKey": "root",
18          "language": "de",
19          "characteristic": {
20            "id": "4300"
21          }
22        },
23        "values": [
24          [
25            "CHANGED German statement"
26          ]
27        ]
28      },
29      {
30        "object": {
31          "id": "'RestTestProduct'@'MASTER'"
32        },
33        "qualification": {
34          "recordKey": "0000.0000.RK",
35          "rootCharacteristic": {
36            "id": "4300"
37          },
38          "parentRecordKey": "root",
39          "language": "en",
40          "characteristic": {
41            "id": "4300"
42          }
43        },
44        "values": [
45          [
46            "CHANGED English statement"
47          ]
48        ]
49      }
50    ]
51  }

```

As a result the following answer will be provided

Characteristic values for Product Result

```

1  {
2      "counters": {
3          "errors": 0,
4          "warnings": 0,
5          "createdObjects": 0,
6          "updatedObjects": 1,
7          "objectsWithErrors": 0,
8          "objectsWithWarnings": 0
9      },
10     "entries": [],
11     "objects": [
12         {
13             "row": 0,
14             "object": {
15                 "id": "158686@1",
16                 "label": "RestTestProduct",
17                 "entityId": 1100
18             },
19             "status": [
20                 "UPDATED"
21             ]
22         }
23     ]

```

POST Characteristic values for Variant: MIME values**Step 1: Upload MIME zip file**

In a first step you have to upload a zip file containing all files that are to be used as MIME values. Details can be found in the chapter [REST File API \(see page 738\)](#).

The result is needed for the next step.

Result of MIME file upload

```

{
  "id": "6942cf90-a4c4-449d-896a-51bdc8e45906",
  "originalFilename": "logos.zip"
}

```

Step 2: Use MIME data to set characteristic values

Important: The uploaded MIME archive can be used for one POST request only, it cannot be re-used for additional requests. So, if that archive contains multiple files, they should all be used within one request.


```
http://<server>:<port>/rest/V2.0/list/Variant/VariantCharacteristicValue
```

This POST request requires the following request body, the result of the file upload is used as `mimeValueArchives` value:

Characteristic values for Variant

```

1  {
2    "columns": [
3      {
4        "identifier": "VariantCharacteristicValueLang.Value"
5      }
6    ],
7    "rows": [
8      {
9        "object": {
10         "id": "'RestTestVariant'@'MASTER'"
11       },
12       "qualification": {
13         "recordKey": "0000.0000.RK",
14         "rootCharacteristic": {
15           "id": "4298"
16         },
17         "parentRecordKey": "root",
18         "language": "-1",
19         "characteristic": {
20           "id": "4298"
21         }
22       },
23       "values": [
24         [
25           {
26             "relativeFilePath": "logo.png"
27           }
28         ]
29       ]
30     },
31     {
32       "object": {
33         "id": "'RestTestVariant2'@'MASTER'"
34       },
35       "qualification": {
36         "recordKey": "0000.0000.RK",
37         "rootCharacteristic": {
38           "id": "4298"
39         },
40         "parentRecordKey": "root",
41         "language": "-1",
42         "characteristic": {
43           "id": "4298"

```

```

44     }
45   },
46   "values": [
47     [
48       {
49         "relativeFilePath": "logo_global.png"
50       }
51     ]
52   ]
53 }
54 ],
55 "mimeValueArchives": [
56   {
57     "id": "6942cf90-a4c4-449d-896a-51bdc8e45906",
58     "originalFilename": "logos.zip"
59   }
60 ]
61 }

```

As a result the following answer will be provided

Characteristic values for Variant Result

```

1  {
2    "counters": {
3      "errors": 0,
4      "warnings": 0,
5      "createdObjects": 0,
6      "updatedObjects": 2,
7      "objectsWithErrors": 0,
8      "objectsWithWarnings": 0
9    },
10   "entries": [],
11   "objects": [
12     {
13       "row": 0,
14       "object": {
15         "id": "158688@1",
16         "label": "RestTestVariant",
17         "entityId": 1200
18       },
19       "status": [
20         "UPDATED"
21       ]
22     },
23     {
24       "row": 1,
25       "object": {
26         "id": "158689@1",
27         "label": "RestTestVariant2",
28         "entityId": 1200

```

```
29         },
30         "status": [
31             "UPDATED"
32         ]
33     }
34 ]
35 }
```

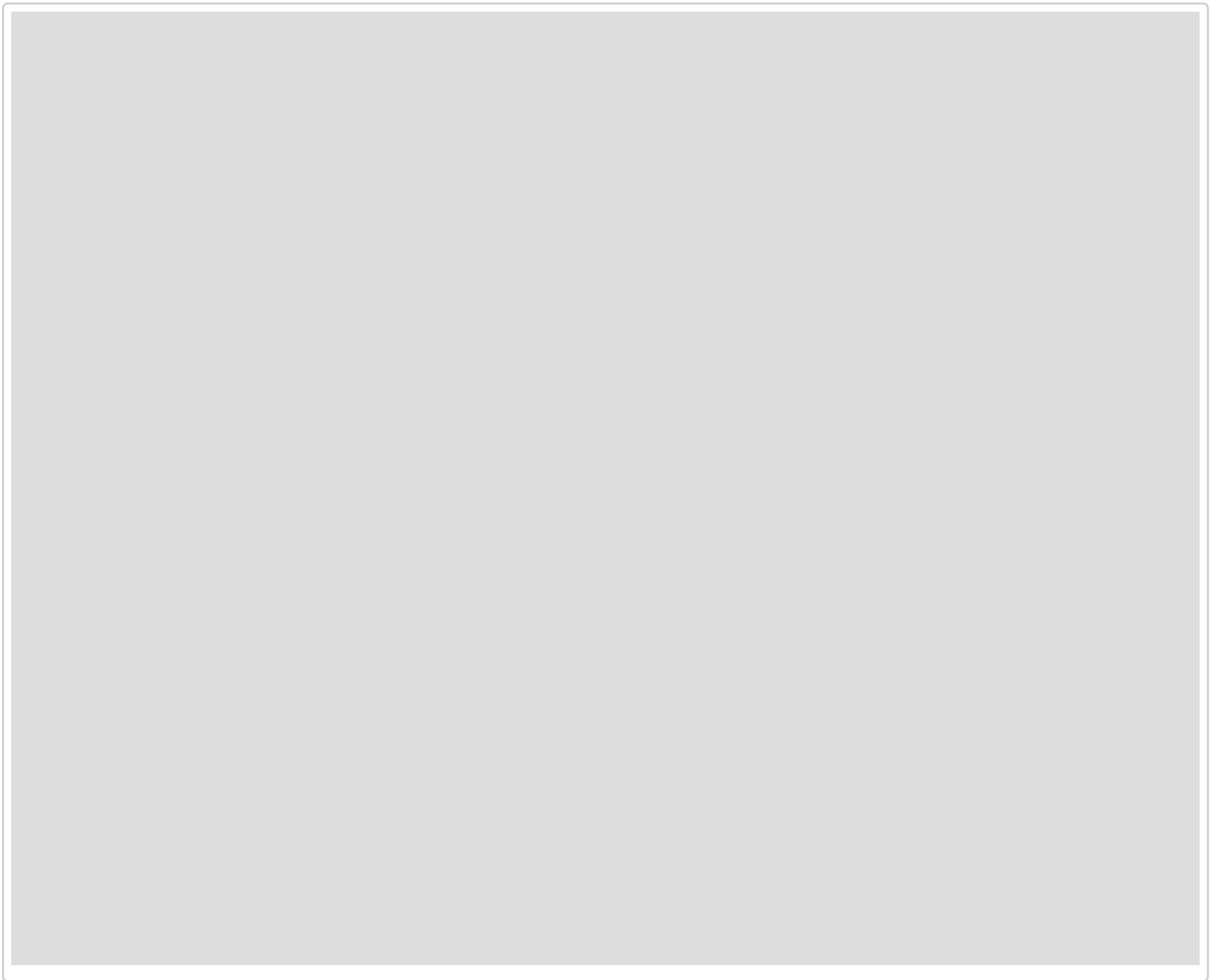
13.6.6 REST Object API

With Version 10.5 we introduce an optimized version of the [Read Object API \(see page 632\)](#) and a new [Write Object API \(see page 674\)](#). The usage of [Version 1 of the Read Object Api \(see page 595\)](#) is supported, but discouraged.

With this all typical CRUD (Create, Read, Update and Delete) operations can be performed for all repository based entities which support the object API.

13.6.6.1 Overview Presentation

This powerpoint presentation gives a rough overview of the Version 2 of the Object API. Please also see the partner portals for the demo videos which accompany this ppt.



13.6.6.2 Examples

Each Object API chapter has some examples directly on the page. For your convenience we also provide a Postman Collection (2.1 Version) with examples ready to use and try out the API. The collection also contains example calls to publish the api calls to the Apache MQ. For details on this see also [ObjectAPI Queue \(see page 945\)](#).

Download the Postman Collection (V2.1) here: `ObjectAPI.postman_collection.json`

13.6.6.3 OpenAPI Support

The Object API supports the OpenAPI standard 3.0.3 for all its operations. As the repository is very individual, a static api file would not be sufficient. Therefore a new Service API endpoint has been created with Product 360 10.5.0.01 to obtain the open api spec individually.

URL Pattern	<code>/object/{entity-identifier}/info</code>											
Method	GET											
Accept	<code>application/json</code> If multiple accept headers are defined, they will be evaluated by their order. The wildcard media type (*) will result in XML.											
Example	<code>/rest/V2.0/object/Article/info</code>											
Return Codes	<table><tr><th>Code</th><th>Reason</th></tr><tr><td><code>200</code> (OK)</td><td>Everything went fine</td></tr><tr><td><code>404</code> (Not Found)</td><td>Entity couldn't be found</td></tr><tr><td><code>403</code> (Forbidden)</td><td>User has no permissions for this entity</td></tr><tr><td><code>500</code></td><td>Internal server error, please check the server logs</td></tr></table>		Code	Reason	<code>200</code> (OK)	Everything went fine	<code>404</code> (Not Found)	Entity couldn't be found	<code>403</code> (Forbidden)	User has no permissions for this entity	<code>500</code>	Internal server error, please check the server logs
Code	Reason											
<code>200</code> (OK)	Everything went fine											
<code>404</code> (Not Found)	Entity couldn't be found											
<code>403</code> (Forbidden)	User has no permissions for this entity											
<code>500</code>	Internal server error, please check the server logs											

The response object corresponds to the OpenAPI standard 3.0.3 and can directly be used in 3rd party applications

Permissions

Entity Permissions

The OpenAPI document will not contain methods for which the user who requests the api document has no permission. E.g., the delete method is missing in case the user which is authenticated for the Service Api has no delete entity permission for the requested entity.

Field Permissions

In case the authenticated user has no write permission for a specific field, the field is returned as readOnly field. In case no read permission is available, the field will not be returned.

13.6.6.4 REST Object API Read V1

⚠ Discouraged

With introduction of [REST Object API Read V2](#) (see page 632) we discourage the usage of this version. Version 2 has improvements which optimize the performance of the api because the response is significantly smaller. Additionally to that, the structure is compatible with [REST Object API Write](#) (see page 674)

The object api provides a canonical rest api for entity items. Contrary to the ListAPI, the object api is optimized for single (multiple) item use and can return all data of the items in the full hierarchical form of them. It is not the correct api for mass data retrieval of a small set of columns. Please use the ListApi for such use cases.

- [Read Requests](#) (see page 595)
 - [Single Item](#) (see page 595)
 - [Multiple Items](#) (since 10.1.0.02) (see page 596)
 - [Query Parameters](#) (see page 598)
 - [Permissions](#) (see page 600)
 - [Object Permissions](#) (see page 600)
 - [Field Permissions](#) (see page 600)
 - [Qualification Permissions \(aka Qualified Field Permissions\)](#) (see page 601)
- [Result](#) (see page 601)
 - [Repository Entities and Fields](#) (see page 601)
 - [Names](#) (see page 601)
 - [Datatypes](#) (see page 601)
 - [Meta attributes](#) (see page 601)
 - [Data Element](#) (see page 602)
 - [Example Result](#) (see page 602)
 - [Characteristic Values](#) (see page 625)
 - [Additional Meta Attributes](#) (see page 626)
 - [Characteristic Records Example](#) (see page 626)
- [Examples](#) (see page 632)

Read Requests

Single Item

URL Pattern

`/object/{entity-identifier}/{entity_item}`

Method	GET										
Accept	application/json application/xml (default)										
Example	/rest/V1.0/object/Article/4711@1 /rest/V1.0/object/Article/'myItem'@'myCatalog'										
Return Codes	<table> <tr> <th>Code</th><th>Reason</th></tr> <tr> <td>200 (OK)</td><td>Everything went fine</td></tr> <tr> <td>404 (Not Found)</td><td>Object couldn't be found</td></tr> <tr> <td>403 (Forbidden)</td><td>User has no read permission for this object</td></tr> <tr> <td>500</td><td>Internal server error, please check the server logs</td></tr> </table>	Code	Reason	200 (OK)	Everything went fine	404 (Not Found)	Object couldn't be found	403 (Forbidden)	User has no read permission for this object	500	Internal server error, please check the server logs
Code	Reason										
200 (OK)	Everything went fine										
404 (Not Found)	Object couldn't be found										
403 (Forbidden)	User has no read permission for this object										
500	Internal server error, please check the server logs										

Multiple Items (since 10.1.0.02)

This endpoint provides a multi-item interface in which the client can provide multiple items which should be returned in one call. As query parameters do have a size limitation, this api is designed as POST. This api has restrictions on the number of items which can be called. In case the limit is exceeded a corresponding return code is provided.

Contrary to the single item GET request, a missing object permission or an unknown item id does not lead to an error return code; it will just not be returned.

URL Pattern	/object/{entity-identifier}/byItems
Method	POST

Headers													
	<table><tr><th>Header</th><th>Supported Values</th><th>Default</th></tr><tr><td>Accept</td><td>application/json application/xml</td><td>application/xml</td></tr><tr><td>Content-Type</td><td>application/x-www-form-urlencoded</td><td></td></tr></table>				Header	Supported Values	Default	Accept	application/json application/xml	application/xml	Content-Type	application/x-www-form-urlencoded	
	Header	Supported Values	Default										
	Accept	application/json application/xml	application/xml										
Content-Type	application/x-www-form-urlencoded												
Form Parameters	<table><tr><th>Parameter</th><th>Datatype</th><th>Description</th><th>Example</th></tr><tr><td>items</td><td>List of ENTITY_ITEM</td><td>String based entity item syntax. The form parameter can be provided multiple times and also as a comma separated list or even a mix of both</td><td>items=4711@1 items='myItem'@'myCatalog' items=4711@1, 'myItem'@'myCatalog'</td></tr></table>				Parameter	Datatype	Description	Example	items	List of ENTITY_ITEM	String based entity item syntax. The form parameter can be provided multiple times and also as a comma separated list or even a mix of both	items=4711@1 items='myItem'@'myCatalog' items=4711@1, 'myItem'@'myCatalog'	
	Parameter	Datatype	Description	Example									
items	List of ENTITY_ITEM	String based entity item syntax. The form parameter can be provided multiple times and also as a comma separated list or even a mix of both	items=4711@1 items='myItem'@'myCatalog' items=4711@1, 'myItem'@'myCatalog'										
Example	/rest/V1.0/object/Article/byItems												

Return Codes	Code	Reason
	200 (OK)	Everything went fine
	413 (Payload too large)	<p>The amount of entity item ids exceeds the configured limit (default: 100).</p> <p>The response contains the "p360-ObjectAPI-MaxItems" header which returns the limit.</p>
	500	Internal server error, please check the server logs

Query Parameters

Please note that the qualification filter applies to all requested sub-entities which have this qualification. So, for example, if you filter for language=English, the ArticleLang and ArticleAttributeValueLang entities will both only return the English values!

Parameter	Required	Default	Datatype	Parameter description	Example
<code>entityFilter</code>	no	none	String	<p>Comma separated list of entity identifiers which should be part of the result. If omitted, the full object with all data is returned. We recommend to provide a list of entities which are required in order to gain performance.</p> <p>Note: In case an entityFilter is provided, also the root entity is a filter value. So by only providing the root entity as filter value, only the fields of the root level are returned in the data.</p>	<p>Only return the root fields: <code>entityFilter=Article</code></p> <p>Only return the root fields, and the sales prices <code>entityFilter=Article,ArticleSalesPrice</code></p> <p>Only return the language specific data like Short and Long Description <code>entityFilter=ArticleLang</code></p>
<code>qualificationFilter</code>	no	none	String	<p>This parameter allows to restrict the output to certain qualifications. One example would be a filter so that only Euro and US-Dollar prices for the customer Informatica are returned.</p> <p>The filter string is a comma-separated list of qualification settings. A qualification setting is a <i>qualification name</i> followed by a comma-separated list of values which is put into parentheses. The qualification name is defined in the repository and is included in the Meta-API.</p>	<p>Example for prices in Euro and US-Dollar and customer Informatica: <code>qualificationFilter=currency(EUR,USD),customer(Informatica)</code></p>

Parameter	Required	Default	Datatype	Parameter description	Example
revision	no	root	ENTITY_ITEM	The revision for which the data should be retrieved	<pre>revision='root'</pre> <pre>revision=1</pre> <pre>revision='myRevisionIdentifier'</pre> <pre>revision=4711</pre> (where 4711 is the internal id of the revision).
includeLabels (available with 10.1.0.02)	no	false	boolean	If set to true, the returned document will contain _label elements for all fields or qualifications which have an enumeration. The label will be returned in the locale of the request like this: <pre>"orderUnit" : { "_current" : { "_key" : { "_entityId" : 3100, "_internalId" : "932", "_externalId" : "'C62'" }, "_code" : "C62", "_label" : "piece" } }</pre>	<pre>includeLabels=true</pre> <pre>includeLabels=false</pre>

Permissions

Object Permissions

The user needs to have the READ object permission for this API. In case he doesn't, the api returns HTTP 403 (forbidden) in case of the single item call with GET, and skip this item in case of the multiple items call.

Field Permissions

The user needs to have the READ field permissions for a field. If he doesn't, the field will not be part of the data element, but no error is returned.

Qualification Permissions (aka Qualified Field Permissions)

The user needs to have the READ permission for a qualification, e.g. for Language = English. If he doesn't, sub-entities which are qualified for this will not be part of the data element, but no error is returned.

Result

Repository Entities and Fields

Names

A new [repository property "shortIdentifier"](#) (see [page 79](#)) has been added in the custom section which is used to define the entity and field names for the json structure. The short identifier is unique within its parent entity only!

The standard repository already contains a short identifier for all fields and enabled qualifications.

Datatypes

- ENTITY_ITEM objects always contain the `_entityId`, `_internalId` and `_externalId` attributes
- MIME_VALUE objects always contain the relative `_filePath`, the `_label` and the `_mimeType`
- timestamp: ISO 8601, e.g.: `2012-04-23T18:25:43.511Z`
- date: ISO 8601, e.g.: `2019-07-04`
- numbers: standard JSON
- "no value" is either provided as null, or the attribute is not in the document.
- Empty string is returned as null, which is omitted when possible!
- Enumeration fields
 - `_key` : always holds the unique key of the enumeration entry
In case the enumeration field is of datatype ENTITY_ITEM, then the key will be provided with the ENTITY_ITEM syntax (see above)
 - `_code` : the external code of the enumeration entry, if the enum entry has one, and it differs from the key!
 - `_label` : the locale dependent label of the enumeration entry. Only if `includeLabels` is set to true. The locale of the http request is used for this.

Meta attributes

All meta attributes are prefixed with an underscore.

- `_entity` = the root entity identifier
- `_entityItem` = the entity item in the ENTITY_ITEM syntax

- `_revision` = the revision of the data as `ENTITY_ITEM`
- `_data` = The actual entity item's data
- `_current` = the current value of the field
As the structure of the document is similar to the [EntityItemChange Document](#) (see page 190) which is used for audit trail the `_current` attribute for the current value is required.
- `_qualification` = the qualification of the record. **The combination of the qualification values in the qualification object define this record as unique within its parent.**

Data Element

In order to be part of the data element of the object api, a sub-entity or field must fulfill multiple criteria:

- The entity or field must be `active` in the Repository
- The entity or field must have `supportsServiceApi = true` in the Repository
- The entity or field must have `supportsAuditTrail = true` in the Repository
- The entity or field must have a `shortIdentifier` in the Repository
- The user must have `READ` permission for the `Entity` or `Field`
- The user must have `READ` permission for the `qualification` of the sub-entity
- The field must not be a `password Field` in the Repository
- `Qualifications` must be `editable` in the Repository

Entities, Fields or Qualifications which do not comply to any of these points will be omitted in the data element. No exception is thrown.

Example Result

```
{
  "_entity": "Article",
  "_entityItem": {
    "_internalId": "13989@1120",
    "_entityId": 1000,
    "_externalId": "\u0027supplierAID-840\u0027@\u0027NEW_CATALOG\u0027"
  },
  "_revision": {
    "_internalId": "1",
    "_entityId": 5600,
    "_externalId": "\u0027root\u0027"
  },
  "_container": {
    "_internalId": "1120",
    "_entityId": 7000,
    "_externalId": "\u0027NEW_CATALOG\u0027"
  },
  "_data": {
    "article": {
      "identifier": {
```

```

    "_current": "supplierAID-840"
  },
  "manufacturerAID": {
    "_current": "L0000000000839"
  },
  "manufacturerName": {
    "_current": "Rolex"
  },
  "deliveryTime": {
    "_current": 3.8900
  },
  "orderUnit": {
    "_current": {
      "_key": {
        "_entityId": 3100,
        "_internalId": "145",
        "_externalId": "\u0027AMH\u0027"
      },
      "_code": "AMH"
    }
  },
  "contentUnit": {
    "_current": {
      "_key": {
        "_entityId": 3100,
        "_internalId": "310",
        "_externalId": "\u0027GLL\u0027"
      },
      "_code": "GLL"
    }
  },
  "noCUPerOU": {
    "_current": 842.3100
  },
  "priceQuantity": {
    "_current": 219.5200
  },
  "quantityMin": {
    "_current": 6.0000
  },
  "quantityInterval": {
    "_current": 1.0000
  },
  "mainSupplier": {
    "_current": {
      "_key": {
        "_entityId": 2800,
        "_internalId": "3",
        "_externalId": "\u0027Heiler Product Manager\u0027"
      },
      "_code": "Heiler Product Manager"
    }
  }
}

```

```

    },
    "currentStatus": {
      "_current": {
        "_key": 100,
        "_code": "NEW"
      }
    },
    "kitParent": {
      "_current": false
    },
    "kitComponent": {
      "_current": false
    },
    "soldOnlyInKits": {
      "_current": false
    },
    "lang": [
      {
        "_qualification": {
          "language": {
            "_key": 9,
            "_code": "eng"
          }
        }
      },
      {
        "descriptionShort": {
          "_current": "the little value he put on his own good qualities.
Elizabeth was pleased to find that he had not betrayed the interference of his"
        },
        "descriptionLong": {
          "_current": "practice. I have told Miss Bennet several times, that
she will never play really well unless she practises more; and though Mrs.
Collins has no instrument, she is very welcome, as I have often told her, to
come to Rosings every day, and play on the pianoforte in Mrs. Jenkinson's room.
She would be in nobody's way, you know, in that part of the house." Mr. Darcy
looked a little ashamed of his aunt's ill-breeding, and made no answer. When
coffee was over, Colonel Fitzwilliam reminded Elizabeth of having promised to
play to him; and she sat down directly to the i"
        }
      },
      {
        "keywords": {
          "_current": [
            "MERCHANTIBILITY"
          ]
        }
      }
    ],
    {
      "_qualification": {
        "language": {
          "_key": 1046,
          "_code": "por"
        }
      },
      {
        "descriptionShort": {

```

```

        "_current": "daughters. Mr. Collins was punctual to his time, and
was receive"
    },
    "descriptionLong": {
        "_current": "against herself; and his disappointed feelings became
the object of compassion. His attachment excited gratitude, his general
character respect; but she could not approve him; nor could she for a moment
repent her refusal, or feel the slightest inclination ever to see him again. In
her own past behaviour, there was a constant source of vexation and regret; and
in the unhappy defects of her family, a subject of yet heavier chagrin. They
were hopeless of remedy. Her father, contented with laughing at them, would
never exert himself to restrain the wild giddiness of his youngest daughters;
and her mother, with manners so far fr"
    },
    "keywords": {
        "_current": [
            "busy",
            "_particularly_",
            "nowhere"
        ]
    }
},
{
    "_qualification": {
        "language": {
            "_key": 7,
            "_code": "deu"
        }
    },
    "descriptionShort": {
        "_current": "has now living, better than any other person."
Elizabeth was at no loss to understand from whence this defe"
    },
    "descriptionLong": {
        "_current": "or expense to the user, provide a copy, a means of
exporting a copy, or a means of obtaining a copy upon request, of the work in
its original "Plain Vanilla ASCII" or other form. Any alternate format must
include the full Project Gutenberg-tm License as specified in paragraph 1.E.1.
1.E.7. Do not charge a fee for access to, viewing, displaying, performing,
copying or distributing any Project Gutenberg-tm works unless you comply with p"
    }
}
],
"pricePurchase": [
{
    "_qualification": {
        "supplier": {
            "_key": {
                "_entityId": 2800,
                "_internalId": "3",
                "_externalId": "\u0027Heiler Product Manager\u0027"
            }
        }
    },

```



```

    "_code": "Heiler Product Manager"
  },
  "type": {
    "_key": 1,
    "_code": "net_list"
  },
  "currency": {
    "_key": "USD"
  },
  "territory": {
    "_key": "DE"
  }
},
"validFrom": {
  "_current": "1899-12-30"
},
"validTo": {
  "_current": "9999-12-31"
},
"value": [
  {
    "_qualification": {
      "lowerBound": 1.0000
    },
    "amount": {
      "_current": 872327.120000
    },
    "factor": {
      "_current": 1.0000
    }
  }
]
},
{
  "_qualification": {
    "supplier": {
      "_key": {
        "_entityId": 2800,
        "_internalId": "3",
        "_externalId": "\u0027Heiler Product Manager\u0027"
      },
      "_code": "Heiler Product Manager"
    },
    "type": {
      "_key": 2,
      "_code": "gross_list"
    },
    "currency": {
      "_key": "CHF"
    },
    "territory": {
      "_key": "DE"
    }
  }
}

```

```

    }
  },
  "validFrom": {
    "_current": "1899-12-30"
  },
  "validTo": {
    "_current": "9999-12-31"
  },
  "value": [
    {
      "_qualification": {
        "lowerBound": 3.0000
      },
      "amount": {
        "_current": 239011.140000
      },
      "factor": {
        "_current": 1.0000
      }
    }
  ]
},
],
"priceSales": [
  {
    "_qualification": {
      "customer": {
        "_key": {
          "_entityId": 2800,
          "_internalId": "1",
          "_externalId": "\u0027Public\u0027"
        }
      },
      "type": {
        "_key": 3,
        "_code": "net_customer"
      },
      "currency": {
        "_key": "CHF"
      },
      "territory": {
        "_key": "DE"
      }
    },
    "validFrom": {
      "_current": "1899-12-30"
    },
    "validTo": {
      "_current": "9999-12-31"
    },
    "value": [
      {

```

```

        "_qualification": {
            "lowerBound": 2.0000
        },
        "amount": {
            "_current": 60216.410000
        },
        "factor": {
            "_current": 1.0000
        }
    }
]
},
"logistic": [
    {
        "_qualification": {
            "territory": {
                "_key": "CH"
            }
        },
        "packageUnit": {
            "_current": {
                "_key": {
                    "_entityId": 3100,
                    "_internalId": "747",
                    "_externalId": "\u0027NEW\u0027"
                },
                "_code": "NEW"
            }
        },
        "packageWeight": {
            "_current": 36.5000
        },
        "originCountry": {
            "_current": {
                "_key": "BB"
            }
        },
        "grossWeight": {
            "_current": 43.9700
        },
        "grossWeightUnit": {
            "_current": {
                "_key": {
                    "_entityId": 3100,
                    "_internalId": "723",
                    "_externalId": "\u0027MON\u0027"
                },
                "_code": "MON"
            }
        }
    }
]
}

```

```

],
"logisticExtension": [
  {
    "_qualification": {
      "party": {
        "_key": {
          "_entityId": 2800,
          "_internalId": "3",
          "_externalId": "\u0027Heiler Product Manager\u0027"
        },
        "_code": "Heiler Product Manager"
      },
      "packagingUnit": {
        "_key": {
          "_entityId": 3100,
          "_internalId": "380",
          "_externalId": "\u00275\u0027"
        },
        "_code": "5"
      },
      "language": {
        "_key": 10,
        "_code": "esl"
      }
    },
    "code128": {
      "_current": "62243677464153991935"
    },
    "code39": {
      "_current": "25028067122421819836"
    },
    "length": {
      "_current": 52873820.0761
    },
    "lengthUnit": {
      "_current": {
        "_key": {
          "_entityId": 3100,
          "_internalId": "936",
          "_externalId": "\u0027AR\u0027"
        },
        "_code": "AR"
      }
    },
    "height": {
      "_current": 6794560.1841
    },
    "heightUnit": {
      "_current": {
        "_key": {
          "_entityId": 3100,
          "_internalId": "7053",

```

```

        "_externalId": "\u0027X_PPC\u0027"
      },
      "_code": "X_PPC"
    }
  }
},
"referencedItem": [
  {
    "_qualification": {
      "type": {
        "_key": 8,
        "_code": "diff_orderunit"
      },
      "referencedIdentifier": "supplierAID-648",
      "referencedCatalogIdentifier": "NEW_CATALOG"
    },
    "referencedItem": {
      "_current": {
        "_entityId": 1000,
        "_internalId": "14452@1120",
        "_externalId":
"\u0027supplierAID-648\u0027@\u0027NEW_CATALOG\u0027"
      }
    },
    "quantity": {
      "_current": 1
    }
  }
],
"structureMap": [
  {
    "_qualification": {
      "structure": {
        "_key": 15,
        "_code": "ECLASS-6.1"
      }
    },
    "structureGroups": {
      "_current": [
        {
          "_entityId": 3000,
          "_internalId": "23576@15",
          "_externalId": "\u0027AKL316004\u0027@\u0027ECLASS-6.1\u0027"
        }
      ]
    }
  }
],
"structureGroupMap": [
  {
    "_qualification": {

```

```

    "structure": {
      "_key": {
        "_entityId": 2300,
        "_internalId": "15",
        "_externalId": "\u0027ECLASS-6.1\u0027"
      },
      "_code": "ECLASS-6.1"
    },
    "structureGroup": {
      "_entityId": 3000,
      "_internalId": "23576@15",
      "_externalId": "\u0027AKL316004\u0027@\u0027ECLASS-6.1\u0027"
    }
  },
  "sequence": {
    "_current": 2147483647
  }
},
"mediaAsset": [
  {
    "_qualification": {
      "type": {
        "_key": "logo"
      }
    },
    "name": {
      "_current": "be the meaning of it? It"
    },
    "category": {
      "_current": "nonproprietary"
    },
    "mediaAsset": {
      "_current": {
        "_entityId": 2400,
        "_internalId": "250",
        "_externalId": "\u0027MediaAsset_1596466573432152\u0027"
      }
    }
  },
  "document": [
    {
      "_qualification": {
        "quality": {
          "_key": "highres"
        },
        "language": {
          "_key": 7,
          "_code": "deu"
        }
      },
      "identifier": {
        "_current": "h1r-system/nhvjsixjcjucktt.jpg"
      }
    }
  ]
}

```

```

    },
    "imageIdentifier": {
      "_current": "h\lr-system/nhvjsixjcjucktt.jpg"
    },
    "order": {
      "_current": 1
    }
  }
]
},
{
  "_qualification": {
    "type": {
      "_key": "thumbnail"
    }
  },
  "name": {
    "_current": "from Jane. “I do not know whe"
  },
  "category": {
    "_current": "counterbalance"
  },
  "mediaAsset": {
    "_current": {
      "_entityId": 2400,
      "_internalId": "255",
      "_externalId": "\u0027MediaAsset_1596466573432157\u0027"
    }
  },
  "document": [
    {
      "_qualification": {
        "quality": {
          "_key": "highres"
        },
        "language": {
          "_key": 56,
          "_code": "kor"
        }
      },
      "identifier": {
        "_current": "h\lr-system/ajomgjomrxyjews.jpg"
      },
      "imageIdentifier": {
        "_current": "h\lr-system/ajomgjomrxyjews.jpg"
      },
      "order": {
        "_current": 1
      }
    }
  ]
}
]
}

```

```

],
"log": [
  {
    "_qualification": {
      "channel": {
        "_key": "100_classifier.id"
      }
    },
    "creationDate": {
      "_current": "2020-10-07T13:56:25.990Z"
    },
    "creationUser": {
      "_current": {
        "_key": {
          "_entityId": 2600,
          "_internalId": "1",
          "_externalId": "\u0027Administrator\u0027"
        }
      }
    },
    "deletionDate": {
      "_current": "9999-12-31T00:00:00.000Z"
    }
  },
  {
    "_qualification": {
      "channel": {
        "_key": "HPM"
      }
    },
    "creationDate": {
      "_current": "2020-10-07T13:56:25.990Z"
    },
    "creationUser": {
      "_current": {
        "_key": {
          "_entityId": 2600,
          "_internalId": "1",
          "_externalId": "\u0027Administrator\u0027"
        }
      }
    },
    "deletionDate": {
      "_current": "9999-12-31T00:00:00.000Z"
    }
  },
  {
    "_qualification": {
      "channel": {
        "_key": "101_classifier.id"
      }
    },
  },

```



```

    "creationDate": {
      "_current": "2020-10-07T13:56:25.990Z"
    },
    "creationUser": {
      "_current": {
        "_key": {
          "_entityId": 2600,
          "_internalId": "1",
          "_externalId": "\u0027Administrator\u0027"
        }
      }
    },
    "deletionDate": {
      "_current": "9999-12-31T00:00:00.000Z"
    }
  ],
  "ownLog": [
    {
      "modificationDate": {
        "_current": "2020-10-07T13:56:26.010Z"
      }
    }
  ],
  "nutrient": [
    {
      "_qualification": {
        "targetMarket": {
          "_key": "CA"
        },
        "preparationState": {
          "_key": "READY_TO_EAT"
        }
      },
      "lang": [
        {
          "_qualification": {
            "language": {
              "_key": 12,
              "_code": "fra"
            }
          },
          "servingSizeDescription": {
            "_current": "of general complaisance, and in all that he said
she heard an accent so removed from _hauteur_ or disdain"
          },
          "dailyValueIntakeReference": {
            "_current": "said Elizabeth, struck with other ideas. "She looks
sickly and cross. Yes, she will do for"
          }
        },
        {

```

```

    "_qualification": {
      "language": {
        "_key": 16,
        "_code": "ita"
      }
    },
    "servingSizeDescription": {
      "_current": "her mother. "And then when you go away, you may
leave one or two of my sisters behind you; and I dare say I shall get husbands
for them before the winter is over." "I thank you for my share of the favour,"
said Elizabeth; "but I do not particularly like your way of getting husbands."
Their visitors were not to remain above ten days with them. Mr. Wickham had
received his commission before he"
    },
    "dailyValueIntakeReference": {
      "_current": "duty of a young man who has been so fortunate as I
have been in early preferment; and I trust I am resigned. Perhaps not the less
so from feeling a doubt of my positive happiness had my fair cousin honoured me
with her hand; for I have often observed that resignation is n"
    }
  ]
}
],
"_characteristicRecords": [
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "12424",
        "_externalId": "\u0027isHomogenised\u0027"
      },
      "_code": "isHomogenised"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "LOOKUP",
  "order": {
    "_current": 1
  },
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": -1,
          "_code": "zxx"
        }
      },
      "values": {
        "_current": [

```

```

        {
            "_entityId": 7300,
            "_internalId": "1209@121",
            "_externalId":
"\u0027FALSE\u0027@\u0027NonBinaryLogicCodes\u0027"
        }
    ]
}
]
},
{
    "_qualification": {
        "characteristic": {
            "_key": {
                "_entityId": 8000,
                "_internalId": "12441",
                "_externalId": "\u0027nutrientFormatTypeCodeReference\u0027"
            },
            "_code": "nutrientFormatTypeCodeReference"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "root"
    },
    "_datatype": "LOOKUP",
    "order": {
        "_current": 1
    },
    "_recordLang": [
        {
            "_qualification": {
                "language": {
                    "_key": -1,
                    "_code": "zxx"
                }
            },
            "values": {
                "_current": [
                    {
                        "_entityId": 7300,
                        "_internalId": "3494@151",
                        "_externalId":
"\u0027AB\u0027@\u0027nutrientFormatTypeCodeReference\u0027"
                    }
                ]
            }
        }
    ]
},
{
    "_qualification": {
        "characteristic": {

```

```

    "_key": {
      "_entityId": 8000,
      "_internalId": "12377",
      "_externalId": "\u0027juiceContentPercentage\u0027"
    },
    "_code": "juiceContentPercentage"
  },
  "recordKey": "0000.0000.RK",
  "parentRecordKey": "root"
},
"_datatype": "DECIMAL",
"_formatPattern": "###.##",
"order": {
  "_current": 1
},
"_recordLang": [
  {
    "_qualification": {
      "language": {
        "_key": -1,
        "_code": "zxx"
      }
    },
    "values": {
      "_current": [
        3169.03
      ]
    }
  }
]
},
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "12339",
        "_externalId": "\u0027fatPercentageInDryMatter\u0027"
      },
      "_code": "fatPercentageInDryMatter"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "DECIMAL",
  "_formatPattern": "##.###",
  "order": {
    "_current": 1
  },
  "_recordLang": [
    {
      "_qualification": {

```

```

        "language": {
            "_key": -1,
            "_code": "zxx"
        }
    },
    "values": {
        "_current": [
            4176.55
        ]
    }
}
]
},
{
    "_qualification": {
        "characteristic": {
            "_key": {
                "_entityId": 8000,
                "_internalId": "12340",
                "_externalId": "\u0027isRindEdible\u0027"
            },
            "_code": "isRindEdible"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "root"
    },
    "_datatype": "LOOKUP",
    "order": {
        "_current": 1
    },
    "_recordLang": [
        {
            "_qualification": {
                "language": {
                    "_key": -1,
                    "_code": "zxx"
                }
            },
            "values": {
                "_current": [
                    {
                        "_entityId": 7300,
                        "_internalId": "1296@121",
                        "_externalId":
"\u0027NOT_APPLICABLE\u0027@\u0027NonBinaryLogicCodes\u0027"
                    }
                ]
            }
        }
    ]
},
{

```

```

    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,
          "_internalId": "12337",
          "_externalId": "\u0027cheeseMaturationPeriodDescription\u0027"
        },
        "_code": "cheeseMaturationPeriodDescription"
      },
      "recordKey": "0000.0000.RK",
      "parentRecordKey": "root"
    },
    "_datatype": "TEXT",
    "order": {
      "_current": 1
    },
    "_recordLang": [
      {
        "_qualification": {
          "language": {
            "_key": 17,
            "_code": "jpn"
          }
        },
        "values": {
          "_current": [
            "of his confidence in his friend. How grievous then was the"
          ]
        }
      }
    ]
  },
  {
    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,
          "_internalId": "12338",
          "_externalId":
"\u0027cheeseMaturationProcessContainerTypeCode\u0027"
        },
        "_code": "cheeseMaturationProcessContainerTypeCode"
      },
      "recordKey": "0000.0000.RK",
      "parentRecordKey": "root"
    },
    "_datatype": "LOOKUP",
    "order": {
      "_current": 1
    },
    "_recordLang": [
      {

```

```

    "_qualification": {
      "language": {
        "_key": -1,
        "_code": "zxx"
      }
    },
    "values": {
      "_current": [
        {
          "_entityId": 7300,
          "_internalId": "2062@123",
          "_externalId":
"\u0027BRINE_SOLUTION\u0027@\u0027cheeseMaturationProcessContainerTypeCode\u0027
"
        }
      ]
    }
  }
},
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "12375",
        "_externalId": "\u0027ingredientOfConcernCode\u0027"
      },
      "_code": "ingredientOfConcernCode"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "LOOKUP",
  "order": {
    "_current": 1
  },
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": -1,
          "_code": "zxx"
        }
      },
      "values": {
        "_current": [
          {
            "_entityId": 7300,
            "_internalId": "2720@136",
            "_externalId":
"\u0027RAW_MILK\u0027@\u0027ingredientOfConcernCode\u0027"

```

```

    }
  ]
}
]
},
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "12376",
        "_externalId": "\u0027ingredientStatement\u0027"
      },
      "_code": "ingredientStatement"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "TEXT",
  "order": {
    "_current": 1
  },
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": 11,
          "_code": "fin"
        }
      },
      "values": {
        "_current": [
          "anything about it, they found at last, on examining th"
        ]
      }
    }
  ]
},
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "12341",
        "_externalId": "\u0027surfaceOfCheeseAtEndOfRipeningCode\u0027"
      },
      "_code": "surfaceOfCheeseAtEndOfRipeningCode"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
},

```



```

    "_datatype": "LOOKUP",
    "order": {
      "_current": 1
    },
    "_recordLang": [
      {
        "_qualification": {
          "language": {
            "_key": -1,
            "_code": "zxx"
          }
        },
        "values": {
          "_current": [
            {
              "_entityId": 7300,
              "_internalId": "1913@122",
              "_externalId":
"\u0027NO_RIND\u0027@\u0027surfaceOfCheeseAtEndOfRipeningCode\u0027"
            }
          ]
        }
      }
    ],
    {
      "_qualification": {
        "characteristic": {
          "_key": {
            "_entityId": 8000,
            "_internalId": "12342",
            "_externalId": "\u0027dietTypeDescription\u0027"
          },
          "_code": "dietTypeDescription"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "root"
      },
      "_datatype": "TEXT",
      "order": {
        "_current": 1
      },
      "_recordLang": [
        {
          "_qualification": {
            "language": {
              "_key": 4,
              "_code": "chi"
            }
          },
          "values": {
            "_current": [

```

```

        "and, for a few moments, she flattered herself tha"
    ]
}
}
]
},
{
    "_qualification": {
        "characteristic": {
            "_key": {
                "_entityId": 8000,
                "_internalId": "12555",
                "_externalId": "\u0027doPackagingMaterialContainLatex\u0027"
            },
            "_code": "doPackagingMaterialContainLatex"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "root"
    },
    "_datatype": "LOOKUP",
    "order": {
        "_current": 1
    },
    "_recordLang": [
        {
            "_qualification": {
                "language": {
                    "_key": -1,
                    "_code": "zxx"
                }
            },
            "values": {
                "_current": [
                    {
                        "_entityId": 7300,
                        "_internalId": "1294@121",
                        "_externalId":
"\u0027TRUE\u0027@\u0027NonBinaryLogicCodes\u0027"
                    }
                ]
            }
        }
    ],
    "_qualification": {
        "characteristic": {
            "_key": {
                "_entityId": 8000,
                "_internalId": "12556",
                "_externalId": "\u0027numberOfPackagesForSerPiecesGTIN\u0027"
            },

```

```

        "_code": "numberOfPackagesForSerPiecesGTIN"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
},
"_datatype": "INTEGER",
"order": {
    "_current": 1
},
"_recordLang": [
    {
        "_qualification": {
            "language": {
                "_key": -1,
                "_code": "zxx"
            }
        },
        "values": {
            "_current": [
                4410
            ]
        }
    }
]
},
{
    "_qualification": {
        "characteristic": {
            "_key": {
                "_entityId": 8000,
                "_internalId": "12332",
                "_externalId": "\u0027fatInMilkContent\u0027"
            },
            "_code": "fatInMilkContent"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "root"
    },
    "_datatype": "DECIMAL",
    "_formatPattern": "##.###",
    "order": {
        "_current": 1
    },
    "_recordLang": [
        {
            "_qualification": {
                "language": {
                    "_key": -1,
                    "_code": "zxx"
                }
            },
            "values": {

```

```

        "_current": [
            58.92
        ]
    }
}
],
{
    "_qualification": {
        "characteristic": {
            "_key": {
                "_entityId": 8000,
                "_internalId": "12333",
                "_externalId": "\u0027rennetTypeCode\u0027"
            },
            "_code": "rennetTypeCode"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "root"
    },
    "_datatype": "LOOKUP",
    "order": {
        "_current": 1
    },
    "_recordLang": [
        {
            "_qualification": {
                "language": {
                    "_key": -1,
                    "_code": "zxx"
                }
            },
            "values": {
                "_current": [
                    {
                        "_entityId": 7300,
                        "_internalId": "1931@126",
                        "_externalId":
"\u0027VEGETABLE_RENNET\u0027@\u0027rennetTypeCode\u0027"
                    }
                ]
            }
        }
    ]
}
]
}
}
}

```

Characteristic Values

Characteristic values are rendered in a specialized structure in order to honor their hierarchical nature. This structure is not identical to the repository as it is fully hierarchical. This is the reason for extra meta attributes which help to not get in conflict with repository based data.

Additional Meta Attributes

- `_characteristicRecords`
Top level element for all characteristic records
- `_recordLang`
Language specific record values. In case the characteristic is not language specific, the qualification for "not language specific" is returned (-1 or xxz)
In case the value of the record is a lookup value and the includeLabels parameter has been set, an additional `_label` element is provided.
- `_children`
children of the current characteristic record
- `_datatype`
The characteristic data type - to be able to interpret the values even in case the characteristic is no longer in the system
- `_formatPattern`
The format pattern of the characteristic - to be able to format the values even in case the characteristic is no longer in the system

Characteristic Records Example

```
{
  "_entity": "Article",
  "_entityItem": {
    "_internalId": "13989@1120",
    "_entityId": 1000,
    "_externalId": "\u0027supplierAID-840\u0027@\u0027NEW_CATALOG\u0027"
  },
  "_revision": {
    "_internalId": "1",
    "_entityId": 5600,
    "_externalId": "\u0027root\u0027"
  },
  "_container": {
    "_internalId": "1120",
    "_entityId": 7000,
    "_externalId": "\u0027NEW_CATALOG\u0027"
  },
  "_data": {
    "article": {
      "identifier": {
        "_current": "supplierAID-841"
      },
      "_characteristicRecords": [
        {
          "_qualification": {
            "characteristic": {
```

```

    "_key": {
      "_entityId": 8000,
      "_internalId": "7",
      "_externalId": "\u0027AnimalIngredient\u0027"
    },
    "_code": "AnimalIngredient"
  },
  "recordKey": "0000.0000.RK",
  "parentRecordKey": "root"
},
"_dataType": "LOOKUP",
"order": {
  "_current": -32767
},
"_recordLang": [
  {
    "_qualification": {
      "language": {
        "_key": -1,
        "_code": "zxx"
      }
    },
    "values": {
      "_current": [
        {
          "_entityId": 7300,
          "_internalId": "23@5",
          "_externalId":
"\u0027Down\u0027@\u0027AnimalIngredient\u0027"
        }
      ]
    }
  }
],
"_children": [
  {
    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,
          "_internalId": "8",
          "_externalId": "\u0027CertDown\u0027"
        },
        "_code": "CertDown"
      },
      "recordKey": "0000.0000.RK",
      "parentRecordKey": "0000.0000.RK"
    },
    "_dataType": "LOOKUP",
    "order": {
      "_current": -32766
    }
  },

```

```

"_recordLang": [
  {
    "_qualification": {
      "language": {
        "_key": -1,
        "_code": "zxx"
      }
    },
    "values": {
      "_current": [
        {
          "_entityId": 7300,
          "_internalId": "25@6",
          "_externalId": "\u00270ther\u0027@\u0027CertDown\u0027"
        }
      ]
    }
  }
],
"_children": [
  {
    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,
          "_internalId": "11",
          "_externalId": "\u0027CertDownExpDate\u0027"
        },
        "_code": "CertDownExpDate"
      },
      "recordKey": "0000.0000.RK",
      "parentRecordKey": "0000.0000.RK"
    },
    "_dataType": "DATE",
    "order": {
      "_current": -32765
    },
    "_recordLang": [
      {
        "_qualification": {
          "language": {
            "_key": -1,
            "_code": "zxx"
          }
        },
        "values": {
          "_current": [
            "1977-05-28"
          ]
        }
      }
    ]
  }
]

```

```

    }
  ]
},
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "8",
        "_externalId": "\u0027CertDown\u0027"
      },
      "_code": "CertDown"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "0000.0000.RK"
  },
  "_dataType": "LOOKUP",
  "order": {
    "_current": -32766
  },
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": -1,
          "_code": "zxx"
        }
      },
      "values": {
        "_current": [
          {
            "_entityId": 7300,
            "_internalId": "25@6",
            "_externalId": "\u00270ther\u0027@\u0027CertDown\u0027"
          }
        ]
      }
    }
  ],
  "_children": [
    {
      "_qualification": {
        "characteristic": {
          "_key": {
            "_entityId": 8000,
            "_internalId": "11",
            "_externalId": "\u0027CertDownExpDate\u0027"
          },
          "_code": "CertDownExpDate"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "0000.0000.RK"
      }
    }
  ]
}

```



```

    },
    "_dataType": "DATE",
    "order": {
        "_current": -32765
    },
    "_recordLang": [
        {
            "_qualification": {
                "language": {
                    "_key": -1,
                    "_code": "zxx"
                }
            },
            "values": {
                "_current": [
                    "1977-05-28"
                ]
            }
        }
    ]
}
],
"_children": [
    {
        "_qualification": {
            "characteristic": {
                "_key": {
                    "_entityId": 8000,
                    "_internalId": "8",
                    "_externalId": "\u0027CertDown\u0027"
                },
                "_code": "CertDown"
            },
            "recordKey": "0000.0000.RK",
            "parentRecordKey": "0000.0000.RK"
        },
        "_dataType": "LOOKUP",
        "order": {
            "_current": -32766
        },
        "_recordLang": [
            {
                "_qualification": {
                    "language": {
                        "_key": -1,
                        "_code": "zxx"
                    }
                }
            }
        ],
    },

```

```

    "values": {
      "_current": [
        {
          "_entityId": 7300,
          "_internalId": "25@6",
          "_externalId": "\u00270ther\u0027@\u0027CertDown\u0027"
        }
      ]
    }
  ],
  "_children": [
    {
      "_qualification": {
        "characteristic": {
          "_key": {
            "_entityId": 8000,
            "_internalId": "11",
            "_externalId": "\u0027CertDownExpDate\u0027"
          },
          "_code": "CertDownExpDate"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "0000.0000.RK"
      },
      "_dataType": "DATE",
      "order": {
        "_current": -32765
      },
      "_recordLang": [
        {
          "_qualification": {
            "language": {
              "_key": -1,
              "_code": "zxx"
            }
          },
          "values": {
            "_current": [
              "1977-05-28"
            ]
          }
        }
      ]
    }
  ]
}

```

Examples

Please see the attached postman example collection for you convenience: ObjectAPI.postman_collection.json

Get Item by ID

```
curl --location --request GET 'http://localhost:1512/rest/V1.0/object/Article/14260@1120' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic cmVzdDpoZWlsZXI='
```

Get Item by Identifier As XML

```
curl --location --request GET 'http://localhost:1512/rest/V1.0/object/Article/'\''supplierAID-42'\''@'\''MySupplierCatalog'\'' \
--header 'Content-Type: application/xml' \
--header 'Authorization: Basic cmVzdDpoZWlsZXI='
```

Get Filtered Item (entity and qualification filter)

```
curl --location --request GET 'http://localhost:1512/rest/V1.0/object/Article/'\''supplierAID-42'\''@'\''MySupplierCatalog'\''?qualificationFilter=language(eng)&entityFilter=ArticleLang' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic cmVzdDpoZWlsZXI='
```

13.6.6.5 REST Object API Read V2



Version 2 of the Object Read API introduces a new parameter "includeIds" and a streamlined response payload which no longer contains the extra "_current" element. Besides that, the response payload's _data element can directly be used for the [Write API](#) (see page 674). If a field or entity is supported by the Object API V2 has now been bound to the newly introduced "supportsObjectApi" flag in the repository. A better control over the fields and entities which are supported is possible with this.

The object api provides a canonical rest api for entity items. Contrary to the ListAPI, the object api is optimized for single (multiple) item use and can return all data of the items in the full hierarchical form of them. It is not the correct api for mass data retrieval of a small set of columns. Please use the ListApi for such use cases.

- [Read Requests](#) (see page 633)
 - [Single Item](#) (see page 633)
 - [Multiple Items](#) (see page 634)
 - [Query Parameters](#) (see page 636)
 - [Permissions](#) (see page 640)
 - [Object Permissions](#) (see page 640)
 - [Field Permissions](#) (see page 640)
 - [Qualification Permissions \(aka Qualified Field Permissions\)](#) (see page 640)
 - [Result](#) (see page 640)
 - [Repository Entities and Fields](#) (see page 640)
 - [Names](#) (see page 640)
 - [Datatypes](#) (see page 641)
 - [Meta attributes](#) (see page 641)
 - [Data Element](#) (see page 641)
 - [Example Result](#) (see page 642)
 - [Characteristic Values](#) (see page 667)
 - [Additional Meta Attributes](#) (see page 668)
 - [Response Examples](#) (see page 668)
 - [Examples](#) (see page 673)

Read Requests

Single Item

URL Pattern	<code>/object/{entity-identifier}/{entity_item}</code>
Method	<code>GET</code>
Accept	<code>application/json</code> <code>application/xml</code> <p>If multiple accept headers are defined, they will be evaluated by their order. The wildcard media type (*) will result in XML.</p>
Example	<code>/rest/V2.0/object/Article/4711@1</code> <code>/rest/V2.0/object/Article/'myItem'@'myCatalog'</code>

Return Codes	Code	Reason
	200 (OK)	Everything went fine
	404 (Not Found)	Object couldn't be found
	403 (Forbidden)	User has no read permission for this object
	500	Internal server error, please check the server logs

Multiple Items

This endpoint provides a multi-item interface in which the client can provide multiple items which should be returned in one call. As query parameters do have a size limitation, this api is designed as POST. This api has restrictions on the number of items which can be called. In case the limit is exceeded a corresponding return code is provided.

Contrary to the single item GET request, a missing object permission or an unknown item id does not lead to an error return code; it will just not be returned.

URL Pattern	/object/{entity-identifier}/byItems
Method	POST

Headers	<table><tr><th>Header</th><th>Supported Values</th></tr><tr><td>Accept</td><td><div>application/json</div><div>application/xml</div><p>If multiple accept headers are defined, they will be evaluated by their order. The wildcard media type (*/*) will result in XML.</p></td></tr><tr><td>Content-Type</td><td>application/x-www-form-urlencoded</td></tr></table>	Header	Supported Values	Accept	<div>application/json</div> <div>application/xml</div> <p>If multiple accept headers are defined, they will be evaluated by their order. The wildcard media type (*/*) will result in XML.</p>	Content-Type	application/x-www-form-urlencoded		
Header	Supported Values								
Accept	<div>application/json</div> <div>application/xml</div> <p>If multiple accept headers are defined, they will be evaluated by their order. The wildcard media type (*/*) will result in XML.</p>								
Content-Type	application/x-www-form-urlencoded								
Form Parameters	<table><tr><th>Parameter</th><th>Datatype</th><th>Description</th><th>Example</th></tr><tr><td>items</td><td>List of ENTITY_ITEM</td><td><p>String based entity item syntax.</p><p>The form parameter can be provided multiple times and also as a comma separated list or even a mix of both</p></td><td><div>items=4711</div><div>@1</div><div>items='myItem'@'myCatalog'</div><div>items=4711</div><div>@1, 'myItem'</div><div>@'myCatalog'</div></td></tr></table>	Parameter	Datatype	Description	Example	items	List of ENTITY_ITEM	<p>String based entity item syntax.</p> <p>The form parameter can be provided multiple times and also as a comma separated list or even a mix of both</p>	<div>items=4711</div> <div>@1</div> <div>items='myItem'@'myCatalog'</div> <div>items=4711</div> <div>@1, 'myItem'</div> <div>@'myCatalog'</div>
Parameter	Datatype	Description	Example						
items	List of ENTITY_ITEM	<p>String based entity item syntax.</p> <p>The form parameter can be provided multiple times and also as a comma separated list or even a mix of both</p>	<div>items=4711</div> <div>@1</div> <div>items='myItem'@'myCatalog'</div> <div>items=4711</div> <div>@1, 'myItem'</div> <div>@'myCatalog'</div>						
Example	<div>/rest/V2.0/object/Article/byItems</div>								

Return Codes	Code	Reason
	200 (OK)	Everything went fine
	413 (Payload too large)	<p>The amount of entity item ids exceeds the configured limit (default: 100).</p> <p>The response contains the "p360-ObjectAPI-MaxItems" header which returns the limit.</p>
	500	Internal server error, please check the server logs

Query Parameters

Please note that the qualification filter applies to all requested sub-entities which have this qualification. So, for example, if you filter for language=English, the ArticleLang and ArticleAttributeValueLang entities will both only return the English values!

Parameter	Required	Default	Datatype	Parameter description	Example
entityFilter	no	none	String	<p>Comma separated list of entity identifiers which should be part of the result. If omitted, the full object with all data is returned. We recommend to provide a list of entities which are required in order to gain performance.</p> <p>Note: In case an entityFilter is provided, also the root entity is a filter value. So by only providing the root entity as filter value, only the fields of the root level are returned in the data.</p>	<p>Only return the root fields: entityFilter=Article</p> <p>Only return the root fields, and the sales prices entityFilter=Article,ArticleSalesPrice</p> <p>Only return the language specific data like Short and Long Description entityFilter=ArticleLang</p>
qualificationFilter	no	none	String	<p>This parameter allows to restrict the output to certain qualifications. One example would be a filter so that only Euro and US-Dollar prices for the customer Informatica are returned.</p> <p>The filter string is a comma-separated list of qualification settings. A qualification setting is a <i>qualification name</i> followed by a comma-separated list of values which is put into parentheses. The qualification name is defined in the repository and is included in the Meta-API.</p>	<p>Example for prices in Euro and US-Dollar and customer Informatica: qualificationFilter=currency(EUR,USD),customer(Informatica)</p>

Parameter	Required	Default	Datatype	Parameter description	Example
revision	no	root	ENTITY_ID	The revision for which the data should be retrieved	<pre>revision='root'</pre> <pre>revision=1</pre> <pre>revision='myRevisionIdentifier'</pre> <pre>revision=4711</pre> (where 4711 is the internal id of the revision).
includeLabels	no	false	boolean	If set to true, the returned document will contain _label elements for all fields or qualifications which have an enumeration. The label will be returned in the locale of the request like this: <pre> "orderUnit" : { "_current" : { "_key" : { "_entityId" : 3100, "_internalId" : "932", "_externalId" : "'C62'" }, "_code" : "C62", "_label" : "piece" } } </pre>	<pre>includeLabels=true</pre> <pre>includeLabels=false</pre>

Parameter	Required	Default	Datatype	Parameter description	Example
includeIds	no	false	boolean	<p>If set to false, the payload avoids to contain any internal ids which might be system specific. This specifically applies to the ENTITY_ITEM datatype which will then no longer contain the "_internalId" element, only the "_externalId". Also the "_entityId" element will be replaced by an "entity" element which then holds the alphanumeric identifier of the entity and not it's numeric id.</p> <p>While reading and writing to the same Product 360 environment, we always recommend to include the ids since write performance is increased as no internal lookup needs to be done. However, when reading from one system and applying to another P360 system (different database schemas!), includeIds should be set to false, otherwise the write operation might fail since the internal ids do not match to the target system.</p> <p>In case the object is read for a 3rd party system, you might very well disable the internalIds to further reduce the response size!</p>	<pre>includeIds=true</pre> <pre>includeIds=false</pre>

Parameter	Required	Default	Datatype	Parameter description	Example
includeMimeValueArchive	no	false	boolean	<p>If set to true a ZIP Archive will be provided on the server which can be downloaded using the File API (see page 738). This archive will contain all MIME_VALUE binaries of all objects which have been returned by the call to the object API.</p> <p>Is the single item read request is used, only the mime values for this single item are part of the archive. In case the multi-item read request is used the archive will contain the values for all items.</p>	

Permissions

Object Permissions

The user needs to have the READ object permission for this API. In case he doesn't, the api returns HTTP 403 (forbidden) in case of the single item call with GET, and skip this item in case of the multiple items call.

Field Permissions

The user needs to have the READ field permissions for a field. If he doesn't, the field will not be part of the data element, but no error is returned.

Qualification Permissions (aka Qualified Field Permissions)

The user needs to have the READ permission for a qualification, e.g. for Language = English. If he doesn't, sub-entities which are qualified for this will not be part of the data element, but no error is returned.

Result

Repository Entities and Fields

Names

A new repository property "shortIdentifier" (see page 79) has been added in the custom section which is used to define the entity and field names for the json structure. The short identifier is unique within its parent entity only!

The standard repository already contains a short identifier for all fields and enabled qualifications.

Datatypes

- ENTITY_ITEM objects. Depending on the includeIds parameter they contain a different subset:
 - includeIds = true: _entityId, _internalId and _externalId
 - includeIds = false: _entity and _externalId
- MIME_VALUE objects always contain the relative _filePath, the _label and the _mimeType
- timestamp: ISO 8601, e.g.: 2012-04-23T18:25:43.511Z
- date: ISO 8601, e.g.: 2019-07-04
- numbers: standard JSON
- "no value" is either provided as null, or the attribute is not in the document.
- Empty string is returned as null, which is omitted when possible!
- Enumeration fields
 - _key : always holds the unique key of the enumeration entry
In case the enumeration field is of datatype ENTITY_ITEM, then the key will be provided with the ENTITY_ITEM syntax (see above)
The key is omitted in case includeIds parameter is set to false, and the enumeration entry does have a code.
 - _code : the external code of the enumeration entry, if the enum entry has one, and it differs from the key!
 - _label : the locale dependent label of the enumeration entry. Only if includeLabels is set to true. The locale of the http request is used for this.

Meta attributes

All meta attributes are prefixed with an underscore.

- _entity = the root entity identifier
- _entityItem = the entity item in the ENTITY_ITEM syntax
- _revision = the revision of the data as ENTITY_ITEM
- _data = The actual entity item's data
- _qualification = the qualification of the record. **The combination of the qualification values in the qualification object define this record as unique within its parent.**

Data Element

In order to be part of the data element of the object api, a sub-entity or field must fulfill multiple criteria:

- The entity or field must be active in the repository

- The entity or field must have `supportsObjectApi = true` in the repository.
In case the corresponding `FieldType` or `EntityType` does not support object api, enabling it in the custom area has no effect.
- The user must have `READ` permission for the `Entity` or `Field`
- The user must have `READ` permission for the `qualification` of the sub-entity
- The field must not be a password `Field` in the repository
- `Qualifications` must be `editable` in the repository

Entities, Fields or Qualifications which do not comply to any of these points will be omitted in the data element. No exception is thrown.

Example Result

includeIds = true	includeIds = false
<pre>{ "_entity": "Article", "_entityItem": { "_internalId": "18@1", "_entityId": 1000, "_externalId": "\u0027supplierAID-1\u0027@\u0027MASTER\u0027" }, "_revision": { "_internalId": "1", "_entityId": 5600, "_externalId": "\u0027root\u0027" }, "_container": { "_internalId": "1", "_entityId": 2900, "_externalId": "\u0027MASTER\u0027" }, "_data": { "identifier": "supplierAID-1", "manufacturerAID": "A0000000000000", "noCUperOU": 907.3600, "deliveryTime": 1.6000, "currentStatus": { "_key": 100, "_code": "NEW" }, "manufacturerName": "UQ Communications", "catalog": { "_key": { "_entityId": 2900, "_internalId": "1", "_externalId": "\u0027MASTER\u0027" } } } }</pre>	<pre>{ "_entity": "Article", "_entityItem": { "_entity": "Article", "_externalId": "\u0027supplierAID-1\u0027 @\u0027MASTER\u0027" }, "_revision": { "_entity": "Revision", "_externalId": "\u0027root\u0027" }, "_container": { "_entity": "MasterCatalog", "_externalId": "\u0027MASTER\u0027" }, "_data": { "identifier": "supplierAID-1", "manufacturerAID": "A0000000000000", "noCUperOU": 907.3600, "deliveryTime": 1.6000, "currentStatus": { "_code": "NEW" }, "manufacturerName": "UQ Communications", } }</pre>

```

    },
    "_code": "MASTER"
  },
  "quantityMin": 2.0000,
  "mainSupplier": {
    "_key": {
      "_entityId": 2800,
      "_internalId": "3",
      "_externalId": "\u0027Heiler Product
Manager\u0027"
    },
    "_code": "Heiler Product Manager"
  },
  "priceQuantity": 501.0400,
  "kitComponent": false,
  "contentUnit": {
    "_key": {
      "_entityId": 3100,
      "_internalId": "536",
      "_externalId": "\u0027MQH\u0027"
    },
    "_code": "MQH"
  },
  "quantityInterval": 1.0000,
  "kitParent": false,
  "orderUnit": {
    "_key": {
      "_entityId": 3100,
      "_internalId": "539",
      "_externalId": "\u0027CMQ\u0027"
    },
    "_code": "CMQ"
  },
  "soldOnlyInKits": false,
  "lang": [
    {
      "_qualification": {
        "language": {
          "_key": 7,
          "_code": "deu"
        }
      },
      "keywords": [
        "disinterestedness",
        "incomprehensible"
      ],
      "descriptionLong": "one side. Elizabeth
smiled at the recollection of all that she had
heard of its inhabitants. At length the
Parsonage was discernible. The garden sloping to
the road, the house standing in it, the green
pales, and the laurel hedge, everything declared

```

```

    "catalog": {
      "_code": "MASTER"
    },
    "quantityMin": 2.0000,
    "mainSupplier": {
      "_code": "Heiler
Product Manager"
    },
    "priceQuantity":
501.0400,
    "kitComponent": false,
    "contentUnit": {
      "_code": "MQH"
    },
    "quantityInterval":
1.0000,
    "kitParent": false,
    "orderUnit": {
      "_code": "CMQ"
    },
    "soldOnlyInKits":
false,
    "lang": [
      {
        "_qualification":
{
          "language": {
            "_code": "eng"
          },
          "keywords": [
            "curious",
            "disinterestedness"
          ],
          "descriptionShort":
"thanks and assurances of
happiness. She had spent
six weeks with great
enjoyment; and the
pleasure of being with
Charlotte, and the"
        },
        {
          "_qualification":
{
            "language": {
              "_code": "fra"
            },
            "keywords": [

```

they were arriving. Mr. Collins and Charlotte appeared at the door, and the carriage stopped at the small gate which led by a short gravel walk to the house, amidst the nods and smiles of the whole party. In a moment they were all out of the chaise, rejoicing at the sight of each other. Mrs. Collins welcomed her friend with the liveliest pleasure, and Elizabeth was more and more satisfied with coming when she found herself so affectionately received. She saw instantly that her cousin's manners w",

"descriptionShort": "a few minutes' conversation with Charlotte, but was scarcely ever prevailed upon to get out. Very few days passed in which Mr."

```
    },
    {
      "_qualification": {
        "language": {
          "_key": 12,
          "_code": "fra"
        }
      },
      "keywords": [
        "communicativeness"
      ],
      "descriptionLong": "yet it would seem, by her manner of talking, as if she wanted to persuade herself that he is really partial to Miss Darcy. I cannot understand it. If I were not afraid of judging harshly, I should be almost tempted to say that there is a strong appearance of duplicity in all this. But I will endeavour to banish every painful thought, and think only of what will make me happy—your affection, and the invariable kindness of my dear uncle and aunt. Let me hear from you very soon. Miss Bingley said something of his never returning to Netherfield again, of giving up the house, but not with any certainty. We had better not mention it. I am extremely glad that you have such pleasant accounts from our friends at Hunsford. Pray go to see them, with Sir William and Maria. I am sure you",
```

"descriptionShort": "it is not true. A great many changes have happened in the neighbourhood, since you went away. Miss Lucas is married and settled. And"

```
    },
    {
      "_qualification": {
        "language": {
          "_key": 9,
```

"communicativeness"

```
    ],
    "descriptionLong":
    "yet it would seem, by her manner of talking, as if she wanted to persuade herself that he is really partial to Miss Darcy. I cannot understand it. If I were not afraid of judging harshly, I should be almost tempted to say that there is a strong appearance of duplicity in all this. But I will endeavour to banish every painful thought, and think only of what will make me happy—your affection, and the invariable kindness of my dear uncle and aunt. Let me hear from you very soon. Miss Bingley said something of his never returning to Netherfield again, of giving up the house, but not with any certainty. We had better not mention it. I am extremely glad that you have such pleasant accounts from our friends at Hunsford. Pray go to see them, with Sir William and Maria. I am sure you",
```

"descriptionShort": "it is not true. A great many changes have happened in the neighbourhood, since you went away. Miss Lucas is married and settled. And"

```
    },
    {
      "_qualification":
    {
      "language": {
        "_code": "deu"
      }
    },
```

```

        "_code": "eng"
      }
    },
    "keywords": [
      "curious",
      "disinterestedness"
    ],
    "descriptionShort": "thanks and
assurances of happiness. She had spent six weeks
with great enjoyment; and the pleasure of being
with Charlotte, and the"
  },
  {
    "_qualification": {
      "language": {
        "_key": 10,
        "_code": "esl"
      }
    },
    "keywords": [
      "intermarriage",
      "disinterestedness",
      "stretched"
    ],
    "descriptionLong": "invitation was
accepted of course, and at a proper hour they
joined the party in Lady Catherine's drawing-
room. Her ladyship received them civilly, but it
was plain that their company was by no means so
acceptable as when she could get nobody else;
and she was, in fact, almost engrossed by her
nephews, speaking to them, especially to Darcy,
mu",
    "descriptionShort": "in the best manner
that his profession might allow-and if he took
orders, desired that a valuable family living
might be his as soon as it becam"
  }
],
"pricePurchase": [
  {
    "_qualification": {
      "supplier": {
        "_key": {
          "_entityId": 2800,
          "_internalId": "3",
          "_externalId": "\u0027Heiler
Product Manager\u0027"
        },
        "_code": "Heiler Product Manager"
      },
      "currency": {

```

```

      "keywords": [
        "disinterestedness",
        "incomprehensible"
      ],
      "descriptionLong":
"one side. Elizabeth
smiled at the recollection
of all that she had heard
of its inhabitants. At
length the Parsonage was
discernible. The garden
sloping to the road, the
house standing in it, the
green pales, and the
laurel hedge, everything
declared they were
arriving. Mr. Collins and
Charlotte appeared at the
door, and the carriage
stopped at the small gate
which led by a short
gravel walk to the house,
amidst the nods and smiles
of the whole party. In a
moment they were all out
of the chaise, rejoicing
at the sight of each
other. Mrs. Collins
welcomed her friend with
the liveliest pleasure,
and Elizabeth was more and
more satisfied with coming
when she found herself so
affectionately received.
She saw instantly that her
cousin's manners w",
      "descriptionShort": "a few
minutes' conversation with
Charlotte, but was
scarcely ever prevailed
upon to get out. Very few
days passed in which Mr."
    },
    {
      "_qualification":
{
        "language": {
          "_code": "esl"
        }

```



```

        "_key": "GBP"
    },
    "type": {
        "_key": 1,
        "_code": "net_list"
    },
    "territory": {
        "_key": "DE"
    },
    "validFrom": "1899-12-30",
    "validTo": "9999-12-31"
},
"value": [
    {
        "_qualification": {
            "lowerBound": 3.0000
        },
        "amount": 224648.610000,
        "factor": 1.0000
    }
]
},
],
"priceSales": [
    {
        "_qualification": {
            "currency": {
                "_key": "CHF"
            },
            "type": {
                "_key": 5,
                "_code": "nrp"
            },
            "customer": {
                "_key": {
                    "_entityId": 2800,
                    "_internalId": "1",
                    "_externalId":
"\u0027Public\u0027"
                },
            },
            "territory": {
                "_key": "DE"
            },
            "validFrom": "1899-12-30",
            "validTo": "9999-12-31"
        },
        "value": [
            {
                "_qualification": {
                    "lowerBound": 1.0000
                },

```

```

    },
    "keywords": [
        "intermarriage",
        "stretched",
        "disinterestedness"
    ],
    "descriptionLong":
"invitation was accepted
of course, and at a proper
hour they joined the party
in Lady Catherine's
drawing-room. Her ladyship
received them civilly, but
it was plain that their
company was by no means so
acceptable as when she
could get nobody else; and
she was, in fact, almost
engrossed by her nephews,
speaking to them,
especially to Darcy, mu",
    "descriptionShort": "in
the best manner that his
profession might allow--and
if he took orders, desired
that a valuable family
living might be his as
soon as it became"
    },
    ],
    "priceSales": [
        {
            "_qualification":
{
                "currency": {
                    "_code": "CHF"
                },
                "type": {
                    "_code": "nrp"
                },
                "customer": {
                    "_key": {
                        "_entity":
"Party",
                    },
                    "_externalId":
"\u0027Public\u0027"
                },
                "territory": {

```

```

        "amount": 45558.800000,
        "factor": 1.0000
    }
]
},
{
    "_qualification": {
        "currency": {
            "_key": "USD"
        },
        "type": {
            "_key": 3,
            "_code": "net_customer"
        },
        "customer": {
            "_key": {
                "_entityId": 2800,
                "_internalId": "1",
                "_externalId":
"\u0027Public\u0027"
            },
            "territory": {
                "_key": "DE"
            },
            "validFrom": "1899-12-30",
            "validTo": "9999-12-31"
        },
        "value": [
            {
                "_qualification": {
                    "lowerBound": 3.0000
                },
                "amount": 18416.580000,
                "factor": 1.0000
            }
        ]
    }
},
{
    "logistic": [
        {
            "_qualification": {
                "territory": {
                    "_key": "MX"
                }
            },
            "grossWeight": 32.3000,
            "packageWeight": 23.0700,
            "packageUnit": {
                "_key": {
                    "_entityId": 3100,
                    "_internalId": "7042",

```

```

            "_code": "DE"
        },
        "validFrom":
"1899-12-30",
        "validTo":
"9999-12-31"
    },
    "value": [
        {
            "_qualification": {
                "lowerBound": 1.0000
            },
            "amount":
45558.800000,
            "factor":
1.0000
        }
    ],
    {
        "_qualification":
{
            "currency": {
                "_code": "USD"
            },
            "type": {
                "_code":
"net_customer"
            },
            "customer": {
                "_key": {
                    "_entity":
"Party",
                    "_externalId":
"\u0027Public\u0027"
                },
                "territory": {
                    "_code": "DE"
                },
                "validFrom":
"1899-12-30",
                "validTo":
"9999-12-31"
            },
            "value": [
                {
                    "_qualification": {

```

```

        "_externalId": "\u0027HEP\u0027"
      },
      "_code": "HEP"
    },
    "originCountry": {
      "_key": "AT"
    },
    "grossWeightUnit": {
      "_key": {
        "_entityId": 3100,
        "_internalId": "7119",
        "_externalId": "\u0027mL/(72.h)
\u0027"
      },
      "_code": "mL/(72.h)"
    }
  ],
  "logisticExtension": [
    {
      "_qualification": {
        "packagingUnit": {
          "_key": {
            "_entityId": 3100,
            "_internalId": "1027",
            "_externalId": "\u0027DI\u0027"
          },
          "_code": "DI"
        },
        "language": {
          "_key": 7,
          "_code": "deu"
        },
        "party": {
          "_key": {
            "_entityId": 2800,
            "_internalId": "3",
            "_externalId": "\u0027Heiler
Product Manager\u0027"
          },
          "_code": "Heiler Product Manager"
        }
      },
      "length": 99122788.3034,
      "code128": "09040990267667268090",
      "code39": "27655316938891158712",
      "lengthUnit": {
        "_key": {
          "_entityId": 3100,
          "_internalId": "3013",
          "_externalId":
"\u0027mig_WA_DATANORM40\u0027"

```

```

      "lowerBound": 3.0000
    },
    "amount":
18416.580000,
    "factor":
1.0000
  ]
},
"pricePurchase": [
  {
    "_qualification":
{
      "supplier": {
        "_code":
"Heiler Product Manager"
      },
      "currency": {
        "_code": "GBP"
      },
      "type": {
        "_code":
"net_list"
      },
      "territory": {
        "_code": "DE"
      },
      "validFrom":
"1899-12-30",
      "validTo":
"9999-12-31"
    },
    "value": [
      {
        "_qualification": {
          "lowerBound": 3.0000
        },
        "amount":
224648.610000,
        "factor":
1.0000
      }
    ],
    "logistic": [
      {
        "_qualification":

```

```

    },
    "_code": "mig_WA_DATANORM40"
  },
  "heightUnit": {
    "_key": {
      "_entityId": 3100,
      "_internalId": "804",
      "_externalId": "\u0027BH\u0027"
    },
    "_code": "BH"
  },
  "height": 36999719.5867
},
{
  "_qualification": {
    "packagingUnit": {
      "_key": {
        "_entityId": 3100,
        "_internalId": "456",
        "_externalId": "\u0027KL\u0027"
      },
      "_code": "KL"
    },
    "language": {
      "_key": 9,
      "_code": "eng"
    },
    "party": {
      "_key": {
        "_entityId": 2800,
        "_internalId": "3",
        "_externalId": "\u0027Heiler
Product Manager\u0027"
      },
      "_code": "Heiler Product Manager"
    }
  },
  "length": 28554703.0225,
  "code128": "82915251799710733746",
  "code39": "21863593155379329992",
  "lengthUnit": {
    "_key": {
      "_entityId": 3100,
      "_internalId": "131",
      "_externalId": "\u0027RM\u0027"
    },
    "_code": "RM"
  },
  "heightUnit": {
    "_key": {
      "_entityId": 3100,
      "_internalId": "404",

```

```

    {
      "territory": {
        "_code": "MX"
      }
    },
    "grossWeight":
32.3000,
    "packageWeight":
23.0700,
    "packageUnit": {
      "_code": "HEP"
    },
    "originCountry": {
      "_code": "AT"
    },
    "grossWeightUnit":
{
  "_code": "mL/
(72.h)"
},
    "logisticExtension": [
      {
        "_qualification":
{
  "packagingUnit":
{
    "_code": "DI"
  },
  "language": {
    "_code": "deu"
  },
  "party": {
    "_code":
"Heiler Product Manager"
  }
},
      "length":
99122788.3034,
      "code128":
"09040990267667268090",
      "code39":
"27655316938891158712",
      "lengthUnit": {
        "_code":
"mig_WA_DATANORM40"
      },
      "heightUnit": {
        "_code": "BH"
      },
      "height":
36999719.5867

```

```

        "_externalId": "\u0027JOU\u0027"
      },
      "_code": "JOU"
    },
    "height": 27955841.1420
  },
  {
    "_qualification": {
      "packagingUnit": {
        "_key": {
          "_entityId": 3100,
          "_internalId": "1013",
          "_externalId": "\u0027WR\u0027"
        },
        "_code": "WR"
      },
      "language": {
        "_key": 29,
        "_code": "sve"
      },
      "party": {
        "_key": {
          "_entityId": 2800,
          "_internalId": "3",
          "_externalId": "\u0027Heiler
Product Manager\u0027"
        },
        "_code": "Heiler Product Manager"
      }
    },
    "length": 53266349.3016,
    "code128": "55240848258305284982",
    "code39": "44402274049146575392",
    "lengthUnit": {
      "_key": {
        "_entityId": 3100,
        "_internalId": "378",
        "_externalId": "\u0027HTZ\u0027"
      },
      "_code": "HTZ"
    },
    "heightUnit": {
      "_key": {
        "_entityId": 3100,
        "_internalId": "928",
        "_externalId": "\u0027D79\u0027"
      },
      "_code": "D79"
    },
    "height": 37828907.4796
  }
],

```

```

    },
    {
      "_qualification":
    {
      "packagingUnit":
    {
      "_code": "KL"
    },
      "language": {
        "_code": "eng"
      },
      "party": {
        "_code":
      "Heiler Product Manager"
      }
    },
      "length":
      28554703.0225,
      "code128":
      "82915251799710733746",
      "code39":
      "21863593155379329992",
      "lengthUnit": {
        "_code": "RM"
      },
      "heightUnit": {
        "_code": "JOU"
      },
      "height":
      27955841.1420
    },
    {
      "_qualification":
    {
      "packagingUnit":
    {
      "_code": "WR"
    },
      "language": {
        "_code": "sve"
      },
      "party": {
        "_code":
      "Heiler Product Manager"
      }
    },
      "length":
      53266349.3016,
      "code128":
      "55240848258305284982",
      "code39":
      "44402274049146575392",

```

```

    "referencedItem": [
      {
        "_qualification": {
          "referencedIdentifier":
"supplierAID-889",
          "referencedCatalogIdentifier":
"MASTER",
          "type": {
            "_key": 9,
            "_code": "consists_of"
          }
        },
        "quantity": 1,
        "referencedItem": {
          "_entityId": 1000,
          "_internalId": "326@1",
          "_externalId":
"\u0027supplierAID-889\u0027@\u0027MASTER\u0027"
        }
      }
    ],
    "structureMap": [
      {
        "_qualification": {
          "structure": {
            "_key": 15,
            "_code": "ECLASS-6.1"
          }
        },
        "structureGroups": [
          {
            "_entityId": 3000,
            "_internalId": "22884@15",
            "_externalId":
"\u0027AKL536004\u0027@\u0027ECLASS-6.1\u0027"
          }
        ],
        "manualMap": [
          "36-13-90-90"
        ]
      }
    ],
    "structureGroupMap": [
      {
        "_qualification": {
          "structure": {
            "_key": {
              "_entityId": 2300,
              "_internalId": "15",
              "_externalId":
"\u0027ECLASS-6.1\u0027"
            },

```

```

          "lengthUnit": {
            "_code": "HTZ"
          },
          "heightUnit": {
            "_code": "D79"
          },
          "height":
37828907.4796
        },
        "referencedItem": [
          {
            "_qualification":
{
              "referencedIdentifier":
"supplierAID-889",
              "referencedCatalogIdentifi
er": "MASTER",
              "type": {
                "_code":
"consists_of"
              },
              "quantity": 1,
              "referencedItem":
{
                "_entity":
"Article",
                "_externalId":
"\u0027supplierAID-889\u00
27@\u0027MASTER\u0027"
              }
            },
            "structureMap": [
              {
                "_qualification":
{
                  "structure": {
                    "_code":
"ECLASS-6.1"
                  },
                  "structureGroups":
[
                    {
                      "_entity":
"StructureGroup",
                      "_externalId":
"\u0027AKL536004\u0027@\u0

```

```

        "_code": "ECLASS-6.1"
      },
      "structureGroup": {
        "_entityId": 3000,
        "_internalId": "22884@15",
        "_externalId":
"\u0027AKL536004\u0027@\u0027ECLASS-6.1\u0027"
      }
    }
  ],
  "mediaAsset": [
    {
      "_qualification": {
        "type": {
          "_key": "BACK_VIEW"
        }
      },
      "mediaAsset": {
        "_entityId": 2400,
        "_internalId": "22",
        "_externalId":
"\u0027MediaAsset_1643730037754020\u0027"
      },
      "name": "not have made the offer of your
hand in any possible way that would have te",
      "category": "preservative",
      "document": [
        {
          "_qualification": {
            "language": {
              "_key": -1,
              "_code": "zxx"
            },
            "quality": {
              "_key": "highres"
            }
          },
          "identifier": "hlr-system/
ugngddhkkynrymx.jpg",
          "imageIdentifier": "hlr-system/
ugngddhkkynrymx.jpg",
          "order": 1
        }
      ]
    },
    {
      "_qualification": {
        "type": {
          "_key": "thumbnail"
        }
      },

```

```

027ECLASS-6.1\u0027"
      }
    ],
    "manualMap": [
      "36-13-90-90"
    ]
  },
  "structureGroupMap": [
    {
      "_qualification":
{
        "structure": {
          "_code":
"ECLASS-6.1"
        },
        "structureGroup": {
          "_entity":
"StructureGroup",
          "_externalId":
"\u0027AKL536004\u0027@\u0
027ECLASS-6.1\u0027"
        }
      }
    ],
    "mediaAsset": [
      {
        "_qualification":
{
          "type": {
            "_code":
"thumbnail"
          },
          "mediaAsset": {
            "_entity":
"MediaAsset",
            "_externalId":
"\u0027MediaAsset_16437300
37754004\u0027"
          },
          "name": "settled,
that had given the",
          "category":
"lessened",
          "document": [
            {
              "_qualification": {
                "language":
{

```

```

      "mediaAsset": {
        "_entityId": 2400,
        "_internalId": "8",
        "_externalId":
"\u0027MediaAsset_1643730037754004\u0027"
      },
      "name": "settled, that had given the",
      "category": "lessened",
      "document": [
        {
          "_qualification": {
            "language": {
              "_key": 19,
              "_code": "dut"
            },
            "quality": {
              "_key": "highres"
            }
          },
          "identifier": "hlr-system/
uwhrxrrlxhohqnn.jpg",
          "imageIdentifier": "hlr-system/
uwhrxrrlxhohqnn.jpg",
          "order": 1
        }
      ]
    },
    "log": [
      {
        "_qualification": {
          "channel": {
            "_key": "HPM"
          }
        },
        "creationUser": {
          "_key": {
            "_entityId": 2600,
            "_internalId": "1",
            "_externalId":
"\u0027Administrator\u0027"
          },
          "_code": "Administrator"
        },
        "creationDate":
"2022-03-11T14:52:36.300Z"
      }
    ],
    "ownLog": [
      {
        "modificationDate":
"2022-03-11T14:52:36.583Z"
      }
    ]
  }
}

```

```

      "_code":
"dut"
    },
    "quality": {
      "_key":
"highres"
    }
  },
  "identifier":
"hlr-system/
uwhrxrrlxhohqnn.jpg",
  "imageIdentifier": "hlr-
system/
uwhrxrrlxhohqnn.jpg",
  "order": 1
}
],
{
  "_qualification":
{
    "type": {
      "_code":
"BACK_VIEW"
    }
  },
  "mediaAsset": {
    "_entity":
"MediaAsset",
    "_externalId":
"\u0027MediaAsset_16437300
37754020\u0027"
  },
  "name": "not have
made the offer of your
hand in any possible way
that would have te",
  "category":
"preservative",
  "document": [
    {
      "_qualification": {
        "language":
{
          "_code":
"zxx"
        },
        "quality": {
          "_key":
"highres"
        }
      }
    ]
  }
}

```



```

    }
  ],
  "marketing": [
    {
      "_qualification": {
        "targetMarket": {
          "_key": "EN"
        }
      },
      "specialItemCode": [
        {
          "_key": "GIFT_WITH_PURCHASE"
        }
      ],
      "couponFamilyCode": [
        "23736760493674713570",
        "59114057533905607311"
      ],
      "lang": [
        {
          "_qualification": {
            "sequence": 1,
            "language": {
              "_key": 12,
              "_code": "fra"
            }
          },
          "marketingMessage": "not so good an
income as yours. Do you draw?" "No, not at all."
"What, none of you?" "Not one." "That is very
strange. But I suppose you had no opportunity.
Your mother should have taken you to town every
spring for the benefit of masters." "My mother
would have had no objection, but my father hates
London." "Has your governess left you?" "We
never had any governess." "No governess! How was
that possible? Five daughters brought up at home
without a governess! I never"
        },
        {
          "_qualification": {
            "sequence": 0,
            "language": {
              "_key": 10,
              "_code": "esl"
            }
          },
          "marketingMessage": "brother-in-law,
Mr. Hurst, merely looked the gentleman; but his
frie"
        }
      ]
    }
  ]

```

```

    },
    "identifier":
"hlr-system/
ugngddhkynrymx.jpg",
"imageIdentifier": "hlr-
system/
ugngddhkynrymx.jpg",
"order": 1
  }
],
"log": [
  {
    "_qualification":
{
      "channel": {
        "_code": "HPM"
      },
      "creationUser": {
        "_code":
"Administrator"
      },
      "creationDate":
"2022-03-11T14:52:36.300Z"
    },
    "ownLog": [
      {
        "modificationDate":
"2022-03-11T14:52:36.583Z"
      },
      "marketing": [
        {
          "_qualification":
{
            "targetMarket":
{
              "_code": "DE"
            },
            "specialItemCode":
[
              {
                "_code":
"LIMITED_PRODUCTION"
              }
            ]
          }
        }
      ]
    }
  ]
}

```

```

    },
    {
      "_qualification": {
        "targetMarket": {
          "_key": "DE"
        }
      },
      "specialItemCode": [
        {
          "_key": "LIMITED_PRODUCTION"
        }
      ],
      "couponFamilyCode": [
        "83337210683270677496",
        "68772233910188927019"
      ],
      "lang": [
        {
          "_qualification": {
            "sequence": 1,
            "language": {
              "_key": 9,
              "_code": "eng"
            }
          },
          "marketingMessage": "It was a
journey of only twenty-four miles, and they
began it so early as to be in Gracechurch Street
by noon. As they drove to Mr. Gardiner's door,
Jane was at a drawing-room window watching
their"
        },
        {
          "_qualification": {
            "sequence": 0,
            "language": {
              "_key": 7,
              "_code": "deu"
            }
          },
          "marketingMessage": "a mixture of
pride and obsequiousness, self-importance and
humility. Having now a good house and a very
sufficient income, he intended to marry; and in
seeking a reconciliation with"
        }
      ]
    }
  ],
  "_characteristicRecords": [
    {
      "_qualification": {

```

```

    ],
    "couponFamilyCode": [
      "83337210683270677496",
      "68772233910188927019"
    ],
    "lang": [
      {
        "_qualification": {
          "sequence":
1,
          "language":
{
            "_code":
"eng"
          }
        },
        "marketingMessage": "It
was a journey of only
twenty-four miles, and
they began it so early as
to be in Gracechurch
Street by noon. As they
drove to Mr. Gardiner's
door, Jane was at a
drawing-room window
watching their"
      },
      {
        "_qualification": {
          "sequence":
0,
          "language":
{
            "_code":
"deu"
          }
        },
        "marketingMessage": "a
mixture of pride and
obsequiousness, self-
importance and humility.
Having now a good house
and a very sufficient
income, he intended to
marry; and in seeking a

```

```

        "characteristic": {
            "_key": {
                "_entityId": 8000,
                "_internalId": "829",
                "_externalId":
"\u0027nutrientFormatTypeCodeReference\u0027"
            },
            "_code":
"nutrientFormatTypeCodeReference"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "root"
    },
    "_datatype": "LOOKUP",
    "order": 1,
    "_recordLang": [
        {
            "_qualification": {
                "language": {
                    "_key": -1,
                    "_code": "zxx"
                }
            },
            "values": [
                {
                    "_key": {
                        "_entityId": 7300,
                        "_internalId": "4129@305",
                        "_externalId":
"\u0027US_FDA_SFP_MULTI_VITAMINS_IN_PACKAGES\u00
27@\u0027nutrientFormatTypeCodeReference\u0027"
                    },
                    "_code":
"US_FDA_SFP_MULTI_VITAMINS_IN_PACKAGES"
                },
                {
                    "_key": {
                        "_entityId": 7300,
                        "_internalId": "4056@305",
                        "_externalId":
"\u0027US_FDA_NFP_2020_STANDARD_INFANTS_TO_12_MO
NTHS\u0027@\u0027nutrientFormatTypeCodeReference
\u0027"
                    },
                    "_code":
"US_FDA_NFP_2020_STANDARD_INFANTS_TO_12_MONTHS"
                },
                {
                    "_key": {
                        "_entityId": 7300,
                        "_internalId": "2997@305",
                        "_externalId":

```

```

reconciliation with"
            }
        ]
    },
    {
        "_qualification":
{
            "targetMarket":
{
                "_code": "EN"
            }
        },
        "specialItemCode":
[
            {
                "_code":
"GIFT_WITH_PURCHASE"
            }
        ],
        "couponFamilyCode": [
            "23736760493674713570",
            "59114057533905607311"
        ],
        "lang": [
            {
                "_qualification": {
                    "sequence":
1,
                    "language":
{
                        "_code":
"fra"
                    }
                },
                "marketingMessage": "not
so good an income as
yours. Do you draw?" "No,
not at all." "What, none
of you?" "Not one." "That
is very strange. But I
suppose you had no
opportunity. Your mother
should have taken you to
town every spring for the
benefit of masters." "My
mother would have had no
objection, but my father
hates London." "Has your

```

```
"\u0027NUTRIENT_DATA_EXEMPT\u0027@\u0027nutrient
FormatTypeCodeReference\u0027"
```

```
    },
    "_code": "NUTRIENT_DATA_EXEMPT"
  }
]
}
],
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "765",
        "_externalId":
"\u0027juiceContentPercentage\u0027"
      },
      "_code": "juiceContentPercentage"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "DECIMAL",
  "_formatPattern": "###.##",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": -1,
          "_code": "zxx"
        }
      },
      "values": [
        1635.15
      ]
    }
  ],
  {
    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,
          "_internalId": "764",
          "_externalId":
"\u0027ingredientStatement\u0027"
        },
        "_code": "ingredientStatement"
      },
      "recordKey": "0000.0000.RK",
      "parentRecordKey": "root"
    }
  }
}
```

governess left you?" "We never had any governess." "No governess! How was that possible? Five daughters brought up at home without a governess! I never"

```
    },
    {
      "_qualification": {
        "sequence":
0,
        "language":
{
          "_code":
"esl"
        }
      },
      "marketingMessage":
"brother-in-law, Mr.
Hurst, merely looked the
gentleman; but his frie"
    }
  ],
  "_characteristicRecords":
[
  {
    "_qualification":
{
      "characteristic": {
        "_code":
"nutrientFormatTypeCodeRef
erence"
      },
      "recordKey":
"0000.0000.RK",
      "parentRecordKey": "root"
    },
    "_datatype":
"LOOKUP",
    "order": 1,
    "_recordLang": [
      {
        "_qualification": {
          "language":
```

```

    },
    "_datatype": "TEXT",
    "order": 1,
    "_recordLang": [
      {
        "_qualification": {
          "language": {
            "_key": 11,
            "_code": "fin"
          }
        },
        "values": [
          "did she think it for Bingley and
her sister that s"
        ]
      }
    ],
    {
      "_qualification": {
        "characteristic": {
          "_key": {
            "_entityId": 8000,
            "_internalId": "763",
            "_externalId":
"\u0027ingredientOfConcernCode\u0027"
          },
          "_code": "ingredientOfConcernCode"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "root"
      },
      "_datatype": "LOOKUP",
      "order": 1,
      "_recordLang": [
        {
          "_qualification": {
            "language": {
              "_key": -1,
              "_code": "zxx"
            }
          },
          "values": [
            {
              "_key": {
                "_entityId": 7300,
                "_internalId": "2518@283",
                "_externalId":
"\u0027IODISED_SALT\u0027@\u0027ingredientOfConc
ernCode\u0027"
              },
              "_code": "IODISED_SALT"
            }
          ]
        }
      ]
    }
  ],
  {
    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,
          "_internalId": "763",
          "_externalId":
"\u0027ingredientOfConcernCode\u0027"
        },
        "_code": "ingredientOfConcernCode"
      },
      "recordKey": "0000.0000.RK",
      "parentRecordKey": "root"
    },
    "_datatype": "LOOKUP",
    "order": 1,
    "_recordLang": [
      {
        "_qualification": {
          "language": {
            "_key": -1,
            "_code": "zxx"
          }
        },
        "values": [
          {
            "_key": {
              "_entityId": 7300,
              "_internalId": "2518@283",
              "_externalId":
"\u0027IODISED_SALT\u0027@\u0027ingredientOfConc
ernCode\u0027"
            },
            "_code": "IODISED_SALT"
          }
        ]
      }
    ]
  }
]
},
{
  "_qualification": {
    "characteristic": {
      "_code":
"juiceContentPercentage"
    },
    "recordKey":
"0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype":
"DECIMAL",
  "_formatPattern":
"###.##",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_code":
"zxx"
        }
      },
      "values": [

```

```

{
  "_code":
"zxx"
},
"values": [
  {
    "_code":
"US_FDA_SFP_MULTI_VITAMINS
_IN_PACKAGES"
  },
  {
    "_code":
"NUTRIENT_DATA_EXEMPT"
  },
  {
    "_code":
"US_FDA_NFP_2020_STANDARD_
INFANTS_TO_12_MONTHS"
  }
]
},
{
  "_qualification": {
    "characteristic": {
      "_code":
"juiceContentPercentage"
    },
    "recordKey":
"0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype":
"DECIMAL",
  "_formatPattern":
"###.##",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_code":
"zxx"
        }
      },
      "values": [

```

```

        }
      ]
    }
  ],
},
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "730",
        "_externalId":
"\u0027dietTypeDescription\u0027"
      },
      "_code": "dietTypeDescription"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "TEXT",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": 29,
          "_code": "sve"
        }
      },
      "values": [
        "admiration of Captain Carter, and
her hope of seeing him in the course of the day"
      ]
    }
  ],
},
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "729",
        "_externalId":
"\u0027surfaceOfCheeseAtEndOfRipeningCode\u0027"
      },
      "_code":
"surfaceOfCheeseAtEndOfRipeningCode"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "LOOKUP",

```

```

        1635.15
      ]
    }
  ],
},
{
  "_qualification":
{
  "characteristic": {
    "_code":
"ingredientStatement"
  },
  "recordKey":
"0000.0000.RK",
  "parentRecordKey": "root"
  },
  "_datatype":
"TEXT",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language":
{
          "_code":
"fin"
        }
      },
      "values": [
        "did she
think it for Bingley and
her sister that s"
      ]
    }
  ],
},
{
  "_qualification":
{
  "characteristic": {
    "_code":
"ingredientOfConcernCode"
  },
  "recordKey":
"0000.0000.RK",
  "parentRecordKey": "root"
  },

```

```

"order": 1,
"_recordLang": [
  {
    "_qualification": {
      "language": {
        "_key": -1,
        "_code": "zxx"
      }
    },
    "values": [
      {
        "_key": {
          "_entityId": 7300,
          "_internalId": "1580@275",
          "_externalId":
"\u0027SOFT_RIPENED_MOULD_RIND\u0027@\u0027surfa
ceOfCheeseAtEndOfRipeningCode\u0027"
        },
        "_code":
"SOFT_RIPENED_MOULD_RIND"
      }
    ]
  }
],
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "728",
        "_externalId":
"\u0027isRindEdible\u0027"
      },
      "_code": "isRindEdible"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "LOOKUP",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": -1,
          "_code": "zxx"
        }
      },
      "values": [
        {
          "_key": {

```

```

    "_datatype":
"LOOKUP",
    "order": 1,
    "_recordLang": [
      {
        "_qualification": {
          "language":
{
            "_code":
"zxx"
          }
        },
        "values": [
          {
            "_code":
"IODISED_SALT"
          }
        ]
      }
    ],
    {
      "_qualification":
{
        "characteristic": {
          "_code":
"dietTypeDescription"
        },
        "recordKey":
"0000.0000.RK",
        "parentRecordKey": "root"
      },
      "_datatype":
"TEXT",
      "order": 1,
      "_recordLang": [
        {
          "_qualification": {
            "language":
{
              "_code":
"sve"
            }
          },
          "values": [
            "admiration
of Captain Carter, and her
hope of seeing him in the

```

```

        "_entityId": 7300,
        "_internalId": "704@271",
        "_externalId":
"\u0027FALSE\u0027@\u0027NonBinaryLogicCodes\u00
27"
    },
    "_code": "FALSE"
  }
]
},
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "727",
        "_externalId":
"\u0027fatPercentageInDryMatter\u0027"
      },
      "_code": "fatPercentageInDryMatter"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "DECIMAL",
  "_formatPattern": "##.###",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": -1,
          "_code": "zxx"
        }
      },
      "values": [
        641.07
      ]
    }
  ],
  {
    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,
          "_internalId": "726",
          "_externalId":
"\u0027cheeseMaturationProcessContainerTypeCode\u
u0027"

```

```

course of the day"
    ]
  }
]
},
{
  "_qualification":
{
  "characteristic": {
    "_code":
"surfaceOfCheeseAtEndOfRip
eningCode"
  },
  "recordKey":
"0000.0000.RK",
  "parentRecordKey": "root"
  },
  "_datatype":
"LOOKUP",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language":
{
          "_code":
"zxx"
        }
      },
      "values": [
        {
          "_code":
"SOFT_RIPENED_MOULD_RIND"
        }
      ]
    }
  ],
  {
    "_qualification":
{
  "characteristic": {
    "_code":
"isRindEdible"
  },
  "recordKey":
"0000.0000.RK",
  "parentRecordKey": "root"

```



```

    },
    "_code":
"cheeseMaturationProcessContainerTypeCode"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "LOOKUP",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": -1,
          "_code": "zxx"
        }
      },
      "values": [
        {
          "_key": {
            "_entityId": 7300,
            "_internalId": "1545@273",
            "_externalId":
"\u0027MOULD\u0027@\u0027cheeseMaturationProcess
ContainerTypeCode\u0027"
          },
          "_code": "MOULD"
        }
      ]
    }
  ],
  {
    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,
          "_internalId": "725",
          "_externalId":
"\u0027cheeseMaturationPeriodDescription\u0027"
        },
        "_code":
"cheeseMaturationPeriodDescription"
      },
      "recordKey": "0000.0000.RK",
      "parentRecordKey": "root"
    },
    "_datatype": "TEXT",
    "order": 1,
    "_recordLang": [
      {
        "_qualification": {

```

```

    },
    "_datatype":
"LOOKUP",
    "order": 1,
    "_recordLang": [
      {
        "_qualification": {
          "language":
{
            "_code":
"zxx"
          }
        },
        "values": [
          {
            "_code":
"FALSE"
          }
        ]
      },
      {
        "_qualification":
{
          "characteristic": {
            "_code":
"fatPercentageInDryMatter"
          },
          "recordKey":
"0000.0000.RK",
          "parentRecordKey": "root"
        },
        "_datatype":
"DECIMAL",
        "_formatPattern":
"##.###",
        "order": 1,
        "_recordLang": [
          {
            "_qualification": {
              "language":
{
                "_code":
"zxx"
              }
            },
            "values": [

```

```

        "language": {
            "_key": 4,
            "_code": "chi"
        }
    },
    "values": [
        "were well, I hope, when you left
Londo"
    ]
}
],
},
{
    "_qualification": {
        "characteristic": {
            "_key": {
                "_entityId": 8000,
                "_internalId": "721",
                "_externalId":
"\u0027rennetTypeCode\u0027"
            },
            "_code": "rennetTypeCode"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "root"
    },
    "_datatype": "LOOKUP",
    "order": 1,
    "_recordLang": [
        {
            "_qualification": {
                "language": {
                    "_key": -1,
                    "_code": "zxx"
                }
            },
            "values": [
                {
                    "_key": {
                        "_entityId": 7300,
                        "_internalId": "1335@276",
                        "_externalId":
"\u0027NO_RENNET\u0027@\u0027rennetTypeCode\u0027"
                    },
                    "_code": "NO_RENNET"
                }
            ]
        }
    ],
},
{

```

```

        641.07
    ]
}
],
},
{
    "_qualification":
{
    "characteristic": {
        "_code":
"cheeseMaturationProcessCo
ntainerTypeCode"
    },
    "recordKey":
"0000.0000.RK",
    "parentRecordKey": "root"
    },
    "_datatype":
"LOOKUP",
    "order": 1,
    "_recordLang": [
        {
            "_qualification": {
                "language":
{
                    "_code":
"zxx"
                }
            },
            "values": [
                {
                    "_code":
"MOULD"
                }
            ]
        }
    ],
},
{
    "_qualification":
{
    "characteristic": {
        "_code":
"cheeseMaturationPeriodDes
cription"
    },
    "recordKey":
"0000.0000.RK",

```

```

    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,
          "_internalId": "720",
          "_externalId":
"\u0027fatInMilkContent\u0027"
        },
        "_code": "fatInMilkContent"
      },
      "recordKey": "0000.0000.RK",
      "parentRecordKey": "root"
    },
    "_datatype": "DECIMAL",
    "_formatPattern": "##.###",
    "order": 1,
    "_recordLang": [
      {
        "_qualification": {
          "language": {
            "_key": -1,
            "_code": "zxx"
          }
        },
        "values": [
          1724.34
        ]
      }
    ]
  },
  {
    "_qualification": {
      "characteristic": {
        "_key": {
          "_entityId": 8000,
          "_internalId": "944",
          "_externalId":
"\u0027numberOfPackagesForSerPiecesGTIN\u0027"
        },
        "_code":
"numberOfPackagesForSerPiecesGTIN"
      },
      "recordKey": "0000.0000.RK",
      "parentRecordKey": "root"
    },
    "_datatype": "INTEGER",
    "order": 1,
    "_recordLang": [
      {
        "_qualification": {
          "language": {
            "_key": -1,

```

```

    "parentRecordKey": "root"
      },
      "_datatype":
"TEXT",
      "order": 1,
      "_recordLang": [
        {
          "_qualification": {
            "language":
{
              "_code":
"chi"
            },
            "values": [
              "were well,
I hope, when you left
Londo"
            ]
          },
          {
            "_qualification":
{
              "characteristic": {
                "_code":
"rennetTypeCode"
              },
              "recordKey":
"0000.0000.RK",
              "parentRecordKey": "root"
            },
            "_datatype":
"LOOKUP",
            "order": 1,
            "_recordLang": [
              {
                "_qualification": {
                  "language":
{
                    "_code":
"zxx"
                  },
                  "values": [

```

```

        "_code": "zxx"
      }
    },
    "values": [
      711
    ]
  }
],
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId": 8000,
        "_internalId": "943",
        "_externalId":
"\u0027doPackagingMaterialContainLatex\u0027"
      },
      "_code":
"doPackagingMaterialContainLatex"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
  },
  "_datatype": "LOOKUP",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language": {
          "_key": -1,
          "_code": "zxx"
        }
      },
      "values": [
        {
          "_key": {
            "_entityId": 7300,
            "_internalId": "704@271",
            "_externalId":
"\u0027FALSE\u0027@\u0027NonBinaryLogicCodes\u00
27"
          },
          "_code": "FALSE"
        }
      ]
    }
  ],
{
  "_qualification": {
    "characteristic": {

```

```

        "_code":
"NO_RENNET"
      }
    ]
  },
  {
    "_qualification":
{
  "characteristic": {
    "_code":
"fatInMilkContent"
  },
  "recordKey":
"0000.0000.RK",
  "parentRecordKey": "root"
},
  "_datatype":
"DECIMAL",
  "_formatPattern":
"##.###",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language":
{
          "_code":
"zxx"
        }
      },
      "values": [
        1724.34
      ]
    }
  ],
  {
    "_qualification":
{
  "characteristic": {
    "_code":
"numberOfPackagesForSerPie
cesGTIN"
  },
  "recordKey":
"0000.0000.RK",

```

```

        "_key": {
            "_entityId": 8000,
            "_internalId": "812",
            "_externalId":
"\u0027isHomogenised\u0027"
        },
        "_code": "isHomogenised"
    },
    "recordKey": "0000.0000.RK",
    "parentRecordKey": "root"
},
"_datatype": "LOOKUP",
"order": 1,
"_recordLang": [
    {
        "_qualification": {
            "language": {
                "_key": -1,
                "_code": "zxx"
            }
        },
        "values": [
            {
                "_key": {
                    "_entityId": 7300,
                    "_internalId": "795@271",
                    "_externalId":
"\u0027NOT_APPLICABLE\u0027@\u0027NonBinaryLogic
Codes\u0027"
                },
                "_code": "NOT_APPLICABLE"
            }
        ]
    }
]
}
]
}
}
}
}

```

```

"parentRecordKey": "root"
    },
    "_datatype":
"INTEGER",
    "order": 1,
    "_recordLang": [
        {
            "_qualification": {
                "language":
{
                    "_code":
"zxx"
                }
            },
            "values": [
                711
            ]
        }
    ],
    {
        "_qualification":
{
            "characteristic": {
                "_code":
"doPackagingMaterialContai
nLatex"
            },
            "recordKey":
"0000.0000.RK",
            "parentRecordKey": "root"
        },
        "_datatype":
"LOOKUP",
        "order": 1,
        "_recordLang": [
            {
                "_qualification": {
                    "language":
{
                        "_code":
"zxx"
                    }
                },
                "values": [
                    {
                        "_code":

```

```

"FALSE"
    }
  ]
}
],
},
{
  "_qualification":
{
  "characteristic": {
    "_code":
"isHomogenised"
  },
  "recordKey":
"0000.0000.RK",
  "parentRecordKey": "root"
  },
  "_datatype":
"LOOKUP",
  "order": 1,
  "_recordLang": [
    {
      "_qualification": {
        "language":
{
          "_code":
"zxx"
        }
      },
      "values": [
        {
          "_code":
"NOT_APPLICABLE"
        }
      ]
    }
  ]
}
]
}
}
}

```

Characteristic Values

Characteristic values are rendered in a specialized structure in order to honor their hierarchical nature. This structure is not identical to the repository as it is fully hierarchical. This is the reason for extra meta attributes which help to not get in conflict with repository based data.

Additional Meta Attributes

- `_characteristicRecords`
Top level element for all characteristic records
- `_recordLang`
Language specific record values. In case the characteristic is not language specific, the qualification for "not language specific" is returned (-1 or xxz)
In case the value of the record is a lookup value and the `includeLabels` parameter has been set, an additional `_label` element is provided.
- `_children`
children of the current characteristic record
- `_datatype`
The characteristic data type - to be able to interpret the values even in case the characteristic is no longer in the system
- `_formatPattern`
The format pattern of the characteristic - to be able to format the values even in case the characteristic is no longer in the system

Response Examples

<code>includeIds = true</code>	<code>includeIds = false</code>
<pre>{ "_entity": "Article", "_entityItem": { "_internalId": "18@1", "_entityId": 1000, "_externalId": "\u0027supplierAID-1\u0027@\u0027MASTER\u0027" }, "_revision": { "_internalId": "1", "_entityId": 5600, "_externalId": "\u0027root\u0027" }, "_container": { "_internalId": "1", "_entityId": 2900, "_externalId": "\u0027MASTER\u0027" }, }</pre>	<pre>{ "_entity": "Article", "_entityItem": { "_entity": "Article", "_externalId": "\u0027supplierAID-1\u0027@\u0027MASTER\u0027" }, "_revision": { "_entity": "Revision", "_externalId": "\u0027root\u0027" }, "_container": { "_entity": "MasterCatalog", "_externalId": "\u0027MASTER\u0027" }, "_data": { "identifier": "supplierAID-1", "noCuperOU": 1, } }</pre>

```

"_data": {
  "identifier": "supplierAID-1",
  "noCUperOU": 1,
  "currentStatus": {
    "_key": 100,
    "_code": "NEW"
  },
  "catalog": {
    "_key": {
      "_entityId": 2900,
      "_internalId": "1",
      "_externalId":
"\u0027MASTER\u0027"
    },
    "_code": "MASTER"
  },
  "quantityMin": 1,
  "mainSupplier": {
    "_key": {
      "_entityId": 2800,
      "_internalId": "3",
      "_externalId": "\u0027Heiler
Product Manager\u0027"
    },
    "_code": "Heiler Product
Manager"
  },
  "priceQuantity": 1,
  "quantityInterval": 1,
  "kitParent": false,
  "soldOnlyInKits": false,
  "_characteristicRecords": [
    {
      "_qualification": {
        "characteristic": {
          "_key": {
            "_entityId": 8000,
            "_internalId": "2176",
            "_externalId":
"\u0027AnimalIngredient\u0027"
          },
          "_code":
"AnimalIngredient"
        },
        "recordKey": "0000.0000.RK",
        "parentRecordKey": "root"
      },
      "_datatype": "LOOKUP",
      "order": -32767,
      "_recordLang": [
        {
          "_qualification": {

```

```

"currentStatus": {
  "_code": "NEW"
},
"catalog": {
  "_code": "MASTER"
},
"quantityMin": 1,
"mainSupplier": {
  "_code": "Heiler Product
Manager"
},
"priceQuantity": 1,
"quantityInterval": 1,
"kitParent": false,
"soldOnlyInKits": false,
"_characteristicRecords": [
  {
    "_qualification": {
      "characteristic": {
        "_code":
"AnimalIngredient"
      },
      "recordKey":
"0000.0000.RK",
      "parentRecordKey": "root"
    },
    "_datatype": "LOOKUP",
    "order": -32767,
    "_recordLang": [
      {
        "_qualification": {
          "language": {
            "_code": "zxx"
          }
        },
        "values": [
          {
            "_code": "Down"
          }
        ]
      }
    ],
    "_children": [
      {
        "_qualification": {
          "characteristic": {
            "_code": "CertDown"
          },
          "recordKey":
"0000.0000.RK",
          "parentRecordKey":
"0000.0000.RK"

```



```

        "language": {
            "_key": -1,
            "_code": "zxx"
        }
    },
    "values": [
        {
            "_key": {
                "_entityId": 7300,
                "_internalId":
"6075@799",
                "_externalId":
"\u0027Down\u0027@\u0027AnimalIngredie
nt\u0027"
            },
            "_code": "Down"
        }
    ]
},
"_children": [
    {
        "_qualification": {
            "characteristic": {
                "_key": {
                    "_entityId": 8000,
                    "_internalId":
"2177",
                    "_externalId":
"\u0027CertDown\u0027"
                },
                "_code": "CertDown"
            },
            "recordKey":
"0000.0000.RK",
            "parentRecordKey":
"0000.0000.RK"
        },
        "_datatype": "LOOKUP",
        "order": -32766,
        "_recordLang": [
            {
                "_qualification": {
                    "language": {
                        "_key": -1,
                        "_code": "zxx"
                    }
                },
                "values": [
                    {
                        "_key": {
                            "_entityId":

```

```

            },
            "_datatype": "LOOKUP",
            "order": -32766,
            "_recordLang": [
                {
                    "_qualification": {
                        "language": {
                            "_code": "zxx"
                        }
                    },
                    "values": [
                        {
                            "_code": "Other"
                        }
                    ]
                }
            ],
            "_children": [
                {
                    "_qualification": {
                        "characteristic": {
                            "_code":
"CertDownExpDate"
                        },
                        "recordKey":
"0000.0000.RK",
                        "parentRecordKey":
"0000.0000.RK"
                    },
                    "_datatype": "DATE",
                    "order": -32765,
                    "_recordLang": [
                        {
                            "_qualification":
{
                                "language": {
                                    "_code":
"zxx"
                                }
                            },
                            "values": [
                                {
                                    "1977-05-28"
                                }
                            ]
                        }
                    ],
                    {
                        "_qualification": {
                            "characteristic": {
                                "_code":
"CertDownDesc"
                            },

```

```

7300,
    "_internalId":
"6077@800",
    "_externalId":
"\u0027Other\u0027@\u0027CertDown\u0027
7"
    },
    "_code": "Other"
    }
    ],
    },
    "_children": [
    {
        "_qualification": {
            "characteristic": {
                "_key": {
                    "_entityId":
8000,
                    "_internalId":
"2180",
                    "_externalId":
"\u0027CertDownExpDate\u0027"
                },
                "_code":
"CertDownExpDate"
            },
            "recordKey":
"0000.0000.RK",
            "parentRecordKey":
"0000.0000.RK"
        },
        "_datatype": "DATE",
        "order": -32765,
        "_recordLang": [
        {
            "_qualification":
{
                "language": {
                    "_key": -1,
                    "_code": "zxx"
                }
            },
            "values": [
                "1977-05-28"
            ]
        }
    ],
    },
    {
        "_qualification": {
            "characteristic": {
                "_key": {

```

```

            "recordKey":
"0000.0000.RK",
            "parentRecordKey":
"0000.0000.RK"
        },
        "_datatype": "TEXT",
        "order": -32763,
        "_recordLang": [
        {
            "_qualification":
{
                "language": {
                    "_code":
"zxx"
                }
            },
            "values": [
                "Language
Independent Description"
            ]
        }
    ],
    },
    {
        "_qualification": {
            "characteristic": {
                "_code":
"CertDownZert"
            },
            "recordKey":
"0000.0000.RK",
            "parentRecordKey":
"0000.0000.RK"
        },
        "_datatype": "MIME",
        "order": -32764,
        "_recordLang": [
        {
            "_qualification":
{
                "language": {
                    "_code":
"zxx"
                }
            },
            "values": [
                {
                    "_path": "34\
\48\\47\
\testMimeValue.c4995547687a9880_-2df8
fd04_17f9d0348a3_-7970.jpg",
                    "_label":

```

```

      "_entityId":
8000,
      "_internalId":
"2179",
      "_externalId":
"\u0027CertDownDesc\u0027"
    },
    "_code":
"CertDownDesc"
  },
  "recordKey":
"0000.0000.RK",
  "parentRecordKey":
"0000.0000.RK"
},
"_datatype": "TEXT",
"order": -32763,
"_recordLang": [
{
  "_qualification":
{
    "language": {
      "_key": -1,
      "_code": "zxx"
    }
  },
  "values": [
    "Language
Independent Description"
  ]
}
],
{
  "_qualification": {
    "characteristic": {
      "_key": {
        "_entityId":
8000,
        "_internalId":
"2178",
        "_externalId":
"\u0027CertDownZert\u0027"
      },
      "_code":
"CertDownZert"
    },
    "recordKey":
"0000.0000.RK",
    "parentRecordKey":
"0000.0000.RK"
  },

```

```

        "_datatype": "MIME",
        "order": -32764,
        "_recordLang": [
            {
                "_qualification":
{
                    "language": {
                        "_key": -1,
                        "_code": "zxx"
                    }
                },
                "values": [
                    {
                        "_path": "8\
\2\\18\
\testMimeValue.931b5ed859092878_29b7f7
65_17f9d0ab375_-796c.jpg",
                        "_label":
"testMimeValue.jpg",
                        "_type":
"image/jpeg"
                    }
                ]
            }
        ]
    }
}

```

Examples

Please see the attached postman example collection for you convenience: ObjectAPI.postman_collection.json

Get Item by ID

```

curl --location --request GET 'http://localhost:1512/rest/V2.0/object/Article/
14260@1120' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic cmVzdDpoZWlsZXI='

```

Get Item by Identifier As XML

```
curl --location --request GET 'http://localhost:1512/rest/V2.0/object/Article/'\''supplierAID-42'\''@'\''MySupplierCatalog'\'' \
--header 'Content-Type: application/xml' \
--header 'Authorization: Basic cmVzdDpoZWlsZXI='
```

Get Filtered Item (entity and qualification filter)

```
curl --location --request GET 'http://localhost:1512/rest/V2.0/object/Article/'\''supplierAID-42'\''@'\''MySupplierCatalog'\''?qualificationFilter=language(eng)&entityFilter=ArticleLang' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic cmVzdDpoZWlsZXI='
```

13.6.6.6 REST Object API Write

Provides the Create, Update and Delete calls for objects. Sub-entity deletion is considered an update of the object. Update calls will check if the given item already exists and will create it when not.

- [Create/Update Object \(see page 674\)](#)
 - [Permissions \(see page 674\)](#)
 - [Body \(see page 674\)](#)
 - [Response \(see page 681\)](#)
 - [Request Syntax \(see page 685\)](#)
- [Examples \(see page 692\)](#)
 - [Create an Item in the Master catalog \(see page 692\)](#)

Create/Update Object**Permissions****Entity Permissions**

The user needs to have the corresponding entity permission. They need to have **CREATE** in case a new item should be created and **EDIT** for any update of an existing item. In case sub-entities have separate entity permissions they need to apply too.

Object Permissions

The user needs to have the **WRITE** object permission in case of an update of an existing entity item. In case he doesn't, the api returns HTTP 403 (forbidden)

Field Permissions

The user needs to have the WRITE field permissions for all fields he wants to write.

Qualification Permissions (aka Qualified Field Permissions)

The user needs to have the WRITE permission for a qualification, e.g. for Language = English. If he doesn't, sub-entities which are qualified for this will not be persisted, and an error is returned in the protocol

Body

The structure of the body for the object api is similar, no matter if a post or put is executed. With one exception:

The attribute which represents the external, alphanumeric identifier can be omitted in case of the create api, as the framework will generate a new identifier automatically.

Fields and Entities

The [repository property "shortIdentifier"](#) (see [page 79](#)) in the custom section is used to define the entity and field names for the json/xml structure. The short identifier is unique within its parent entity only!

The standard repository already contains a short identifier for all fields and enabled qualifications.

Meta attributes

All meta attributes are prefixed with an underscore. Here is a list of the meta attributes used in the body besides the ones already described in the [datatypes](#) (see [page 676](#)) section.

<code>_qualification</code>	The qualification of the record. The combination of the qualification values in the qualification object define this record as unique within its parent.
-----------------------------	---

<div>_changeType</div>	<p>Optional attribute which defines how a sub-entity record should be handled.</p> <table border="1"> <tr> <td data-bbox="367 358 636 530"> <div>put</div> </td><td data-bbox="636 358 1425 530"> <p>The sub-entity record is updated in case it already exists. It will be created if it doesn't exist. This is the default mode in case this attribute is missing</p> </td></tr> <tr> <td data-bbox="367 530 636 734"> <div>remove</div> </td><td data-bbox="636 530 1425 734"> <p>The sub-entity record is removed in case it exists. If not, nothing happens. To uniquely identify the record to be removed, the qualification element is needed. Other attributes can be omitted in this case.</p> </td></tr> </table>	<div>put</div>	<p>The sub-entity record is updated in case it already exists. It will be created if it doesn't exist. This is the default mode in case this attribute is missing</p>	<div>remove</div>	<p>The sub-entity record is removed in case it exists. If not, nothing happens. To uniquely identify the record to be removed, the qualification element is needed. Other attributes can be omitted in this case.</p>
<div>put</div>	<p>The sub-entity record is updated in case it already exists. It will be created if it doesn't exist. This is the default mode in case this attribute is missing</p>				
<div>remove</div>	<p>The sub-entity record is removed in case it exists. If not, nothing happens. To uniquely identify the record to be removed, the qualification element is needed. Other attributes can be omitted in this case.</p>				

Datatypes

ENTITY_ITEM	<p>Objects can be specified by either the <code>_internalId</code> or the <code>_externalId</code> attribute. In case both is given, the <code>_internalId</code> takes precedence due to performance reasons, there is no fallback logic to the <code>_externalId</code>.</p> <p>In case the entity item can not be resolved an error is returned.</p> <p>Please note the Syntax (don't overlook the single quotes for external id!):</p> <p>Entity Items with container:</p> <pre> _externalId: 'Identifier'@'ContainerIdentifier' _internalId: InternalID@ContainerInternalID </pre> <p>Entity Items without container:</p> <pre> _externalId: 'Identifier' _internalId: InternalID </pre> <p>Single quotes as part of the identifiers need to be escaped with a backslash. Like</p> <pre>'Iden\tifier'@'ContainerIdentifir'</pre> <p>Please note that quotes, backslashes or the @ symbol should be escaped in URLs. Please take a look at the REST Datatypes (see page 436) page for more information.</p>	<p>Entity item with the external id:</p> <pre>"catalog": { "_externalId": "'MY_SUPPLIER_CATALOG'" }</pre> <p>Entity item with the internal id:</p> <pre>"catalog": { "_internalId": "1120" }</pre>
--------------------	---	--

MIME_
VALUE

A reference to a mime file which must be provided in a separate zip archive. A mime value has a label. The label is shown in the UI in case a preview picture is not used there. The file path defines the location of the file inside of the zip archive when you download or upload the mime value.

The label and type are optional in case you want to write a mime value. They will be extracted from the `filePath` if omitted. Please note that the relative file path of a mime value will change once it's uploaded and saved as Product 360 makes sure that every mime value is unique in the system. If you retrieve the mime value again after the upload, the `path` will be different. It should always be treated as a black box.

MIME_VALUE object, which at least must contain the relative `_filePath`.

The file path must correspond to the file location within the uploaded archive file.

```
{
  "_label": "Example Picture",
  "_type": "image/jpg",
  "_path":
    "mySubFolderInTheZip\\examplePicture.jpg"
}
```

<code>_label</code>	Optional: Human readable label for the mime value. Defaults to the filename in the path attribute (without extension)
<code>_type</code>	Optional: The official mime type of the file. If omitted, the mime type will be determined based on the file extension using the mapping provided in the <code>mime.types</code> file in the Product 360 configuration folder.
<code>_path</code>	Mandatory: File path to the mime file. The path is relative to the ZIP Archive which has been uploaded prior to the REST call with the REST File API (see page 738).

times tamp	ISO 8601 (YYYY-MM-DDTHH:mm:ss.sss)	2012-04-23T18:25:43.511
date	ISO 8601 (YYYY-MM-DD)	2019-07-04
time	ISO 8601 (HH:mm:ss.sss)	18:25:43.511
numbe rs	<p>Standard JSON. A dot is the decimal separator, no thousand separators</p> <p>Rounding logic for decimal fields:</p> <p>The number of fraction digits is defined in the repository (see page 79) with the scale attribute. In case the given value has a higher precision (a higher scale) the API will apply a rounding algorithm to the value to bring it to the maximum scale which is defined in the repository.</p> <p><i>The rounding algorithm used is HALF_UP:</i> Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up. If the discarded fraction is ≥ 0.5 it's rounded up; otherwise, down. Note that this is the rounding mode commonly taught at school.</p> <p>For more details on this rounding mode you can have a look at the Java documentation for it.</p>	123.45
text	Standard JSON	"My Text"
boole an	Standard JSON, true/false	true false

Enumeration Fields

Enumeration fields can have `_key` or `_code` or `_label`. Always only one of the three possibilities is considered, even if multiples are provided. Key is used before code, and code is used before label.

In case the label must be used, the HTTP requests locale is used to interpret it. For performance reasons we strongly recommend to use `_key` or `_code`.

In case the enumeration field is of datatype ENTITY_ITEM, then the `_key` must be provided with the ENTITY_ITEM syntax (see above)

No Value / Empty / Null

If an attribute is given with `null`, `null` will be applied to the entity item's field. An empty string will be treated as null for text fields.

If an attribute is **not part of the document, it will be ignored**. In case the attribute is defined as mandatory, a corresponding error will be returned in the protocol

Multi-Value fields

Single fields which are defined as multi-value in the repository (`upperBound>1`) must be provided as array. It is not possible to add or remove a single value. The existing values will be replaced with the given ones.

Mime-Value fields

To be able to set mime values you need to first upload the binaries in form of a ZIP archive which contains all files which are referenced from the JSON payload. If the file is missing, a corresponding error is returned.

The file can contain folders and sub folders, however, the JSON payload needs to include the full path to the picture within the archive.

See the [REST File API \(see page 738\)](#) for details on how to upload the archive.

A single archive file can be used for multiple items. It will automatically be deleted 24 hours after it has been uploaded.

Examples

Example which uses only external ids for ENTITY_ITEM fields or the code for enumeration fields

This is the typical scenario in which the client of this api has no deeper knowledge on the internal id's or keys of entities and enumeration values.

```
{
  "identifier": "identifier-1",
  "manufacturerAID": "A0000000000000",
  "manufacturerName": "BMW",
  "catalog": {
```

```

    "_code": "MySupplierCatalog"
  },
  "orderUnit": {
    "_code": "C62"
  },
  "lang": [
    {
      "_qualification": {
        "language": {
          "_code": "deu"
        }
      },
      "keywords": [
        "shift"
      ],
      "segment": "be treating his father's favourite in such a manner, one
whom",
      "manufacturerTypeDescription": "of him. Proud tha",
      "descriptionShort": "feel he had been wrong; he had liberality, and he had
the mean",
      "remarks": "be at present; that you actually declared your resolution of
never taking orders"
    }
  ]
}

```

Response

The response contains useful meta attributes to easily identify the newly created entity item. The entity as well as the new identifier and the container it has been created in as well as the ready to use ENTITY_ITEM syntax.

Meta attributes

<code>_status</code>	The http status code of the response
<code>_entity</code>	The root entity identifier
<code>_entityItem</code>	The entity item in the ENTITY_ITEM syntax

<code>_identifier</code>	The external, alphanumeric identifier of the entity item. Although this identifier is already part of the <code>_entityItem</code> attribute, it's added separately for your convenience. The identifier is unique within the container
<code>_container</code>	The container of the entity item in the ENTITY_ITEM syntax. For items, products and variants it's the catalog, for structure groups it is the structure system, etc. Not all entities do have a containers.
<code>_protocol</code>	An array of protocol entries which provide details in case the validation returned errors or warnings. See below for details

```
{
  "_entity": "Article",
  "_entityItem": {
    "_internalId": "13989@1120",
    "_entityId": 1000,
    "_externalId": "\u0027NewItem\u0027@\u0027MY_SUPPLIER_CATALOG\u0027"
  },
  "_identifier": "NewItem",
  "_container": {
    "_internalId": "1120",
    "_entityId": 7000,
    "_externalId": "\u0027MY_SUPPLIER_CATALOG\u0027"
  },
  "_protocol": {
    "infoCounter": 0,
    "warningCounter": 0,
    "errorCounter": 1,
    "entries": [
      {
        "severity": "ERROR",
        "category": "DATATYPE",
        "property": "Article.OrderUnit",
        "message": "\u0027orderUnit\u0027 (Article.OrderUnit): Code value \u0027NoIdea\u0027 not found in enumeration \u0027Enum.OrderUnits\u0027. Either the code is not part of the enumeration or the user has no read permission"
      }
    ]
  }
}
```

Protocol

<code>infoCounter</code>	Number of entries with INFO severity
--------------------------	--------------------------------------

<code>warningCounter</code>	Number of entries with WARNING severity
<code>errorCounter</code>	Number of entries with ERROR severity
<code>entries</code>	Array of protocol entries

Protocol Entry

<code>severity</code>	INFO, WARNING or ERROR
-----------------------	------------------------

category	<p>The category of the protocol entry. Possible categories are:</p> <table border="1"> <tr> <td data-bbox="526 389 831 510">SYSTEM</td><td data-bbox="831 389 1425 510">Category for general system errors</td></tr> <tr> <td data-bbox="526 510 831 651">UNIQUENESS</td><td data-bbox="831 510 1425 651">Category for identity checks, unique constraints etc.</td></tr> <tr> <td data-bbox="526 651 831 792">DATATYPE</td><td data-bbox="831 651 1425 792">Category for data type conversion problems as well as parsing issues</td></tr> <tr> <td data-bbox="526 792 831 960">CARDINALITY</td><td data-bbox="831 792 1425 960">Category for problems with mandatory fields/elements as well as min/max number of elements (e.g. keywords)</td></tr> <tr> <td data-bbox="526 960 831 1104">RANGE</td><td data-bbox="831 960 1425 1104">Category for min-/max length or value range violations</td></tr> <tr> <td data-bbox="526 1104 831 1227">CONSISTENCY</td><td data-bbox="831 1104 1425 1227">Category for general consistency checks</td></tr> <tr> <td data-bbox="526 1227 831 1368">SECURITY</td><td data-bbox="831 1227 1425 1368">Category for any kind of permission/security problems</td></tr> <tr> <td data-bbox="526 1368 831 1489">NOTE</td><td data-bbox="831 1368 1425 1489">Category for info notes</td></tr> <tr> <td data-bbox="526 1489 831 1610">SUMMARY</td><td data-bbox="831 1489 1425 1610">Category for summary entries</td></tr> </table>	SYSTEM	Category for general system errors	UNIQUENESS	Category for identity checks, unique constraints etc.	DATATYPE	Category for data type conversion problems as well as parsing issues	CARDINALITY	Category for problems with mandatory fields/elements as well as min/max number of elements (e.g. keywords)	RANGE	Category for min-/max length or value range violations	CONSISTENCY	Category for general consistency checks	SECURITY	Category for any kind of permission/security problems	NOTE	Category for info notes	SUMMARY	Category for summary entries
SYSTEM	Category for general system errors																		
UNIQUENESS	Category for identity checks, unique constraints etc.																		
DATATYPE	Category for data type conversion problems as well as parsing issues																		
CARDINALITY	Category for problems with mandatory fields/elements as well as min/max number of elements (e.g. keywords)																		
RANGE	Category for min-/max length or value range violations																		
CONSISTENCY	Category for general consistency checks																		
SECURITY	Category for any kind of permission/security problems																		
NOTE	Category for info notes																		
SUMMARY	Category for summary entries																		
property	<p>Some form of unique identifier of the field or entity. Might be a combination of field type identifier and entity identifier or, like in this example, the field identifier. In any case it uniquely identifies the repository element</p>																		

message	A human readable message in the locale of the request
---------	---

```
{
  "_protocol": {
    "infoCounter": 0,
    "warningCounter": 0,
    "errorCounter": 1,
    "entries": [
      {
        "severity": "ERROR",
        "category": "DATATYPE",
        "property": "Article.OrderUnit",
        "message": "\u0027orderUnit\u0027 (Article.OrderUnit): Code value
\u0027NoIdea\u0027 not found in enumeration \u0027Enum.OrderUnits\u0027. Either
the code is not part of the enumeration or the user has no read permission"
      }
    ]
  }
}
```

Request Syntax

Create Object

Creates a new object of the given entity. In case the payload contains the corresponding external identifier field, this identifier will be used for the object.

In case no identifier is given, the system will automatically create one for the new object.

If the entity requires a container (e.g. for Items the container is the catalog), the corresponding container field must also be part of the payload or provided as query parameter.

URL Pattern	/object/{entity-identifier}
Method	POST
Content Type (for the body)	application/json (recommended!) application/xml
Accept (for the response)	application/json (recommended) application/xml

URL Parameters	<div><div>{entity-identifier}</div></div>		Unique identifier of the repository entity																
Query Parameter	<table><tr><th>Name</th><th>Datatype</th><th>Default</th><th>Description</th></tr><tr><td><div><div>updateIfExists</div><div>ts</div></div></td><td><div><div>boolean</div></div></td><td><div><div>false</div></div></td><td>Based on this parameter an already existing entity item will be updated. In order for this to work, the identifier and container (if the entity requires one) must be part of the payload.</td></tr><tr><td><div><div>mimeValueArchive</div></div></td><td><div><div>String</div></div></td><td></td><td>The unique ID of the uploaded mime archive file as returned from the REST File API (see page 738)</td></tr><tr><td><div><div>container</div></div></td><td><div><div>String</div></div></td><td></td><td>Optional parameter which will set or override the container as it is defined in the JSON payload. In case of an Item, this would be a ENTITY_ITEM Syntax of a catalog, as the Catalog is the container entity of Item. This parameter is especially useful in testing scenarios in which the same payload needs to be processed repeatedly for different containers.</td></tr></table>			Name	Datatype	Default	Description	<div><div>updateIfExists</div><div>ts</div></div>	<div><div>boolean</div></div>	<div><div>false</div></div>	Based on this parameter an already existing entity item will be updated. In order for this to work, the identifier and container (if the entity requires one) must be part of the payload.	<div><div>mimeValueArchive</div></div>	<div><div>String</div></div>		The unique ID of the uploaded mime archive file as returned from the REST File API (see page 738)	<div><div>container</div></div>	<div><div>String</div></div>		Optional parameter which will set or override the container as it is defined in the JSON payload. In case of an Item, this would be a ENTITY_ITEM Syntax of a catalog, as the Catalog is the container entity of Item. This parameter is especially useful in testing scenarios in which the same payload needs to be processed repeatedly for different containers.
Name	Datatype	Default	Description																
<div><div>updateIfExists</div><div>ts</div></div>	<div><div>boolean</div></div>	<div><div>false</div></div>	Based on this parameter an already existing entity item will be updated. In order for this to work, the identifier and container (if the entity requires one) must be part of the payload.																
<div><div>mimeValueArchive</div></div>	<div><div>String</div></div>		The unique ID of the uploaded mime archive file as returned from the REST File API (see page 738)																
<div><div>container</div></div>	<div><div>String</div></div>		Optional parameter which will set or override the container as it is defined in the JSON payload. In case of an Item, this would be a ENTITY_ITEM Syntax of a catalog, as the Catalog is the container entity of Item. This parameter is especially useful in testing scenarios in which the same payload needs to be processed repeatedly for different containers.																
Example	<div><div>/rest/V2.0/object/Article</div></div>																		

Return Codes	Code	Reason
	200 (Ok)	Entity item has been updated successfully. (only in case the <code>updateIfExists</code> parameter is true and the unique identifier of the object has been provided in the payload)
	201 (Created)	Entity Item has been created successfully
	400 (Bad Request)	Entity item has not been created/updated successfully due to at least one validation error. The validation protocol is part of the response
	403 (Forbidden)	User has no create permission for this entity
	404 (Not Found)	Entity has not been found
	429 (Too many Requests)	<p>The number of concurrent write calls is too high. Clients should slow down their requests or use the message queue interface instead (recommended).</p> <p>Details on thread pool configuration for the Object API can be found here: REST Object API (see page 592)</p>
	500	Entity item has not been created/updated due to an unexpected server error. The server log should be checked.
	503 (Service unavailable)	In case the execution has been interrupted or cancelled unexpectedly any currently waiting executions will return server unavailable. This is typically when the server is shutdown forcefully while there are still executions waiting.

Update Object

An existing entity item can be updated with this endpoint.

URL Pattern	/object/{entity-identifier}/{entityItem}	
Method	PUT	
Content Type (for the body!)	application/json (recommended) application/xml	
Accept (for the response!)	application/json (recommended) application/xml	
URL Parameters	{entity-identifier}	Unique identifier of the repository entity
	{entityItem}	The ENTITY_ITEM either in the internalId or externalId syntax. E.g. 'myItem'@'MASTER' or 42@1

Query Parameters	Name	Datatype	Default	Description
	<code>createIfMissing</code>	<code>boolean</code>	<code>true</code>	Based on this parameter a missing entity item will be created with the given identifier. In order for this to work, the <code>externalId</code> must be used for the <code>entityItem</code> parameter. In case the <code>internalId</code> is used and the object is not in the system a 404 code will be returned.
	<code>mimeValueArchive</code>	<code>String</code>		The unique ID of the uploaded mime archive file as returned from the REST File API (see page 738)
Example	<pre>/rest/V2.0/object/Article/'myItem'@'MASTER'</pre> <pre>/rest/V2.0/object/Article/42@1</pre>			

Return Codes	Code	Reason
	200 (Ok)	Entity item has been updated successfully
	201 (Created)	Entity item has been created successfully
	400 (Bad Request)	Entity item has not been created/updated successfully due to at least one validation error
	403 (Forbidden)	User has no create or edit permission for this entity or object. Object permission is only checked in case of update
	404 (Not Found)	<ul style="list-style-type: none"> Entity has not been found Entity item has not been found. Only in case the <code>internalId</code> is used for the <code>entityItem</code> parameter, or the <code>createIfMissing</code> parameter is set to <code>false</code>.
	429 (Too many Requests)	<p>The number of concurrent write calls is too high. Clients should slow down their requests or use the message queue interface instead (recommended).</p> <p>Details on thread pool configuration for the Object API can be found here: REST Object API (see page 592)</p>
	500	Entity item has not been created/updated due to an unexpected server error. The server log should be checked.

	Code	Reason
	503 (Service unavailable)	In case the execution has been interrupted or cancelled unexpectedly any currently waiting executions will return server unavailable. This is typically when the server is shutdown forcefully while there are still executions waiting.

Delete Object

This API is essentially performing the same deletion as the [REST List API Delete \(see page 539\)](#). In case many items should be deleted, always use the List API. Executing the Object API's delete method for many items is causing more load on the system than executing the [REST List API Delete \(see page 539\)](#) just once because the application can not optimize the database requests.

URL Pattern	/object/{entity-identifier}/{entityItem}	
Method	DELETE	
Content Type	none	
Accept Type	none	
URL Parameters	{entity-identifier}	Unique identifier of the repository entity
	{entityItem}	The ENTITY_ITEM either in the <code>internalId</code> or <code>externalId</code> syntax. E.g. <code>'myItem'@'MASTER'</code> or <code>42@1</code>

Example	<pre>/rest/V2.0/object/Article/'myItem'@'MASTER'</pre> <pre>/rest/V2.0/object/Article/42@1</pre>	
Return Codes	Code	Reason
	200 (Ok)	Entity item has been deleted successfully
	404 (Not Found)	Entity item has not been found
	403 (Forbidden)	User has no delete permission for this entity or object
	500	Entity item has not been deleted due to an unexpected server error. The server log should be checked.

Examples

Create an Item in the Master catalog

This call will create a new item in the master catalog with an auto generated identifier. This is the most minimal call to create a new item. It's of course always better to directly provide all data the item should have.

Request

```
curl --location --request POST 'http://localhost:1512/rest/V2.0/object/Article' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic cmVzdDpoZWlsZXI=' \
--data-raw '{
  "catalog": {
    "_key": {
      "_externalId": "'\''MASTER'\''"
    }
  }
}'
```

Response

```
{
  "_entity": "Article",
  "_entityItem": {
    "_internalId": "5@1",
    "_entityId": 1000,
    "_externalId": "'Article_1643730037713001'@'MASTER'"
  },
  "_identifier": "Article_1643730037713001",
  "_container": {
    "_internalId": "1",
    "_entityId": 2900,
    "_externalId": "'MASTER'"
  },
  "_protocol": {
    "infoCounter": 0,
    "warningCounter": 0,
    "errorCounter": 0,
    "entries": []
  }
}
```

Payload Examples

Examples are usually better than most other documentations. This page will provide many hands-on examples. In order to have it easier to understand, all examples are written in JSON - but work likewise as XML.

The xml syntax is identical to the JSON one. Please have also a look at the attached Article_00.xml file which contains a full example of a write payload in the XML format. See also section Full Example.

Create or Update

The payload for create and update is essentially the same. The only difference is how an already existing item is handled. In case of a create call, existing items will return a corresponding error, in case of an update they will be updated.

There are query parameters to change this behavior for update and create. Please see the request definitions on [REST Object API Write \(see page 674\)](#).

General Examples

This section shows some general examples which basically work for all entities which are defined in the repository. For ease of understanding the examples are also made with the "Item" entity, but as mentioned, a generic by nature.

Simple fields on the root level of the entity

This example has no identifier, so the identifier will be generated in case it's used for create. In case it's used for update, the identifier needs to be provided in the request URL anyway.

```

1  {
2    "catalog": {
3      "_key": {
4        "_externalId": "'ObjectApiWriterTest'"
5      }
6    },
7    "gtin": "123456789",
8    "orderUnit": {
9      "_code": "C62"
10   },
11   "deliveryTime": 42.42
12 }

```

Remove a sub-entity record

This example will remove the english language sub-entity record

```

1  {
2    "lang": [
3      {
4        "_changeType": "remove",
5        "_qualification": {
6          "language": {
7            "_code": "en"
8          }
9        }
10   }
11 ]
12 }

```

Item, Product, Variants

Create prices for an item

The prices sub-entity has a small specialty. Contrary to the repository/meta definition, the object api does not have the "validAt" qualification. The object api used the `validFrom` and `validTo` qualifications instead.

This example creates an item with three purchase price records

```

1  {
2    "pricePurchase": [
3      {

```

```

4      "_qualification": {
5          "supplier": {
6              "_code": "MAIN_SUPPLIER"
7          },
8          "type": {
9              "_code": "net_list"
10         },
11         "currency": {
12             "_code": "EUR"
13         },
14         "territory": {
15             "_code": "DE"
16         },
17         "validFrom": "2021-01-01",
18         "validTo": "2021-05-28"
19     },
20     "value": [
21         {
22             "_qualification": {
23                 "lowerBound": 1.0
24             },
25             "amount": 11.11
26         }
27     ]
28 },
29 {
30     "_qualification": {
31         "supplier": {
32             "_code": "MAIN_SUPPLIER"
33         },
34         "type": {
35             "_code": "net_list"
36         },
37         "currency": {
38             "_code": "EUR"
39         },
40         "territory": {
41             "_code": "DE"
42         },
43         "validFrom": "2021-05-29",
44         "validTo": "2021-09-30"
45     },
46     "value": [
47         {
48             "_qualification": {
49                 "lowerBound": 1.0
50             },
51             "amount": 22.22
52         }
53     ]
54 },
55 {
56     "_qualification": {

```

```

57     "supplier": {
58         "_code": "MAIN_SUPPLIER"
59     },
60     "type": {
61         "_code": "net_list"
62     },
63     "currency": {
64         "_code": "EUR"
65     },
66     "territory": {
67         "_code": "DE"
68     },
69     "validFrom": "2021-10-01",
70     "validTo": "2021-12-31"
71 },
72 "value": [
73     {
74         "_qualification": {
75             "lowerBound": 1.0
76         },
77         "amount": 33.33
78     }
79 ]
80 }
81 ]
82 }

```

Update a single price

To update the amount of a single price, you need to specify the full list of qualifications in order to uniquely identify the record to update.

```

1  {
2      "pricePurchase": [
3          {
4              "_qualification": {
5                  "supplier": {
6                      "_code": "MAIN_SUPPLIER"
7                  },
8                  "type": {
9                      "_code": "net_list"
10                 },
11                 "currency": {
12                     "_code": "EUR"
13                 },
14                 "territory": {
15                     "_code": "DE"
16                 },
17                 "validFrom": "2021-05-29",
18                 "validTo": "2021-09-30"
19             },

```

```

20     "value": [
21       {
22         "_qualification": {
23           "lowerBound": 1.0
24         },
25         "amount": 44.44
26       }
27     ]
28   }
29 ]
30 }

```

Add Attribute with multiple values

Adds a new attribute "Material" with String as datatype and adds three values for it.

```

1  {
2    "attribute": [
3      {
4        "_qualification": {
5          "nameInKeyLang": "Material"
6        },
7        "datatype": {
8          "_code": "String"
9        },
10       "multiValue": true,
11       "mandatory": false,
12       "value": [
13         {
14           "_qualification": {
15             "language": {
16               "_code": "eng"
17             },
18             "identifier": "SECOND"
19           },
20           "value": "Steel"
21         },
22         {
23           "_qualification": {
24             "language": {
25               "_code": "eng"
26             },
27             "identifier": "THIRD"
28           },
29           "value": "Copper"
30         },
31         {
32           "_qualification": {
33             "language": {
34               "_code": "eng"
35             },

```

```

36         "identifier": "DEFAULT"
37     },
38     "value": "Wood"
39 }
40 ]
41 }
42 ]
43 }

```

Remove item attribute value

Removes a single value from a multi-value item attribute

```

{
  "attribute": [
    {
      "_qualification": {
        "nameInKeyLang": "Material"
      },
      "value": [
        {
          "_qualification": {
            "language": {
              "_code": "eng"
            },
            "identifier": "SECOND"
          },
          "_changeType": "remove"
        }
      ]
    }
  ]
}

```

Remove item attribute (with all values)

```

{
  "attribute": [
    {
      "_qualification": {
        "nameInKeyLang": "Material"
      },
      "_changeType": "remove"
    }
  ]
}

```

Assign an item to a structure group

The ArticleStructureMap sub-entity is not available in the object api. It's successor, the ArticleStructureGroupMap is. To make it easier for everyone, we omit the Structure as a qualification field. It would have been redundant since the StructureGroup is already a qualification field.

This element assigns the item, product or variant to "MyNode" within the "MyStructure" structure system. Please note that the StructureGroup field has no enumeration, therefore we can not use the code syntax for it but must use the entity_items

```

1  {
2    "structureGroupMap": [
3      {
4        "_qualification": {
5          "structureGroup": {
6            "_externalId": "'MyNode'@'MyStructure'"
7          }
8        }
9      }
10   ]
11  }
```

You can also combine this with `"_changeType"="remove"` which would remove the assignment.

Set Attribute Values (based on Structure Attributes)

In case the structure is a maintenance structure, the structure can define attributes which then can have actual values on the item level. Here is an example how to set the Attribute value with a mapping to the structure feature.

```

1  {
2    "attribute": [
3      {
4        "_qualification": {
5          "nameInKeyLang": "Product name"
6        },
7        "identifier": "BAA316003",
8        "datatype": {
9          "_code": "String"
10       },
11       "value": [
12         {
13           "_qualification": {
14             "language": {
15               "_code": "eng"
```

```

16         },
17         "identifier": "DEFAULT"
18     },
19     "value": "spirits. Mrs. Bennet invited him to dine with
them; but, with many expres"
20     }
21 ],
22 "structureGroupAttributeMap": [
23     {
24         "_qualification": {
25             "structureGroup": {
26                 "_entity": "StructureGroup",
27                 "_externalId": "'AKL837004'@'ECLASS-6.1'"
28             }
29         },
30         "structureAttribute": {
31             "_key": {
32                 "_entity": "StructureFeature",
33                 "_externalId": "'BAA316003'@'ECLASS-6.1'"
34             }
35         }
36     }
37 ]
38 }
39 }

```

Add characteristic records with child records

This example creates/updates an item's characteristic values.

The characteristic model which has been used looks roughly like this:

AnimalIngredients (Characteristic, Root)

```

|
|-- CertificateDown (1:1 for AnimalIngredient: DOWN)
|   |-- CertDownZert (1:1 for CertificateDown: OTHER)
|   |-- CertDownDesc (1:1 for CertificateDown: OTHER)
|   |-- CertDownExpDate (1:1 for CertificateDown: OTHER)

```

Characteristic values are hierarchical by nature, but the repository model for them is different. The normal generic logic can not be used here as it would be way too complicated.

- The `recordKey` qualification can be omitted in case there is only one record for this characteristic. In this case the default (0000.0000.RK) is used.
- The `parentRecordKey` can always be omitted as the parent is defined by the hierarchical structure itself

- The `language` qualification can be omitted in case the value is not language dependent. (In case you want to provide it anyway, please use `zxx` as language code (this means "language independent"))
- All validations which are defined in the validation expressions are executed and will be reported in the result object!

```

1  {
2    "_characteristicRecords": [
3      {
4        "_qualification": {
5          "characteristic": {
6            "_code": "AnimalIngredient"
7          }
8        },
9        "_recordLang": [
10       {
11         "values": [
12           {
13             "_code": "Down"
14           }
15         ]
16       }
17     ],
18     "_children": [
19       {
20         "_qualification": {
21           "characteristic": {
22             "_code": "CertDown"
23           }
24         },
25         "_recordLang": [
26           {
27             "values": [
28               {
29                 "_code": "Other"
30               }
31             ]
32           }
33         ],
34         "_children": [
35           {
36             "_qualification": {
37               "characteristic": {
38                 "_code": "CertDownExpDate"
39               }
40             },
41             "_recordLang": [
42               {
43                 "values": [
44                   "1977-05-28"
45                 ]
46               }

```



```

47         ]
48     },
49     {
50         "_qualification": {
51             "characteristic": {
52                 "_code": "CertDownZert"
53             }
54         },
55         "_recordLang": [
56             {
57                 "values": [
58                     {
59                         "_path": "archiveSubFolder/testMimeType.jpg"
60                     }
61                 ]
62             }
63         ]
64     },
65     {
66         "_qualification": {
67             "characteristic": {
68                 "_code": "CertDownDesc"
69             }
70         },
71         "_recordLang": [
72             {
73                 "values": [
74                     "Language Independent Description"
75                 ]
76             }
77         ]
78     }
79 ]
80 }
81 ]
82 }
83 ]
84 }

```

Remove a root characteristic record

To remove a characteristic record of an item, use the `_changeType` attribute. It will remove also all children of this record.

```

1  {
2      "catalog": {
3          "_code": "ObjectApiWriterTest"
4      },
5      "_characteristicRecords": [
6          {

```

```

7      "_changeType" : "remove",
8      "_qualification": {
9          "characteristic": {
10             "_code": "AnimalIngredient"
11         },
12         "recordKey": "0000.0000.RK"
13     }
14 }
15 ]
16 }

```

Structures

Create a new child node below the "MyParent" node.

The structure for groups is the same as the catalog for items. Their container object. It can be omitted depending on how the object api is called. We left it in this example for educational purposes.

```

1  {
2      "structure": {
3          "_entity": "Structure",
4          "_externalId": "'MyStructure'"
5      },
6      "parent": {
7          "_entity": "StructureGroup",
8          "_externalId": "'MyParent'@'MyStructure'"
9      }
10 }

```

Full Example

This example contains a value for pretty much all fields which we have defined in the repository, this includes GDSN fields. However, as the values are artificially generated, they do not make too much sense, but at least persisting this object will not lead to an error (assuming all dependencies are met).

It's merely here to give a big example as how the object api looks like. We have the same data created once as JSON document, and once as XML document. The xml document has a root element named <_data>, json obviously has no name of the root object.

Full Example in XML: Article_00.xml

Full Example in JSON: Article_00.json

13.6.7 REST Management API

The [REST Management API \(see page 703\)](#) provides access to control the server side management processes like Import, Merge, Export, and others. Typically these processes will be able to be started or scheduled, the process status can be evaluated and the process results can be requested and accessed.

- [REST Assortment API \(see page 703\)](#) — The Assortment API combines all methods around assortments. Currently the retrieval of the content of an assortment is possible as well as updating it's content by reevaluating dynamic rules. A modification of assortments rules is not yet possible using the Service API.
- [REST Data Quality API \(see page 705\)](#) — The Rest Data Quality API allows to schedule data quality jobs.
- [REST Environment API \(see page 721\)](#)
- [REST Export API \(see page 729\)](#) — The Rest Export API allows to schedule new export jobs and to cancel running export jobs.
- [REST File API \(see page 738\)](#) — The Rest File API can be used for uploading and downloading files to the Product Manager Server, e.g. for providing the source data of an import or for downloading the mime values of other api's.
- [REST Import API \(see page 741\)](#) — The Rest Import API allows to schedule new import jobs and to cancel running import jobs.
- [REST KPI API \(see page 747\)](#) — The Rest KPI API provides some management aspect features, e.g. to schedule jobs to calculate and persist key performance indicator values or delete certain persisted KPI results.
- [REST Merge API \(see page 755\)](#) — The Rest Merge API allows to schedule merge jobs.
- [REST Revision API \(see page 769\)](#) — The Rest Revision API allows to create revisions and add entity items to them (this process is called "release to a revision" or a revision release job. The API is available with Version 8.1.0.01 of the product
- [REST System API \(see page 777\)](#)
- [REST Task API \(see page 783\)](#) — The Rest Task API supports the creation, updating and content setting of PIM tasks.
- [REST Workflow API \(see page 796\)](#) — The Rest Workflow API allows to manage workflows, workflow status and process status. This API is used for the Informatica BPM workflow engine integration.

13.6.7.1 REST Assortment API

The Assortment API combines all methods around assortments. Currently the retrieval of the content of an assortment is possible as well as updating it's content by reevaluating dynamic rules. A modification of assortments rules is not yet possible using the Service API.

- [Get assortment content \(see page 703\)](#)

Get assortment content

URL Pattern	/manage/assortment/{assortment-id}/content
Method	GET

Parameters	<code>updateAssortment</code> : update the assortment if set to true, by default - false.
Content types	-
Media types	application/json
Result	report result object

Returns report result object, which contains all information from internal report result + entity identifier.

If `updateAssortment` is set, corresponding assortment will be updated (which can be time consuming) before returning the result.

13.6.7.2 REST Data Quality API

The Rest Data Quality API allows to schedule data quality jobs.

- [Rule Execution \(since 8.0.03\) \(see page 705\)](#)
 - [Content \(see page 705\)](#)
 - [Result \(see page 707\)](#)
 - [Examples \(see page 708\)](#)
- [Schedule Rule Execution \(see page 711\)](#)
 - [Query Parameters \(see page 711\)](#)
 - [Content \(see page 713\)](#)
 - [Result \(see page 713\)](#)
 - [Workflow Callback \(since 8.0.03\) \(see page 714\)](#)
 - [Workflow Callback \(since 10.0\) \(see page 714\)](#)
- [Examples \(see page 716\)](#)
 - [Executing a DQ checks for the catalog TOOLS \(see page 716\)](#)
- [Deprecated \(see page 717\)](#)
 - [Execute a data quality rule configuration group immediately \(see page 718\)](#)
 - [Execute a set of data quality rule configurations \(see page 719\)](#)
 - [Execute channel's set of data quality rule configurations \(see page 719\)](#)
 - [Examples \(see page 720\)](#)

Rule Execution (since 8.0.03)

Executes all specified rules for a given set of items. The rules to be executed can be defined either by a list of single rule configurations, rule configuration groups, channels or a combination of all options. Please note that this is a synchronous call. No job will be triggered and the call will return as soon as the execution is finished. It's not recommended to use this method for large data sets as there might be HTTP based connection issues during the call in case it takes too long. On the contrary, it definitely is recommended to use

this method for small data sets as it does not impose the overhead which comes with the job framework and it's easier to handle (especially in workflow situations)

URL Pattern	/manage/dataquality/executions
Method	POST
Content types	application/json, application/xml
Media types	application/json, application/xml
Result	A rule execution result object containing the protocol of the execution as well as the detail status information for each item and rule

Content

The content has to be a DataQualityProfile JSON object which is described in the following table

Field	Required	Default	Datatype	Parameter description
rules	no		String Array	Optional parameter which specifies individual rule names in form of a string array. Either <code>rules</code> , <code>ruleGroups</code> , <code>channels</code> or a combination of them must be specified!
ruleGroups	no		String Array	Optional parameter which specifies rule group names in form of a string array. Either <code>rules</code> , <code>ruleGroups</code> , <code>channels</code> or a combination of them must be specified!

Field	Required	Default	Datatype	Parameter description
channels	no		ENTITY_ITEM	Optional parameter which specifies channels in form of an entity item array. Either <code>rules</code> , <code>ruleGroups</code> , <code>channels</code> or a combination of them must be specified!
reportQuery	yes		ReportQuery	the report query which defines the input data set for the data quality check.
entityIdentifier	yes		String	the entity identifier which describes the data type for which the data quality check will be executed. It must correspond to the entity identifier of the rule configuration group.

Result

A data quality result object which contains all relevant information about the execution for each item.

Properties of the returned object		
Field	Data type	Description
ruleIds	Map	A map of rule names to rule ID's. The ID's are only valid in this result object and are needed to reference the results of each item
numberOfSuccessfulItems	Integer	The number of items which completed all rules successfully
numberOfFailedItems	Integer	The number of items which failed for at least one rule

items		An array of item objects which provide the results for all executed rules per item
<code>entityItem</code>	ENTITY_ITEM	The reference to the entity item for which the rules have been executed
<code>status</code>	String	The overall status for all rules. Might be SUCCESSFUL, FAILED or UNKNOWN. SUCCESSFUL means that all rules have completed successfully FAILED means that at least a single rule failed UNKNOWN means that for at least one rule no results are available. Usually this can only happen in case some unexpected error happens - please check the protocol in this case.
<code>failedRuleIds</code>	Array of Integer	The id's of all rules which have failed for this item (see ruleIds above)
<code>successfulRuleIds</code>	Array of Integer	The id's of all rules which have succeeded for this item (see ruleIds above)
<code>protocol</code>		The protocol (also known as problem log) of the execution.
<code>infoCounter</code>	Integer	number of protocol entries with the INFO severity
<code>warningCounter</code>	Integer	number of protocol entries with the WARNING severity
<code>errorCounter</code>	Integer	number of protocol entries with the ERROR severity
<code>entries</code>	Array of protocol entries	

severity	String	The severity of the protocol entry. Might be INFO, WARNING or ERROR
category	String	The category of the protocol entry
message	String	The message of the protocol entry
logDate	Date	The date when the protocol entry has been created
logTime	Time	The time when the protocol entry has been created

Examples

In the following examples we assume that there are several rule configurations configured in the PIM system.

Java Rest Client Example

Rest Client Java Code

```
EntityItemReference toolsCatalog = EntityItemReferenceFactory.createByIdentifier(
    "TOOLS" );
EntityItemReference webShopChannel = EntityItemReferenceFactory.createByIdentifier(
    "WebShop" );
EntityItemReference erpChannel = EntityItemReferenceFactory.createByIdentifier( "ERP"
);

ReportQuery reportQuery = new ReportQuery( "byCatalog" ); //$NON-NLS-1$
reportQuery.addParameterValue( "catalog", toolsCatalog );

DataQualityProfile profile = new DataQualityProfile();
profile.setReportQuery( reportQuery );
profile.setEntityIdentifier( "Article" );
profile.addRuleConfigurations( "item_description_rule1", "item_description_rule3" );
profile.addRuleConfigurationGroups( "Item Texts", "Item Attributes" );
profile.addChannels( webShopChannel, erpChannel );

DataQualityRequest request = getRestClient().createDataQualityRequest();
```



```
DataQualityResult result = request.execute( profile );
```

JSON Examples

Execute individual rules "item_description_rule1" and "item_description_rule3" on the TOOLS catalog

```
//POST to http://localhost:1501/rest/V1.0/manage/dataquality/executions
{
  "rules":["item_description_rule1","item_description_rule3"],
  "ruleGroups":["Item Texts","Item Attributes"],
  "channels":["WebShop","ERP"],
  "entityIdentifier":"Article",
  "reportQuery":{
    "identifier":"byCatalog",
    "parameterList":[
      {
        "key":"Catalog",
        "value":"'TOOLS'"
      }
    ]
  }
}
```

DataQuality Result object in JSON format

```
{
  "ruleIds": {
    "CheckGtinMED": 12
  },
  "numberOfSuccessfulItems": 5,
  "numberOfFailedItems": 0,
  "items": [
    {
      "entityItem": {
        "id": "15@1"
      },
      "status": "SUCCESSFUL",
      "failedRuleIds": [],
      "successfulRuleIds": [
        12
      ]
    },
    {
      "entityItem": {
        "id": "137@1"
      },
      "status": "SUCCESSFUL",
```

```

    "failedRuleIds": [],
    "successfulRuleIds": [
      12
    ]
  },
  {
    "entityItem": {
      "id": "121@1"
    },
    "status": "SUCCESSFUL",
    "failedRuleIds": [],
    "successfulRuleIds": [
      12
    ]
  },
  {
    "entityItem": {
      "id": "19@1"
    },
    "status": "SUCCESSFUL",
    "failedRuleIds": [],
    "successfulRuleIds": [
      12
    ]
  },
  {
    "entityItem": {
      "id": "149@1"
    },
    "status": "SUCCESSFUL",
    "failedRuleIds": [],
    "successfulRuleIds": [
      12
    ]
  }
],
"protocol": {
  "infoCounter": 3,
  "warningCounter": 0,
  "errorCounter": 0,
  "entries": [
    {
      "severity": "INFO",
      "category": "SUMMARY",
      "message": "1 Regel wird auf 5 Objekte des Typs 'Artikel' angewendet",
      "logDate": "2016-01-04",
      "logTime": "14:52:00"
    },
    {
      "severity": "INFO",
      "category": "SUMMARY",
      "message": "Ausgeführte Regeln: CheckGtinMED",

```

```

        "logDate": "2016-01-04",
        "logTime": "14:52:00"
    },
    {
        "severity": "INFO",
        "category": "SUMMARY",
        "message": "Verarbeitung der Regeln beendet.",
        "logDate": "2016-01-04",
        "logTime": "14:52:00"
    }
]
}
}

```

Schedule Rule Execution

Executes all rules for a given amount of items. The rules to be executed can be defined either by a list of single rule configurations, rule configuration groups, channels or a combination of all options. Please note that this method replaces all other methods which are now deprecated (see below).

URL Pattern	/manage/dataquality/jobs
Method	POST
Content types	application/json, application/xml
Media types	application/json, application/xml
Result	The job object of the scheduled data quality job

Query Parameters

Parameter	Required	Default	Datatype	Parameter description
workflowServiceEndpoint	no		String	Informatica BPM callback parameter. Defines the name of the service endpoint which must be available in an attached Informatica BPM instance.

Parameter	Required	Default	Datatype	Parameter description
workflowCorrelationId	no		String	Informatica BPM callback parameter. An arbitrary id which is used by the Informatica BPM workflow to identify the correct workflow process.
workflowCommunicationMode	no	REST	REST/ QUEUE	Informatica BPM callback parameter. Defines the communication mode which can be using JMS message queue and REST communication.
workflowQueueId	no	First trigger queue id in server.properties	String	Informatica BPM callback parameter. An queue id defined in the server properties in the message queue section which is used as response queue

Content

The content has to be a DataQualityProfile JSON object which is described in the following table

Field	Required	Default	Datatype	Parameter description
rules	no		String Array	Optional parameter which specifies individual rule names in form of a string array. Either rules, ruleGroups, channels or a combination of them must be specified!

Field	Required	Default	Datatype	Parameter description
ruleGroups	no		String Array	Optional parameter which specifies rule group names in form of a string array. Either <code>rules</code> , <code>ruleGroups</code> , <code>channels</code> or a combination of them must be specified!
channels	no		ENTITY_ITEM	Optional parameter which specifies channels in form of an entity item array. Either <code>rules</code> , <code>ruleGroups</code> , <code>channels</code> or a combination of them must be specified!
reportQuery	yes	<input type="checkbox"/>	ReportQuery	the report query which defines the input data set for the data quality check.
entityIdentifier	yes	<input type="checkbox"/>	String	the entity identifier which describes the data type for which the data quality check will be executed. It must correspond to the entity identifier of the rule configuration group.

Result

An object reference to the data quality job.

Properties of the returned object		
Field	Data type	Description
id	Integer	The job ID

Workflow Callback (since 8.0.03)

If a dataquality run is executed with the workflowServiceEndpoint parameter given, the job will create a callback request to the Informatica BPM server with additional information by the time of finish.

Additional information include job information, as well as report information about the filtered entity objects. Based on the report ids it is even possible using the list api to retrieve the items regarding the different status groups.

Workflow Callback (since 10.0)

It is possible to add the query parameters workflowServiceEndpoint and workflowQueueId and workflowCommunicationMode which allows to specify that the response is send via the message queue. The workflowServiceEndpoint is send back as JMS property P360TargetService which allows BPM to call workflow endpoints. The workflowQueueId parameter specifies a queue configured in the server.properties with syntax "queue.[queueId].name".

Workflow Callback Example

In the following example two items have been processed with two rules (GTINRule,ManufacturerRule).

The report results lists reports for following entry keys:

- SUCCESSFUL
- FAILED
- ManufacturerRule_FAILED
- GTINRule_SUCCESSFUL
- GTINRule_FAILED

The first two categories (SUCCESSFUL, FAILED) shows how many items did fail with any rule (FAILED) or were successful for all rules (SUCCESSFUL).

The further categories are rule specific (ManufacturerRule_FAILED,GTINRule_SUCCESSFUL,GTINRule_FAILED). They show e.g. for the GTINRule_FAILED entry, how many items did fail specific for the 'GTINRule' rule.

With the report result id it is possible the retrieve the affected items via the byReport query of the List API.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jobFinished>
  <jobId>24</jobId>
  <stateIdentifier>finished.info</stateIdentifier>
  <stateLabel>Completed</stateLabel>
  <reportResults>
    <entry key="SUCCESSFUL"/>
    <entry key="FAILED">
      <reportResult>
        <id>30</id>
        <dataSource>PCM_MASTER</dataSource>
      </reportResult>
    </entry>
  </reportResults>
</jobFinished>
```

```

        <type>1</type>
        <purpose>1</purpose>
        <resultTableName>ReportStoreTempB7</resultTableName>
        <count>2</count>
        <entityIdentifier>Article</entityIdentifier>
    </reportResult>
</entry>
<entry key="UNKNOWN"/>
<entry key="GTINRule_FAILED">
    <reportResult>
        <id>21</id>
        <dataSource>PCM_MASTER</dataSource>
        <type>1</type>
        <purpose>1</purpose>
        <resultTableName>ReportStoreTempB8</resultTableName>
        <count>1</count>
        <entityIdentifier>Article</entityIdentifier>
    </reportResult>
</entry>
<entry key="ManufacturerRule_FAILED">
    <reportResult>
        <id>30</id>
        <dataSource>PCM_MASTER</dataSource>
        <type>1</type>
        <purpose>1</purpose>
        <resultTableName>ReportStoreTempB7</resultTableName>
        <count>2</count>
        <entityIdentifier>Article</entityIdentifier>
    </reportResult>
</entry>
<entry key="GTINRule_SUCCESSFUL">
    <reportResult>
        <id>32</id>
        <dataSource>PCM_MASTER</dataSource>
        <type>1</type>
        <purpose>1</purpose>
        <resultTableName>ReportStoreTempB9</resultTableName>
        <count>1</count>
        <entityIdentifier>Article</entityIdentifier>
    </reportResult>
</entry>
</reportResults>
</jobFinished>

```

Examples

Executing a DQ checks for the catalog *TOOLS*

In the following examples we assume that there are several rule configurations configured in the PIM system.

Java Rest Client Example

Rest Client Java Code

```
EntityItemReference toolsCatalog = EntityItemReferenceFactory.createByIdentifier(
    "TOOLS" );
EntityItemReference webShopChannel = EntityItemReferenceFactory.createByIdentifier(
    "WebShop" );
EntityItemReference erpChannel = EntityItemReferenceFactory.createByIdentifier( "ERP"
);

ReportQuery reportQuery = new ReportQuery( "byCatalog" ); //$NON-NLS-1$
reportQuery.addParameterValue( "catalog", toolsCatalog );

DataQualityProfile profile = new DataQualityProfile();
profile.setReportQuery( reportQuery );
profile.setEntityIdentifier( "Article" );
profile.addRuleConfigurations( "item_description_rule1", "item_description_rule3" );
profile.addRuleConfigurationGroups( "Item Texts", "Item Attributes" );
profile.addChannels( webShopChannel, erpChannel );

DataQualityRequest request = getRestClient().createDataQualityRequest();
EntityItemReference job = request.scheduleExecution( profile );
```

JSON Examples

Execute individual rules "item_description_rule1" and "item_description_rule3" on the TOOLS catalog

```
//POST to http://localhost:1501/rest/V1.0/manage/dataquality/jobs
{
    "rules":["item_description_rule1","item_description_rule3"],
    "ruleGroups":["Item Texts","Item Attributes"],
    "channels":["'WebShop'", "'ERP'"],
    "entityIdentifier":"Article",
    "reportQuery":{
```



```

    "identifier": "byCatalog",
    "parameterList": [
      {
        "key": "Catalog",
        "value": "'TOOLS'"
      }
    ]
  }
}

```

Execute all rules for the rule groups "Item Texts" and "Item Attributes"

```

//POST to http://localhost:1501/rest/V1.0/manage/dataquality/jobs
{
  "ruleGroups": ["Item Texts", "Item Attributes"],
  "entityIdentifier": "Article",
  "reportQuery": {
    "identifier": "byCatalog",
    "parameterList": [
      {
        "key": "Catalog",
        "value": "'TOOLS'"
      }
    ]
  }
}

```

Execute all rules which are mapped to the "WebShop" and "ERP" channels

```

//POST to http://localhost:1501/rest/V1.0/manage/dataquality/jobs
{
  "channels": ["'WebShop'", "'ERP'"],
  "entityIdentifier": "Article",
  "reportQuery": {
    "identifier": "byCatalog",
    "parameterList": [
      {
        "key": "Catalog",
        "value": "'TOOLS'"
      }
    ]
  }
}

```

Deprecated

Please use `/manage/dataquality/jobs` instead of the following old URL's.

Execute a data quality rule configuration group immediately

General Info

URL Pattern	/manage/dataquality/job/{rule-configuration-group-name}
Method	POST
Parameters	-
Content types	application/json
Media types	application/json
Result	The job object of the scheduled data quality job

Content

The content has to be a JSON object which includes the properties listed below. It's called DataQualityProfile.

DataQualityProfile Properties					
	Field	Required	Default	Datatype	Parameter description
	reportQuery	yes		ReportQuery	the report query which defines the input data set for the data quality check.
	entityId	yes		String	the entity identifier which describes the data type for which the data quality check will be executed. It must correspond to the entity identifier of the rule configuration group.

Result

An object reference to the data quality job.

Properties of the returned object			
	Field	Data type	Description
	id	Integer	The job ID

Execute a set of data quality rule configurations

General Info

URL Pattern	/manage/dataquality/execution
Method	POST
Parameters	rules=<rule conf name1>,<conf name 2>,...
Content types	application/json
Media types	application/json
Result	The job object of the scheduled data quality job

Content and result are the same as for the configuration group.

Execute channel's set of data quality rule configurations

General Info

URL Pattern	/manage/dataquality/execution
Method	POST

Parameters	channel=<channelId>
Content types	application/json
Media types	application/json
Result	The job object of the scheduled data quality job

Content and result are the same as for the configuration group. *ChannelId* is an internal id as it appears in

```
http://localhost:1501/rest/V1.0/list/Channel/bySearch?
query=%20not%20(Channel.Identifier%20is%20empty)&fields=ChannelLang.Name(en)
```

or an external id in single quotes.

Examples

Executing a dq check on the item descriptions of the catalog *TOOLS*

In the following example we assume that there is a rule configuration group which is called "item_descriptions" configured within the PIM system. To run this rule configuration immediately against the catalog "TOOLS" you need to execute the following code snippet.

Rest Client Java Code

```
EntityItemReference catalog = EntityItemReferenceFactory.createByIdentifier( "TOOLS"
);

ReportQuery reportQuery = new ReportQuery( "byCatalog" );
reportQuery.addParameterValue( "catalog", catalog );

DataQualityProfile dataQualityProfile= new DataQualityProfile();
dataQualityProfile.setReportQuery( reportQuery );
dataQualityProfile.setEntityIdentifier( "Article" );

EntityItemReference result =
getRestClient().createDataQualityRequest().scheduleRuleConfigurationGroup( "item_desc
riptions",
dataQualityProfile );
```

To run individual rules item_description_rule1 and item_description_rule3 on the same catalog with JSON

JSON example

```
//POST to http://localhost:1501/rest/V1.0/manage/dataquality/execution?
rules=item_description_rule1,item_description_rule3
{
  "reportQuery":{
    "identifier":"byCatalog",
    "parameterList":[{"key":"Catalog","value":"'TOOLS'"}]
  },
  "entityIdentifier":"Article"
}
```

13.6.7.3 REST Environment API

REST Environment APIs used for Extraction, Download and Integration of Environment transfer

Extraction

URL Pattern	/manage/environment/transfer/extraction
Method	POST
Content types	application/json, application/xml
Parameters	transferHandlers (eg: characteristics,lookups)
Returns	jobId, currentState and progress of extraction job If the job is finished, also the link to download the extraction archive as well as the protocol of the job will be returned

Query Parameters

Parameter	Required	Default	Datatype	Parameter description
transferHandlers	no	all the available handlers	String	A comma separated string of transfer handlers that specify the required transfer parameters for extraction of the environment.

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" "Content-Type: application/json" -X POST http://localhost:1512/rest/V1.0/manage/environment/transfer/extraction?transferHandlers=lookups,characteristics
```

Result

An object including required details as follows:

JSON Result:

```
{
  "job": {
    "id": "615"
  },
  "currentState": "scheduled",
  "progress": 0
}
```

XML Result:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jobProgress>
  <job>
    <id>615</id>
  </job>
  <currentState>scheduled</currentState>
  <progress>0</progress>
</jobProgress>
```

Download

URL Pattern	/manage/environment/transfer/{jobId}/archive
Method	GET
Content types	application/json, application/xml
Parameters	jobId received by triggering the extraction job
Returns	stream of the object to be downloaded.

URL Parameters

Parameter	Required	Default	Datatype	Parameter description
{jobId}	yes		Integer	Unique ID of the job responsible for extraction

Rest Call

```
curl -u rest:heiler -H "Accept: application/octet-stream" "Content-Type: application/json" -X GET http://localhost:1512/rest/V1.0/manage/environment/transfer/615/archive
```

Result

A stream of the object to be downloaded is returned.

Integration

URL Pattern	/manage/environment/transfer/integrate
Method	POST
Content types	multipart/form-data
Parameters	transferHandlers (eg: characteristics,lookups)
Returns	jobId and currentState and progress of integrate job

Query Parameters

Parameter	Required	Default	Datatype	Parameter description
transferHandlers	no	all the available handlers	String	A comma separated string of transfer handlers that specify the required transfer parameters for integration of the environment.

Form Data Parameters

Parameter	Required	Default	Datatype	Parameter description
file	yes		Stream	An handler for the file stream that is required for environment integration

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" "Content-Type: multipart/form-data"
--data-binary "@C:/users/shared/pim_2018-08-22_05_23_24.428.zip" -X POST http://
localhost:1512/rest/V1.0/manage/environment/transfer/integrate?
transferHanlers=lookups,characteristics
```

Result

An object including required details as follows:

JSON Result:

```
{
  "job": {
    "id": "616"
  },
  "currentState": "scheduled",
  "progress": 0
}
```

XML Result:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jobProgress>
  <job>
    <id>616</id>
  </job>
  <currentState>scheduled</currentState>
  <progress>0</progress>
</jobProgress>
```

Checking the Progress of the Extraction/Integration Job

URL Pattern	/manage/environment/transfer/{jobId}
Method	GET
Content types	application/json, application/xml

Parameters	<code>jobId</code> received by triggering the extraction job
Returns	<p><code>jobId</code> and <code>currentState</code> and progress of extraction job</p> <p>If the job is finished, also the link to download the extraction archive as well as the protocol of the job will be returned only for extraction job</p>

URL Parameters

Parameter	Required	Default	Datatype	Parameter description
<code>{jobId}</code>	yes		Integer	Unique ID of the job to check progress

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" "Content-Type: application/json" -X
GET http://localhost:1512/rest/V1.0/manage/environment/transfer/615
```

Result

An object including required details as follows:

JSON Result:

```
{
  "job": {
    "id": "615"
  },
  "currentState": "finished.info",
  "progress": 100,
  "archiveUrl": "http://localhost:1512/rest/V1.0/manage/environment/transfer/615/
archive",
  "protocol": {
    "infoCounter": 0,
    "warningCounter": 0,
    "errorCounter": 0,
    "entries": []
  }
}
```

}

XML Result:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jobProgress>
  <job>
    <id>615</id>
  </job>
  <currentState>finished.info</currentState>
  <progress>100</progress>
  <archiveUrl>http://localhost:1512/rest/V1.0/manage/environment/transfer/615/
archive</archiveUrl>
  <protocol>
    <infoCounter>0</infoCounter>
    <warningCounter>0</warningCounter>
    <errorCounter>0</errorCounter>
    <entries/>
  </protocol>
</jobProgress>
```

Possible Values of "currentState"

State	Value	Discription
SCHEDULED	scheduled	Server Job state constant which indicated that the job is currently scheduled
RUNNING_OK	running.ok	Server Job state constant which indicated that the job is currently running and no protocol info is available, till now
RUNNING_INFO	running.info	Server Job state constant which indicated that the job is currently running and info protocol entries are available
RUNNING_WARNING	running.warning	Server Job state constant which indicated that the job is currently running and warning protocol entries are available
RUNNING_ERROR	running.error	Server Job state constant which indicated that the job is currently running and error protocol entries are available

State	Value	Discription
FINISHED_OK	finished.ok	Server Job state constant which indicated that the job is currently currently finished, having only info protocol entries
FINISHED_INFO	finished.info	Server Job state constant which indicated that the job is currently finished, having at least one warning protocol entry
FINISHED_WARNING	finished.warning	Server Job state constant which indicated that the job is currently finished, having at least one error protocol entry
FINISHED_ERROR	finished.error	Server Job state constant which indicated that the job has been canceled by the user
CANCELED_OK	canceled.ok	Server Job state constant which indicated that the job has been canceled by the user, having only info protocol entries
CANCELED_INFO	canceled.info	Server Job state constant which indicated that the job has been canceled by the user, having at least one warning protocol entry
CANCELED_WARNING	canceled.warnin g	Server Job state constant which indicated that the job has been canceled by the user, having at least one error protocol entry
CANCELED_ERROR	canceled.error	Server Job state constant which indicated that the job has been canceled by a kill of the server (and not a proper shutdown)
CANCELED_SYSTEM	canceled.system	Server Job state constant which indicated that the job has been canceled due to a serious exception
CANCELED_CRITICAL	canceled.critical	Server Job state constant which indicated that the job has been canceled during a proper server shutdown

State	Value	Discription
CANCELED_SHUTDOWN	canceled.shutdown	Server Job state constant which indicated that the job is still running, but a cancel has been requested, either by the user or by a server shutdown
CANCELED_PENDING	canceled.pending	Server Job state constant which indicates that the job is queued and waiting for the next free execution slot
QUEUED_WORKLOAD	queued.workload	All Job States, which indicate, that the job is running.

13.6.7.4 REST Export API



The Rest Export API allows to schedule new export jobs and to cancel running export jobs.

- [Schedule Export Job](#) (see page 729)
 - [Query Parameters](#) (see page 730)
 - [Content](#) (see page 731)
 - [Property variables](#) (available with 8.0.02) (see page 732)
 - [Property dataSources](#) (available with 8.0.02) (see page 732)
 - [Result](#) (see page 734)
 - [Complete example](#) (see page 735)
- [Information about an export profile](#) (see page 735)
 - [Result](#) (see page 736)
 - [Example](#) (see page 736)
- [Cancel Export Job](#) (see page 737)
 - [URL Parameters](#) (see page 737)
 - [Result](#) (see page 737)

Schedule Export Job

Schedules an export.

URL Pattern	<code>/manage/export</code>
Method	POST

Content types	application/json, application/xml
Media types	application/json, application/xml
Result	The job object of the scheduled export job

Query Parameters

Parameter	Required	Default	Datatype	Parameter description
workflowServiceEndpoint	no		String	Informatica BPM callback parameter. Defines the name of the service endpoint which must be available in an attached Informatica BPM instance.
workflowCorrelationId	no		String	Informatica BPM callback parameter. An arbitrary id which is used by the Informatica BPM workflow to identify the correct workflow process.
workflowCommunicationMode	no	REST	REST/ QUEUE	Informatica BPM callback parameter. Defines the communication mode which can be using JMS message queue and REST communication.
workflowQueueId	no	First trigger queue id in server.properties	String	Informatica BPM callback parameter. An queue id defined in the server properties in the message queue section which is used as response queue

Content

The content has to be a JSON object or an exportSchedulingProfile entity which includes the properties listed below.

Properties				
Field	Required	Default	Datatype	Parameter description
profileName	yes		String	Export profile name
comment	no		String	Comment for the process overview
executionDate	no	now	DATETIME	Date and time when the export should be started (Note: Before 8.0.02 the following format has to be used: "yyyy-MM-dd HH:mm:ss", e.g., "2015-11-16 16:06:34")
variables (available with 8.0.02)	no		Map<String, Object>	Sets the variables specified in the export template
dataSources (available with 8.0.02)	no		Map<String, DataSource>	Allows to set a report query for each data source
parameters (deprecated with 8.0.02)	no		Map<String, String>	Map of export parameters (key/value pairs)

Property `variables` (available with 8.0.02)

The property `variables` is a set of key / value pairs, whereby the key is the name of the variable. The value has to be valid according to the underlying datatype and the enumeration of the variable.

The datatype and the enumeration (if available) can be obtained by requesting information about an export profile (`/manage/export/{ExportProfileName}` , see below) . The valid formatting for the data types is described in [REST Datatypes](#) (see page 436).

Example:

```

1  {
2    "profileName": "MyExportProfile",
3    "variables": {
4      "BooleanVar": "true",
5      "CurrencyVar": "USD",
6      "DateAndTimeVar": "2015-11-16T16:06:34",
7      "DateTimeVar": "2015-11-16",
8      "LongVar": "87",
9      "StringVar": "Hello Export"
10   }
11 }
```

Property `dataSources` (available with 8.0.02)

The property `dataSources` is a key value map which allows to set a report query for each data source. The key is the name of the data source, the value is an object containing the report query.

The available data sources can be obtained by requesting information about an export profile (`/manage/export/{ExportProfileName}` , see below).

Internally, the property `assortment` of the data source in the export profile is used for the report query.

This property has to be set to editable in the data source otherwise it is not possible to specify a report query for that data source. Note that the report query allows to specify an assortment, so it is still possible to specify an assortment.

The report query has the following properties:

Properties				
Field	Required	Default	Datatype	Parameter description

Properties				
identifier	yes		String	Identifier of the report
parameters	no		Map<String,String>	Parameters
assortmentFilter	no		ENTITY_ITEM	Reference to an assortment (must be a reference object)
updateAssortment	no	false	Boolean	Specifies if the assortment should be updated before the

Examples:

```

1  {
2    "profileName": "MyExportProfile",
3    "dataSources": {
4      "Item list" : {
5        "reportQuery": {
6          "identifier": "bySearch",
7          "parameters": {
8            "query": "ArticleLang.DescriptionShort(en) startsWith
9          \"Long live the Export\""
10         },
11         "assortmentFilter" : { "id": "4" },
12         "updateAssortment" : true
13       }
14     }
15   }

```

```

1  {"profileName": "MyExportProfile",
2    "dataSources": {
3      "Item list" : {
4        "reportQuery": {
5          "identifier": "byCatalog",
6          "parameters": {

```

```

7         "catalog": "'mySupplierCatalog'"
8     }
9 }
10 }
11 }
12 }

```

parameters (deprecated since 8.0.02)

Variables can be passed as special parameters. The name of the variable has to end with `##variable`, e.g., `myVar##variable`.

```

{
  "profileName": "REST",
  "parameters": {"myVar##variable": "sampleValue"}
}

```

or

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<exportSchedulingProfile>
  <profileName>REST</profileName>
  <parameters>
    <entry>
      <key>myVar##variable</key>
      <value>sampleValue</value>
    </entry>
  </parameters>
</exportSchedulingProfile>

```

Note that `##variable` **has** to be appended to the variable name to create a corresponding parameter.

Result

Field	Data type	Description
id	Integer	The ID of the export job
label	String	The label of the export job

Complete example

A complete example with two variables and two data sources

```

1  {
2    "profileName":"MyExportProfile",
3    "comment":"MyComment",
4    "executionDate":"2015-11-06T12:00:00",
5    "variables":{
6      "FirstVar":"Value for first varibale",
7      "SecondVar":"Value for second varibale",
8    },
9    "dataSources": {
10     "Item list 1" :{
11       "reportQuery": {
12         "identifier": "bySearch",
13         "parameters": {
14           "query": "ArticleLang.DescriptionShort(en) startsWith
15         \"Long live the Export\""
16       },
17       "assortmentFilter" : { "id": "4"},
18       "updateAssortment" : true
19     },
20     "Item list 2" :{
21       "reportQuery": {
22         "identifier": "bySearch",
23         "parameters": {
24           "query": "Article.SupplierAID startsWith \"A1\""
25         }
26       }
27     }
28   }
29 }

```

Information about an export profile

Returns the available variables and data sources for the specified export profile.

URL Pattern	/manage/export/{ExportProfileName}
Method	GET
Media types	application/json, application/xml

Result	Available variables and data sources of the specified export profile.
--------	---

Result

Field	Data type	Description
profileName	String	The name of the export profile
variables	Map<String, VarInfo>	Contains variable names together with information about the data type, the enumeration (if available), if the variable is mandatory and the default value
dataSources	Map<String, DataSourceInfo>	Contains the available data sources and for each data source the entity

Example

```
{
  "profileName": "MyExportProfile",
  "variables": {
    "BooleanVar": {
      "dataType": "BOOLEAN",
      "mandatory": false,
      "defaultValue": "0"
    },
    "CurrencyVar": {
      "dataType": "STRING",
      "enumeration": "Enum.Currency",
      "mandatory": false,
      "defaultValue": "Euro"
    },
  },
  "dataSources": {
    "Item list 1": {
      "entity": "Article"
    },
    "Item list 2": {
      "entity": "Article"
    }
  }
}
```

```
}
}
```

Cancel Export Job

Cancels a scheduled or running exportjob. If the job has been finished already, this request has no effect.

URL Pattern	/manage/export/{ jobId }/cancel
Method	POST
Content types	application/json, application/xml
Media types	application/json, application/xml
Result	The job-id, the old job state and the new job state.

URL Parameters

Parameter	Required	Default	Datatype	Parameter description
{ jobId }	yes		Integer	Unique ID of the job to cancel

Result

Field	Data type	Description
id	Integer	The job ID
oldState	String	The old state of the job.

Field	Data type	Description
<code>newState</code>	<code>String</code>	The new state of the job.

13.6.7.5 REST File API

The Rest File API can be used for uploading and downloading files to the Product Manager Server, e.g. for providing the source data of an import or for downloading the mime values of other api's.

- [Downloading a File](#) (see page 738)
- [Uploading a File](#) (see page 739)
 - [Result](#) (see page 739)
 - [Examples](#) (see page 739)
- [Retention of uploaded files](#) (see page 740)

Downloading a File

Downloads the binary content of the file identified by the file id parameter.

URL Pattern	<code>/manage/file/{fileId}</code>	
Method	<code>GET</code>	
URL Parameter	<code>fileId</code>	<p>the unique id of the file. It's typically in the form of a UUID, e.g.</p> <p>4033ac68-eb3b-4cdb-bfac-953c06a819ea</p> <p>This is the same id which is returned from the upload api.</p>
Accept types	<code>application/octet-stream</code>	
Returns	The binary file stream of the file for the given file id	

Uploading a File

Uploads the binary stream into a new folder on the server which uniquely identifies this upload. The file will be created in this folder with the given original filename. Every upload will create a separate folder as not to interfere with each other.

URL Pattern	<code>/manage/file</code>
Method	<code>POST</code>
Query Parameter	<code>originalFilename</code>
Content-Type	<code>application/octet-stream</code>
Accept	<code>application/json</code> <code>application/xml</code>
Returns	a file object containing the ID of the upload folder and the file name

Result

An object including an ID and a file name is returned, e.g.:

```
{
  "originalFilename": "Data1.xls",
  "id": "5b588807-5ad5-47db-a08b-3b3ca6573b41"
}
```

The file name is the same as provided in the parameter, the id represents the folder name the file has been stored in (under the upload directory of the Product Manager).

Examples

Uploading a file with file name Data1.xls

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -H "Content-Type:application/octet-stream" --data-binary "@Data.xlsx" -X POST http://localhost:1501/rest/V1.0/manage/file?originalFilename=Data.xls
```

An object including an ID and an URL is returned:

```
{
  "id":"fc645958-b169-44eb-91df-6ce30b3c4b11",
  "originalFilename":"Data.xls"
}
```

Rest Client Java Code

```
FileUploadRequest fileUploadRequest = restClient.createFileUploadRequest();

FileReference result = fileUploadRequest.uploadFile( "Data.xlsx", inputStream );
```

Uploading big files

When uploading big files with curl the `-T` command is recommended because the `--data-binary` command loads the complete file into memory and can lead to out of memory issues.

Using `-T` requires to send the header `-H "Transfer-Encoding: chunked"` in addition.

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -H "Content-Type:application/octet-stream" -H "Transfer-Encoding: chunked" -T "BigData.xlsx" -X POST "http://localhost:1501/rest/V1.0/manage/file?originalFilename=BigData.xlsx"
```

Retention of uploaded files

Uploaded files will be deleted either by the process which uses them or by an automated cleanup process.

If you upload a file to use it for an import process, the import will delete the file once it has copied it to his internal structure. If you request a `mimeValueArchive` for download, the file will be deleted latest after 24 hours.

The retention time of files which are automatically deleted can be configured on the server. By default the job executes every hour and will delete all files which are older than 24 hours.

13.6.7.6 REST Import API

The Rest Import API allows to schedule new import jobs and to cancel running import jobs.

- [Schedule Import Job](#) (see page 741)
 - [Query Parameters](#) (see page 741)
 - [Content](#) (see page 742)
 - [Result](#) (see page 745)
- [Cancel Import Job](#) (see page 745)
 - [Result](#) (see page 746)
- [Examples](#) (see page 746)
 - [Scheduling an import for a certain date and time](#) (see page 746)
 - [Cancel a running import job](#) (see page 747)

Schedule Import Job

Schedules an import. The import files have to be uploaded first using the [REST File API](#) (see page 738).

URL Pattern	/manage/import
Method	POST
Content types	application/json
Media types	application/json, application/xml
Result	The job object of the scheduled import job

Query Parameters

Parameter	Required	Default	Datatype	Parameter description
workflowServiceEndpoint	no		String	Informatica BPM callback parameter. Defines the name of the service endpoint which must be available in an attached Informatica BPM instance.

Parameter	Required	Default	Datatype	Parameter description
workflowCorrelationId	no		String	Informatica BPM callback parameter. An arbitrary id which is used by the Informatica BPM workflow to identify the correct workflow process.
workflowCommunicationMode	no	REST	REST/ QUEUE	Informatica BPM callback parameter. Defines the communication mode which can be using JMS message queue and REST communication.
workflowQueueId	no	First trigger queue id in server.properties	String	Informatica BPM callback parameter. An queue id defined in the server properties in the message queue section which is used as response queue

Content

The content has to be a JSON object which includes the properties listed below.

Properties				
Field	Required	Default	Datatype	Parameter description
executionDate	no	now	DATETIME (see page 438)	Date and time when the import should be started

Properties				
Field	Required	Default	Datatype	Parameter description
files	yes		A list of file objects	A list of objects describing an uploaded file. The object has to include the property <code>id</code> . The number and order of the files has to match the files specified in the import mapping.
mapping	yes		ENTITY_ITEM	A reference to the import mapping stored in the database. Import mappings have the entity <i>ImportProfile</i> . The property <code>id</code> has to be specified.
entitySpecificData	no		A list of JSON objects	Each object contains information that is specific to the imported entity, for example, the catalog in case of <i>Articles</i> .
options	no		JSON object	A JSON object containing the import options
Properties of the entitySpecificData element				
Field	Required	Default	Datatype	Parameter description
entityIdentifier	yes		String	The identifier of the entity that should be imported, for example, Article
properties	no		JSON object	Properties for the specified entity

Properties of the `properties` element for entity *Article* (standard without customizing)

Field	Required	Default	Datatype	Parameter description
<code>parent</code>	yes		String	The id of the catalog.
<code>groupAssignmentMode</code>	no	0	Integer	If set to <ul style="list-style-type: none"> 0: References to structure groups are added to existing structure groups 1: References to structure groups replace existing structure groups

Properties of the `options` element

Field	Required	Default	Datatype	Parameter description
<code>createNewObjects</code>	no	true	Boolean	If set to <code>true</code> , new objects are created during the import. If set to <code>false</code> , objects that do not already exist are not imported.
<code>updateExistingObjects</code>	no	true	Boolean	If set to <code>true</code> , objects that exist already are updated. If set to <code>false</code> , objects that do not already exist are not imported.
<code>dryRunBeforeImport</code>	no	false	Boolean	A dry run is executed before the actual import. Only if the dry run does not contain any errors, the actual import is executed.
<code>dryRunOnly</code>	no	false	Boolean	Only a dry run is executed, no objects are imported.

Properties of the options element				
Field	Required	Default	Datatype	Parameter description
errorMode	no	TOLERANT	String	Either TOLERANT (objects containing errors are imported as far as possible) or RESTRICTIVE (objects)
numberOfThreads	no	Depends on server settings	Integer	Number of threads which are used for the import. The number of threads can be reduced to lower the server load. <i>Currently not supported, will be available in HPM 7.0.01.</i>
useMappingDefault	no	false	boolean	If set to true, the import settings of the import mapping will be used, instead of the options. (since version 8.0.6.05)

Result

An object reference to the import job.

Properties of the returned object		
Field	Data type	Description
id	Integer	The job ID
label	String	The label of the job

Cancel Import Job

Cancels a scheduled or running import job. If the job has been finished already, this request has no effect.

URL Pattern	/manage/import/{jobId}/cancel
Method	POST
Content types	application/json
Result	The job-id, the old job state and the new job state.

Result

Properties of the returned object		
Field	Data type	Description
id	Integer	The job ID
oldState	String	The old state of the job.
newState	String	The new state of the job.

Examples

Scheduling an import for a certain date and time

1	<pre>curl -u rest:heiler -H "Accept: application/json" -H "Content-Type:application/octet-stream" --data-binary "@Data.xlsx" -X POST http://localhost:1501/rest/V1.0/manage/file?originalFilename=Data.xlsx</pre>
---	---

1	{	"executionDate": "2013-04-11T11:00:00",
2	"files": [
3	{ "id": "fc645958-b169-44eb-91df-6ce30b3c4b11" }	
4],	
5	"mapping": { "id": "73" },	
6	"entitySpecificData": [{	
7	"entityIdentifier" : "Article",	

```

8      "properties" : {
9          "parent": "1",
10         "groupAssignmentMode": "1"
11     }
12 } ],
13 "options": {
14     "createNewObjects": true,
15     "updateExistingObjects": true,
16     "dryRunBeforeImport": false,
17     "dryRunOnly": false,
18     "errorMode": "TOLERANT",
19     "numberOfThreads": 10
20 }
21 }
```

A reference to the import job is returned:

```
{
  "id": "291",
  "label": "Import_291"
}
```

Cancel a running import job

Request to cancel the job with the id 291:

```
1  curl -u rest:heiler -H "Accept: application/json" -X POST http://
    localhost:1501/rest/V1.0/manage/import/291/cancel
```

Response containing the old and the new state of the job.

```
{
  "oldState": "scheduled",
  "newState": "canceled.ok",
  "id": "291"
}
```

13.6.7.7 REST KPI API



The Rest KPI API provides some management aspect features, e.g. to schedule jobs to calculate and persist key performance indicator values or delete certain persisted KPI results.

Schedule calculation of key performance indicator (KPI) values

Schedules a KPI calculation job that calculates and persists KPI values for a list of specified KPIs and a time interval.

Returns the scheduled job as EntityItemReference.

The user executing this request must have the permission 'Manage KPI values'. Otherwise, no calculation job will be scheduled and a response with HTTP status forbidden (403) will be returned.

URL Pattern	/manage/kpi/calculationJob
Method	POST
Content types	application/json, application/xml
Media types	application/json, application/xml
Result	The job object of the scheduled KPI calculation job

Content

The content has to be a KPICalculationRequestParameter JSON object which is described in the following table.

Properties of the POST request				
Field	Required	Default	Datatype	Parameter description
kpiToCalculate	no	-	String Array	<p>Specifies a list of KPIs for which the calculation of values should be done. If a KPI with a specific identifier is not registered, then there will be no calculation for it (will be logged as warning), but the remaining KPIs in the list are still considered.</p> <p>In case this parameter is not given, then all KPIs will be used that are contributed as KPI extension point.</p>
referenceTime	no	Defaults to the current date when the job is scheduled.	DATETIME	Reference time that specifies the end time of all day intervals that will be considered.
daysToBeBacktracked	no	<p>Defaults individually for each KPI to the daysToBeBacktracked from the server preferences.</p> <p>If then is no value provided for a specific KPI, defaults individually for each KPI to the daysToBeBacktracked from the KPI extension point.</p>	Integer	<p>Specifies a global value for the days to be backtracked ending at the date from the referenceTime parameter. The global value is used for all KPIs in the kpiToCalculate list.</p> <p>Must be convertible to an integer that is ≥ 0.</p>

Result

An object reference to the KPI calculation job.

Properties of the returned object in the POST response

Field	Data type	Description
id	Integer	The job ID
label	String	The label of the job

Examples

In the examples it is assumed that there are KPI extension points contributed that have "AverageTimeSpentInWorkflow" and "ProcessTimes" as KPI identifier.

Schedule KPI values calculation job for KPIs "AverageTimeSpentInWorkflow" and "ProcessTimes", a reference time of "2013-04-11T11:00:00" and 2 days to be backtracked

With a referenceTime set to "2013-04-11T11:00:00" and "daysToBeBacktracked" set to "2" the following day intervals will be passed to the calculators of each KPI:

```
[2013-04-09T00:00:00 - 2013-04-10T00:00:00), [2013-04-10T00:00:00 -
2013-04-11T00:00:00), [2013-04-11T00:00:00 - 2013-04-11T11:00:00)
```

Java Rest Client**Rest Client Java Code**

```
KPICalculationRequestParameter requestParameter = new
    KPICalculationRequestParameter();
requestParameter.setKpisToCalculate( Arrays.asList( "AverageTimeSpentInWorkflow",
    "ProcessTimes" ) );
requestParameter.setReferenceTime( "2013-04-11T11:00:00" );
requestParameter.setDaysToBeBacktracked( "2" );

KPICalculationRequest calculationRequest =
    getRestClient().createKPICalculationRequest();

calculationRequest.scheduleCalculateKPIValuesJob( requestParameter );
```

JSON

```
//POST to http://localhost:1501/rest/V1.0/manage/kpi/calculationJob as body
{
  "kpisToCalculate": ["AverageTimeSpentInWorkflow", "ProcessTimes"],
  "referenceTime": "2013-04-11T11:00:00",
  "daysToBeBacktracked": "2"
}
```

The response contains a reference to the scheduled calculation job.

Returned response for scheduling a KPI values calculation

```
//Returned response contains job reference
{
  "id": "291",
  "label": "CalculateKPIValues_SingleJob_291"
}
```

Remove persisted results of key performance indicator (KPI) calculations

Removes all persisted values and execution data for certain KPIs and time interval. Thus, by doing this, it is possible to force a recalculation of KPI of the desired time interval which are also persisted again.

Reasons to use this call may be that

- the calculation logic of the contributed KPI calculator was wrong, so the persisted KPI values are wrong as well.
- certain data for a specific time period was not available at the time the calculation execution was done. By removing the execution results it is then possible to have the additional data added as well.

Returns Deletion Result objects containing a Webservice Protocol. This contains log informations (information, warning, error entries) that have been written to the (transient) problem log during the deletion operation.

The user executing this request must have the permission 'Manage KPI values'. Otherwise, no calculation job will be scheduled and a response with HTTP status forbidden (403) will be returned.

URL Pattern	/manage/kpi/removeCalculationResults
Method	POST
Content types	application/json, application/xml

Media types	<code>application/json</code> , <code>application/xml</code>
Result	A Deletion Result object containing a Webservice Protocol of the execution.

Content

The content has to be a KPICalculationRequestParameter JSON object which is described in the following table

Properties of the POST request				
Field	Required	Default	Datatype	Parameter description
kpiToDelete	yes	-	String Array	Specifies a list of KPIs for which persisted results (values and execution information) are to be deleted. If a KPI with a specific identifier is not registered, then there will be no deletion for it (will be logged as warning), but the remaining KPIs in the list are still considered. In case this parameter is not given, then all KPIs will be used that are contributed as KPI extension point.
referenceTime	yes	-	DATETIME	Reference time that specifies the end time of the time interval that will be considered.
daysToBeBacktracked	yes	-	Integer	Specifies the days to be backtracked ending at the Date from the <code>referenceTime</code> parameter. Must be convertible to an integer that is ≥ 0 .

Result

A Deletion result object containing the protocol of the deletion execution.

Properties of the returned object in the POST response		
Field	Data type	Description
protocol		The protocol (also known as problem log) of the execution.
infoCounter	Integer	number of protocol entries with the INFO severity
warningCounter	Integer	number of protocol entries with the WARNING severity
errorCounter	Integer	number of protocol entries with the ERROR severity
entries	Array of protocol entries	
severity	String	The severity of the protocol entry. Might be INFO, WARNING or ERROR
category	String	The category of the protocol entry
message	String	The message of the protocol entry
logDate	Date	The date when the protocol entry has been created
logTime	Time	The time when the protocol entry has been created

Examples

In the examples it is assumed that there is a KPI extension point contributed that has "AverageTimeSpentInWorkflow" as KPI identifier.

However there is no KPI extension contributed with the identifier "ProcessAllTimes".

Remove KPI calculation results for contributed KPI "AverageTimeSpentInWorkflow" and not contributed "ProcessAllTimes", a reference time of "2013-04-11T11:00:00" and 2 days to be backtracked

With a referenceTime set to "2013-04-11T11:00:00" and "daysToBeBacktracked" set to "2" the following time interval will be passed to the deletion execution for each KPI:

```
[2013-04-09T00:00:00 - 2013-04-11T11:00:00)
```

and there will be a warning entry in the protocol object of the result that there is no KPI registered with the identifier "ProcessAllTimes":

```
The KPI with identifier 'ProcessAllTimes' (which is defined in the rest parameters)
is not registered.
```

Java Rest Client

Rest Client Java Code

```
KPIResultsDeletionRequestParameter requestParameter = new
    KPIResultsDeletionRequestParameter();
requestParameter.setKpisToDelete( Arrays.asList( "AverageTimeSpentInWorkflow",
    "ProcessAllTimes" ) );
requestParameter.setReferenceTime( "2013-04-11T11:00:00" );
requestParameter.setDaysToBeBacktracked( "2" );

KPIResultsDeletionRequest deletionRequest =
    getRestClient().createKPIResultsDeletionRequest();

deletionRequest.deleteKPIExecutions( requestParameter );
```

JSON

```
//POST to http://localhost:1501/rest/V1.0/manage/kpi/removeCalculationResults as body
{
    "kpisToDelete": ["AverageTimeSpentInWorkflow", "ProcessAllTimes"],
    "referenceTime": "2013-04-11T11:00:00",
    "daysToBeBacktracked": "2"
}
```

The response contains a KPIDeletionResult object.

KPIDeletionResult object in JSON format

```
{
  "protocol": {
    "infoCounter": 1,
    "warningCounter": 1,
    "errorCounter": 0,
    "entries": [
      {
        "severity": "WARNING",
        "category": "CONSISTENCY",
        "message": "The KPI with identifier 'ProcessAllTimes' (which is defined
in the rest parameters) is not registered.",
        "logDate": "2018-01-04",
        "logTime": "14:52:00"
      },
      {
        "severity": "INFO",
        "category": "SUMMARY",
        "message": "Removal of KPI values finished successfully.",
        "logDate": "2017-09-20",
        "logTime": "08:21:22"
      }
    ]
  }
}
```

13.6.7.8 REST Merge API

The Rest Merge API allows to schedule merge jobs.

- [Schedule Merge Job](#) (see page 755)
 - [Query Parameters](#) (see page 756)
 - [Content](#) (see page 757)
 - [Result](#) (see page 766)
- [Workflow Callback](#) (see page 766)
 - [Workflow Callback \(since 10.0\)](#) (see page 766)
 - [Workflow Callback Example](#) (see page 767)
- [Examples](#) (see page 768)
 - [Schedule merge of the full "Apparel" catalog](#) (see page 768)

Schedule Merge Job

Schedules a merge.

URL Pattern

/manage/merge

Method	POST
Content types	application/json
Media types	application/json
Result	The job object of the scheduled merge job

Query Parameters

Parameter	Required	Default	Datatype	Parameter description
updateAssortment	false	false	Boolean	If <code>updateAssortment</code> is set, corresponding assortment will be updated (which can be time consuming) before performing merge.
workflowServiceEndpoint	no		String	Informatica BPM callback parameter. Defines the name of the service endpoint which must be available in an attached Informatica BPM instance.
workflowCorrelationId	no		String	Informatica BPM callback parameter. An arbitrary id which is used by the Informatica BPM workflow to identify the correct workflow process.

Parameter	Required	Default	Datatype	Parameter description
workflowCommunicationMode	no	REST	REST/ QUEUE	Informatica BPM callback parameter. Defines the communication mode which can be using JMS message queue and REST communication.
workflowQueueId	no	First trigger queue id in server.properties	String	Informatica BPM callback parameter. An queue id defined in the server properties in the message queue section which is used as response queue


Content

The content has to be a `MergeProfile` JSON object which includes the properties listed below.

Field	Required	Default	Datatype	Parameter description
rootEntity	no	"Article"	String	The identifier of the root entity this merge profile refers to. At the moment only the "Article" entity is supported - so this parameter is optional.
catalog	either catalog or entityReportQuery is mandatory!		ENTITY_ITEM	The supplier catalog to be merged from. If no assortment is given, all objects of the specified rootEntity will be merged from this catalog
assortment	no		ENTITY_ITEM	Optional assortment to be merged

Field	Required	Default	Datatype	Parameter description
entityReportQuery	no		EntityReportQuery	An entity report query which allows to use the available reports like 'byCatalog', 'byItems'... (Since 8.0.03)
entityReportQueryExecution	no	DEFERRED	IMMEDIATE or DEFERRED	Defines when the content of the entity report query is resolved: at the moment of calling the rest method or when the merge job runs. If set to DEFERRED, the report is executed when the merge job begins to run. (Since 8.0.03)
executionDate	no	now	DATETIME (see page 438)	Date and time when the merge should be started.

Field	Required	Default	Datatype	Parameter description
defaultAction	no	"SUPPLEMENT"	String	<p>Default merge mode for all fields in case the corresponding supplier catalog has no persisted merge profile.</p> <p>"SUPPLEMENT", "OVERWRITE", "CLEAN_COPY", "NO_MERGE"</p> <p>This parameter doubles as the default action for the entityQualification parameter object.</p>

Field	Required	Default	Datatype	Parameter description
				<p> There is a priority order for the merge actions! Highest priority has the <code>entityQualification</code> parameter. In case this one is given, then this is how the merge is going to be performed. If this parameter is not given, the merge will be done as defined in the <u>merge settings of the supplier catalog</u>! The merge settings of the supplier catalog can be adjusted in the desktop client only. Last priority has the <code>defaultAction</code> parameter. In case the <code>entityQualification</code> is omitted and there are no supplier catalog merge settings, then all fields and entities will be merged according to this <code>defaultAction</code> parameter. Finally, in case this one is also not there, a SUPPLEMENT merge is performed.</p>

Field	Required	Default	Datatype	Parameter description
entityQualification	no		EntityQualification	Optional parameter to fine grain the merge actions down to the field level. Provides also the ability to filter based on qualifications (like: merge only the short description in English). See also special chapter below.

Entity Report Query

The items to be merged can be defined in two different ways. You can either specify the supplier catalog and optionally an assortment, or a more sophisticated entityReportQuery. The entity report query option provides a fine grained way to define even single items. For this you can use all available entity reports of the system. What reports are available can differ from Product 360 to Product 360 installation and therefore are documented by the Service API itself. The documentation can be obtained using the List Info API.

Using byCatalog report query

POST /rest/V1.0/manage/merge

```
{
  "entityReportQuery": {
    "identifier": "byCatalog",
    "entityIdentifier": "Article",
    "parameterList": [{
      "key": "catalog",
      "value": "'Apparel'"
    }]
  }
}
```

Using byItems report query

Note: the items must have the same container, it is not possible to specify items from multiple catalogs

POST /rest/V1.0/manage/merge

```
{
```

```

"entityReportQuery": {
  "identifier": "byItems",
  "entityIdentifier": "Article",
  "parameterList": [{
    "key": "items",
    "value": "'Article1'@'CatalogIdentifier',123@1000"
  }]
}
}

```

Using bySearch report query

Note: the items must have the same container, it is not possible to specify items from multiple catalogs

POST /rest/V1.0/manage/merge

```

{
  "entityReportQuery":{
    "identifier":"bySearch",
    "entityIdentifier":"Article",
    "parameterList":[
      {
        "key":"query",
        "value":"Article.EAN in ( \"1234\" , \"1235\" )"
      } ,
      {
        "key":"catalog",
        "value":"'TestCatalog'"
      }
    ]
  }
}

```

Merge Actions

Mode	Description	Field	Supplier Item	Master Item (Before)	Master Item (After)
NO_MERGE	This is probably the easiest one. If this action is defined, no merge will be performed for the entity or field	Description (en)	"My Changed English Value"	"My English Value"	"My English Value"

Mode	Description	Field	Supplier Item	Master Item (Before)	Master Item (After)
SUPPLEMENT	A field or entity is only merged in case the master item has not already an object/value for this entity or field	Description (en)	"My Changed English Value"	"My English Value"	"My English Value"
		Description (en)	"My Changed English Value"		"My Changed English Value"
OVERWRITE	A field or entity is always merged, in case the master catalog already has a value for this field, the value is overwritten	Description (en)	"My Changed English Value"	"My English Value"	"My Changed English Value"
CLEAN_COPY	Similar to OVERWRITE for fields, but in case of entities is also deletes sub-entities which do not exist in the supplier item The example shows that the master item has a value for German and English, but after the clean copy merge, only the value from the supplier item survives.	Description (en)	"My Changed English Value"	"My English Value"	"My Changed English Value"
		Description (de)		"Mein Deutscher Wert"	

EntityQualification

The `EntityQualification` parameter is supported with version 8.0.6.01 and following only.

The `EntityQualification` is a complex data structure which defines how exactly each field of each sub-entity should be merged. The `EntityQualification` structure is **recursive**, thus, it has children which have the same structure.

❗ It is required to qualify the whole path to the desired subentity. The qualification has to include the root entity.

On each level you can define a `defaultAction` which sets the action for this entity and its fields and all sub-entities which are not explicitly part of the `EntityQualification`. So, you only need to specify `EntityQualification` objects and their `fieldActions` in case they differ from the corresponding `defaultAction`.

Each `EntityQualification` defines the identifier of the `entity`, the `defaultAction`, optional `qualificationFilters` (like, only for a certain language), the `fieldActions` (in case they are different than the `defaultAction`) and the `children` (in case they have a different actions as the `defaultAction`).

Attribute	Datatype	Description
<code>entity</code>	<code>String</code>	the unique identifier of the entity / sub-entity. See the Meta API (see page 870) for details on the possible entities.
<code>defaultAction</code>	<code>String</code>	one of the Merge Actions (<code>NO_MERGE</code> , <code>SUPPLEMENT</code> , <code>OVERWRITE</code> , <code>CLEAN_COPY</code>)
<code>qualificationFilters</code>	<code>String[]</code>	<p>With the <code>qualificationFilters</code> array you can limit this <code>EntityQualification</code> so it only applies for a subset (just a specific language for example)</p> <p>The <code>qualificationFilters</code> is an array of qualification settings. A qualification setting is a <i>qualification name</i> followed by a comma-separated list of values which is put into parentheses .</p> <p>The qualification name is defined in the repository and is included in the Meta-API (see page 870).</p> <p>Example for German language: <code>"qualificationFilters" : ["language(DE)"]</code> The syntax is quite similar to the List API (see page 490) for sub-entities.</p>

Attribute	Datatype	Description
fieldActions	FieldAction[]	Array of FieldAction elements. Each FieldAction holds the fieldIdentifier of a field and an fieldAction . In case a field is not part of this array, it's treated like defined in the defaultAction .
children	EntityQualification[]	optional array of EntityQualification elements to define the sub-entities. If omitted, the sub-entities will be merged like this entity

Example

The following example defines as default " NO_MERGE " which means, that by default, nothing should be merged. It really depends on the use case what to pick as default. Usually the default should be whatever the most fields/children have, so you don't need to specify too much.

So in this example we only want to merge the German (= language(DE)) Name (= ArticleLang.Name) of an item and the English (= language(EN)) Description (= ArticleLang.Description). In order to be able to have different settings based on the qualification or the entity, you can specify an EntityQualification multiple times. However, you must make sure that the qualification values do not overlap (otherwise the result of the merge for this will not be predictable).

All Fields of the ArticleLang entity in SUPPLEMENT mode, but the Name in OVERWRITE

```
"entityQualification": {
  "entity": "Article",
  "defaultAction": "NO_MERGE",
  "children": [
    {
      "entity": "ArticleLang",
      "defaultAction": "NO_MERGE",
      "qualificationFilters": [
        "language(DE)"
      ],
      "fieldActions": [
        {
          "fieldIdentifier": "ArticleLang.DescriptionShort",
```

```

        "fieldAction": "OVERWRITE"
      }
    ]
  },
  {
    "entity": "ArticleLang",
    "defaultAction": "NO_MERGE",
    "qualificationFilters": [
      "language(EN)"
    ],
    "fieldActions": [
      {
        "fieldIdentifier": "ArticleLang.DescriptionLong",
        "fieldAction": "OVERWRITE"
      }
    ]
  }
]
}

```

Result

An object reference to the merge job.

Field	Data type	Description
id	Integer	The job ID
label	String	The label of the job

Workflow Callback

If a merge job is scheduled with a `workflowServiceEndpoint` parameter, a callback is given to the Informatica BPM server.

This call will include data about the job itself as well as reports about the merge result. With help of the id of the report information it is even possible to retrieve the merged items (using the [List API \(see page 470\)](#)).

Workflow Callback (since 10.0)

It is possible to add the query parameters `workflowQueueServiceEndpoint` and `workflowQueueId` and `workflowCommunicationMode` which allows to specify that the response is send via the message queue. The `workflowServiceEndpoint` is send back as JMS property `P360TargetService` which allows

BPM to call workflow endpoints. The `workflowQueueId` parameter specifies a queue configured in the server.properties with syntax "queue.[queueId].name".

Workflow Callback Example

An example callback body to the service endpoint of Informatica BPM if a workflow service endpoint parameter has been specified.

Workflow Callback Example XML

```
<jobFinished>
  <jobId>35</jobId>
  <stateIdentifier>finished.info</stateIdentifier>
  <stateLabel>Beendet</stateLabel>
  <reportResults>
    <entry key="TOTAL">
      <reportResult>
        <id>2557</id><!-- With help of this reportid and the byReportId report of the
list API you can retrieve all items related to this report result -->
        <dataSource>PCM_MASTER</dataSource>
        <type>1</type>
        <purpose>3</purpose>
        <resultTableName>ReportStore0</resultTableName>
        <count>2</count>
        <entityIdentifier>Article</entityIdentifier>
      </reportResult>
    </entry>
    <entry key="UNMODIFIED"/>
    <entry key="NEW">
      <reportResult>
        <id>2558</id>
        <dataSource>PCM_MASTER</dataSource>
        <type>1</type>
        <purpose>3</purpose>
        <resultTableName>ReportStore1</resultTableName>
        <count>2</count>
        <entityIdentifier>Article</entityIdentifier>
      </reportResult>
    </entry>
    <entry key="UPDATED"/>
    <entry key="NEW_AND_UPDATED">
      <reportResult>
        <id>2559</id>
        <dataSource>PCM_MASTER</dataSource>
        <type>1</type>
        <purpose>3</purpose>
        <resultTableName>ReportStore2</resultTableName>
        <count>2</count>
        <entityIdentifier>Article</entityIdentifier>
      </reportResult>
    </entry>
  </reportResults>
</jobFinished>
```

```

    </reportResult>
  </entry>
<entry key="FILTERED_BY_DQ"/>
<entry key="FILTERED_BY_PRODUCT_FINDER"/>
</reportResults>
</jobFinished>

```

Examples

Schedule merge of the full "Apparel" catalog

Java Client API: Merge full "Apparel" catalog with OVERWRITE

```

MergeProfile mergeProfile = new MergeProfile();
mergeProfile.setCatalog( EntityItemReferenceFactory.createByIdentifier( "Apparel"
) );
mergeProfile.setDefaultAction( "OVERWRITE" );

MergeRequest mergeRequest = getRestClient().createMergeRequest();
EntityItemReference mergeJobReference =
mergeRequest.scheduleMergeJob( mergeProfile );

```

Merge Profile: Full Apparel catalog with OVERWRITE action for everything

```

"catalog" : {
  "id" : "Apparel"
},
"rootEntity" : "Article",
"defaultAction" : "OVERWRITE"

```

Merge Profile: Full Apparel Catalog, but only the Long Description in English

```

{
  "rootEntity" : "Article",
  "catalog" : {
    "id" : "Apparel"
  },
  "defaultAction" : "NO_MERGE",
  "entityQualification" : {
    "entity" : "Article",
    "defaultAction" : "NO_MERGE",
    "children" : [ {
      "entity" : "ArticleLang",
      "defaultAction" : "NO_MERGE",

```

```

    "qualificationFilters" : [ "language(EN)" ],
    "fieldActions" : [ {
      "fieldIdentifier" : "ArticleLang.DescriptionLong",
      "fieldAction" : "OVERWRITE"
    } ]
  }
}
}

```

13.6.7.9 REST Revision API

The Rest Revision API allows to create revisions and add entity items to them (this process is called "release to a revision" or a revision release job. The API is available with Version 8.1.0.01 of the product

Adding objects to a revision is a performance and database storage intensive process. Essentially the object will be copied including all the objects it references. Clients and Projects must always consider if the use case really needs the revision logic. Most times the audit trail feature is sufficient!

- [Schedule Revision Release Job](#) (see page 769)
 - [Query Parameters](#) (see page 770)
 - [Content](#) (see page 771)
 - [Result](#) (see page 775)
- [Get Revision Job Information](#) (see page 775)
 - [Result](#) (see page 776)

Schedule Revision Release Job

Schedules a revision release job to be executed as soon as possible and returns the job id for the scheduled job

URL Pattern	/manage/revision
Example	http://localhost:1512/rest/V2.0/manage/revision
Method	POST
Content types	application/json, application/xml
Media types	application/json, application/xml

Result

HTTP-202 (accepted) and the job object of the scheduled revision release job



Please note that the Revision Process will also add all entity items which are referenced from the entity items specified by the report! For example: If you add a single Item to a revision, also its variant will be added and also the structure group the item belongs to. So, the full dependency tree will be added to the revision. It can be compared with a snapshot. In case an object has already been added to the revision and its content didn't change meanwhile, it will not be added a second time. In case the content did change, it will be overwritten in that revision. So there is always only one copy of any object within a revision.

Query Parameters

Parameter	Required	Default	Datatype	Parameter description
workflowServiceEndpoint	no		String	Informatica BPM callback parameter. Defines the name of the service endpoint which must be available in an attached Informatica BPM instance.
workflowCorrelationId	no		String	Informatica BPM callback parameter. An arbitrary id which is used by the Informatica BPM workflow to identify the correct workflow process.
workflowCommunicationMode	no	REST	REST/ QUEUE	Informatica BPM callback parameter. Defines the communication mode which can be using JMS message queue and REST communication.

Parameter	Required	Default	Datatype	Parameter description
workflowQueueId	no	First trigger queue id in server.properties	String	Informatica BPM callback parameter. An queue id defined in the server properties in the message queue section which is used as response queue

Workflow Callback

If a revision job is scheduled with a workflowServiceEndpoint parameter, a callback is given to the Informatica BPM server.

This call will include data about the job itself

An example callback body to the service endpoint of Informatica BPM if a workflow service endpoint parameter has been specified.

Workflow Callback Example XML

```
<jobFinished>
  <jobId>35</jobId>
  <stateIdentifier>finished.info</stateIdentifier>
  <stateLabel>Finished</stateLabel>
</jobFinished>
```

Workflow Callback (since 10.0)

It is possible to add the query parameters workflowQueueServiceEndpoint and workflowQueueId and workflowCommunicationMode which allows to specify that the response is send via the message queue. The workflowServiceEndpoint is send back as JMS property P360TargetService which allows BPM to call workflow endpoints. The workflowQueueId parameter specifies a queue configured in the server.properties with syntax "queue.[queueId].name".

Content

The content has to be a JSON/XML object which includes the properties listed below.

Field	Required	Default	Datatype	Parameter description
revision	yes		EntityItem	The revision to add objects to. The revision will be created in case it does not already exist. Otherwise it will be used and the objects will be just added to it
entityReportQuery	yes		EntityReportQuery	An entity report query which allows to use the available reports like 'byCatalog', 'byItems'... (Since 8.0.03)

Entity Report Query

The items to be released into the revision must be defined by an entity report query. For this you can use all available entity reports of the system. What reports are available can differ from Product 360 to Product 360 installation and therefore are documented by the Service API itself. The documentation can be obtained using the [List Info API \(see page 465\)](#).

The EntityReportQuery also allows to define an optional assortment.

Parameter	Required	Default	Datatype	Parameter description
identifier	yes		String	Identifier of the entity report
entityIdentifier	yes		String	The unique identifier of the root entity
parameterList	no		Parameter	A list of parameters, each having a key and the value. Which parameters are supported or required for which entity report can be obtained with the List Info API (see page 465) .

Parameter	Required	Default	Datatype	Parameter description
assortmentFilter	no		EntityItem	An entity item representation of an assortment. The assortment's item entity must match with the entity of the items the report returns. Additionally to that, also the item parent must match. For example, it's not allowed to query all items of a Catalog "Henri" and specifying an assortment which is bound to the master catalog. You will receive an appropriate error message in case the assortment does not fit.
updateAssortment	no	false	Boolean	This parameter works only in combination with the <code>assortment</code> parameter. If set to <code>true</code> the given assortment will be updated before it is used. Please note that updating an assortment might be a performance intensive task, so take care when using this parameter and think about it if you really need to have the assortment evaluated every time.

JSON Examples

Using byCatalog report query with an assortment

POST /rest/V2.0/manage/revision

```
{
  "revision": "'MyRevisionIdentifier'",
  "entityReportQuery": {
    "identifier": "byCatalog",
    "entityIdentifier": "Article",
    "parameterList": [{
      "key": "catalog",
      "value": "'Apparel'"
    }],
    "assortmentFilter": "4711",
  }
}
```

```

    "updateAssortment": false
  }
}

```

Using byItems report query

Note: the items must have the same container, it is not possible to specify items from multiple catalogs

POST /rest/V2.0/manage/revision

```

{
  "revision": "'MyRevisionIdentifier'",
  "entityReportQuery": {
    "identifier": "byItems",
    "entityIdentifier": "Article",
    "parameterList": [{
      "key": "items",
      "value": "'Article1'@'CatalogIdentifier',123@1000"
    }]
  }
}

```

Using bySearch report query

Note: the items must have the same container, it is not possible to specify items from multiple catalogs

POST /rest/V2.0/manage/revision

```

{
  "revision": "'MyRevisionIdentifier'",
  "entityReportQuery": {
    "identifier": "bySearch",
    "entityIdentifier": "Article",
    "parameterList": [
      {
        "key": "query",
        "value": "Article.EAN in ( \"1234\" , \"1235\" )"
      }, {
        "key": "catalog",
        "value": "'TestCatalog'"
      }
    ]
  }
}

```

Example with the Rest Client API

Schedule revision of the full "TOOLS" catalog**Java Client API: Merge full "Apparel" catalog with OVERWRITE**

```
EntityItemReference catalog = EntityItemReferenceFactory.createByIdentifier( "TOOLS"
);
ReportQuery reportQuery = new ReportQuery( "byCatalog" );
reportQuery.addParameterValue( "catalog", catalog );

RevisionRequest request = getRestClient().createRevisionRequest();
RevisionResult result = request.scheduleRevisionJob( "Article", reportQuery );
```

Result

A result object containing the entity item of the job

Field	Data type	Description
job	EntityItem	The job entity item

Example of the result in JSON

```
{
  "id": "20045",
  "label": "com.heiler.ppm.revision.jobType_20045"
}
```

Get Revision Job Information

Returns current job information about a revision job including the protocol of the job

URL Pattern	/manage/revision/{jobId}
Example	http://localhost:1512/rest/V2.0/manage/revision/20045
Method	GET

Content types	application/json, application/xml
Media types	application/json, application/xml
Result	HTTP-200 (ok) and the job information object about the current state of the release job
	HTTP-404 (not found) in case the job id does not exist or is not a revision release job

Result

Example of the result in JSON

```
{
  "id": 20045,
  "jobType": "com.heiler.ppm.revision.jobType",
  "jobGroup": "DataMaintenanceGroup",
  "user": "rest",
  "creationTime": "2018-01-05T15:07:35:920+0100",
  "modificationTime": "2018-01-05T15:07:36:470+0100",
  "scheduledAt": "2018-01-05T15:07:35:920+0100",
  "currentState": "finished.info",
  "progress": 100,
  "serverIdentifier": "pim-server1",
  "problemLogIdentifier": "com.heiler.ppm.revision.jobType_20046_20046",
  "protocol": {
    "infoCounter": 1,
    "warningCounter": 0,
    "errorCounter": 0,
    "entries": [
      {
        "objectType": "Item",
        "severity": "INFO",
        "category": "SUMMARY",
        "message": "1 objects released in 200 ms",
        "logDate": "2018-01-05",
        "logTime": "15:07:36"
      }
    ]
  }
}
```

Example of the result in XML

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<revisionJobInfo>
  <id>20045</id>
  <jobType>com.heiler.ppm.revision.jobType</jobType>
  <jobGroup>DataMaintenanceGroup</jobGroup>
  <user>rest</user>
  <creationTime>2018-01-05T15:07:35:920+0100</creationTime>
  <modificationTime>2018-01-05T15:07:36:470+0100</modificationTime>
  <scheduledAt>2018-01-05T15:07:35:920+0100</scheduledAt>
  <currentState>finished.info</currentState>
  <progress>100</progress>
  <serverIdentifier>pim-server1 (DEW1PC05KDRB IP: 10.43.104.101)</serverIdentifier>
  <problemLogIdentifier>com.heiler.ppm.revision.jobType_20046_20046</
problemLogIdentifier>
  <protocol>
    <infoCounter>1</infoCounter>
    <warningCounter>0</warningCounter>
    <errorCounter>0</errorCounter>
    <entries>
      <entry>
        <objectType>Item</objectType>
        <severity>INFO</severity>
        <category>SUMMARY</category>
        <message>1 objects released in 200 ms</message>
        <logDate>2018-01-05T00:00:00+01:00</logDate>
        <logTime/>
      </entry>
    </entries>
  </protocol>
</revisionJobInfo>

```

13.6.7.10 REST System API

Create a thread dump

URL Pattern	/manage/system/info/threaddump
Method	GET
Parameters	

Content types	
Returns	a thread dump from the current server

Request:

```
curl -u rest:heiler -X GET http://localhost:1501/rest/V1.0/manage/system/info/threaddump
```

Response:

a thread dump in a common format which can be read by automated systems like fastthread.io

Create thread dumps for all servers

URL Pattern	/manage/system/info/threaddump/allServers
Method	GET
Parameters	
Content types	
Returns	a zip containing a thread dump for every server

Request:

```
curl -u rest:heiler -X GET http://localhost:1501/rest/V1.0/manage/system/info/threaddump/allservers
```

Response:

a zip with thread dumps in a common format which can be read by automated systems like fastthread.io

Echo Text

Simple function to test server availability, authentication and correct charset/encoding use.

URL Pattern	/manage/system/echo/{echoText}
Method	GET
Parameters	<code>echoText</code> : the text string which get echoes back.
Content types	application/json
Returns	the given echoText with the server internal charset/encoding.

Example:

Request:

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/manage/system/echo/helloWorld
```

Response:

```
helloWorld
```

Create token for Token Basic Authentication

Creates a token, which can be used to authenticate, instead of supplying username/password (for further details see chapter [REST Service Authentication](#) (see page 434)). The token is generated for the authenticated user of this REST call and is valid for one hour.

URL Pattern	/manage/system/security/token
-------------	-------------------------------

Method	POST
Parameters	<code>grant_type</code> : the type of token. " <code>authorization_code</code> " for an access token
Content types	application/json
Returns	A JSON Object, containing: <code>access-token</code> : the token <code>expires-in</code> : period of validity of the token (in seconds)

Example:

Request:

```
POST /rest/V1.0/manage/system/security/token
grant_type: authorization_code
```

Response:

```
{
  "access-token":
  "4ABE78F118D7068322B9A1AEB5B76C385C32777D52FB971FF70593C28CE94C8DB43BD5E7EF4EAB84DA8A
E435F231B5F274F3643A2B2527350861E38029A31D5CD826A28D68D70625E8D1F47...",
  "expires-in": 3600
}
```

Create token for Token Basic Authentication, based on a SAML Response

When using SAML for authentication (see chapter SAML Configuration), a token can be retrieved for the user contained in a SAML Response.

URL Pattern	/manage/system/security/tokenForSaml
Method	POST

Parameters	<p><code>grant_type</code> : the type of token. " <code>authorization_code</code> " for an access token</p> <p><code>saml_response</code> : Base64-Encoded SAML 2.0 Response (containing a SAML Assertion)</p>
Content types	application/json
Returns	<p>A JSON Object, containing:</p> <p><code>access-token</code> : the token</p> <p><code>expires-in</code> : period of validity of the token (in seconds)</p>

The token validity time can be configured via the preference `com.heiler.ppm.webservice.server/accessTokenExpirationTime.SAML` (in the server's `plugin_customization.ini`), and is one day by default.

Example:

Request:

```
POST /rest/V1.0/manage/system/security/tokenForSaml
grant_type: authorization_code

saml_response:
28CE94C8DB43BD5E7EF4EAB84DA8AE435F231B5F274F3643A2B2527350861E38029A31D5CD826A28D68D7
0625E8D1F47FFC79712AEC227AD53818203C0496EB6D38E6BFE4B68E...
```

Response:

```
{
  "access-token":
  "4ABE78F118D7068322B9A1AEB5B76C385C32777D52FB971FF70593C28CE94C8DB43BD5E7EF4EAB84DA8A
  E435F231B5F274F3643A2B2527350861E38029A31D5CD826A28D68D70625E8D1F47...",
  "expires-in": 86400
}
```

REST System API for License



Available since 7.0.04

Retrieve License Information

Retrieve license information containing maximum users and available

URL Pattern	/manage/system/license
Method	GET
Parameters	-
Content types	application/json
Media types	application/json
Result	The license object

Result

An object containing license information.

Properties of the returned object			
	Field	Data type	Description
	maxPimUsers	Integer	The maximum number of PIM users.
	maxCollaborative Users	Integer	The maximum number of collaborative users.
	languages	List of String	The names of all language locales that have been licensed, sorted ascending. The locales are formatted in the two conventional Java pattern - Language in ISO 639 + "_" + country code in ISO 3166-1 alpha-2, f.e. en_US.

13.6.7.11 REST Task API



This page does **not** describe the handling of special "Workflow Tasks". Workflow tasks are visualized in the same UI components than tasks, but have a fundamentally different behavior. For details on how to use Workflow tasks please see [REST Workflow Task \(see page 811\)](#)

The Rest Task API supports the creation, updating and content setting of PIM tasks.

- [Create Task \(see page 783\)](#)
 - [Content \(see page 784\)](#)
 - [Result \(see page 790\)](#)
- [Update Task \(see page 791\)](#)
 - [Content \(see page 791\)](#)
- [Get Task \(see page 791\)](#)
- [Delete Task \(see page 791\)](#)
- [Update Task Content \(see page 792\)](#)
 - [Content \(see page 792\)](#)
- [Get Task Content \(see page 793\)](#)
- [Workflow callback \(see page 793\)](#)
- [Information Page \(see page 794\)](#)
- [Examples \(see page 794\)](#)
 - [Create a task for all items in the catalog TOOLS \(see page 794\)](#)
 - [Create a task for supplier \(see page 795\)](#)
 - [Update a task: Change name and description \(see page 795\)](#)
 - [Update a task's content: Add another item \(see page 796\)](#)

Create Task

URL Pattern	<code>/manage/task</code>
Method	<code>POST</code>
Content types	<code>application/json</code>
Media types	<code>application/json</code>
Result	ENTITY_ITEM object of the new task.

Content

The content has to be a JSON object `TaskCreationProfile` which includes the properties listed below.

Field	Required	Datatype	Parameter description	
task	yes	Task	The task's initial properties. Following Task fields are mandatory: name	
content	no	ReportQuery	The report query which defines the content of the task.	
Properties of the Task object				
Field	Datatype	ReadOnly	Parameter description	Remarks
taskType	String		Type of the task, either "SingleTask" or "TaskGroup"	Mandatory. Error if invalid type is provided. Can only be changed for tasks without content.
dynamic	Boolean		Whether task is based on a dynamic query or has static content.	Default is static (false). Can only be changed using the appropriate Update Task Content <code>contentMode</code> .
entity	String		Entity identifier of the task's content type.	If task <i>content</i> is provided, <i>entity</i> must be set. Error if invalid entity. Can only be changed for tasks without content.
name	String		The name of the task. Ideally a meaningful short description of what is to do.	Only if feature 'Language specific Task data' is disabled. Mandatory.

Properties of the Task object				
Field	Datatype	Read Only	Parameter description	Remarks
description	String		Optional long description of the task.	Only if feature 'Language specific Task data' is disabled.
parent	ENTITY_ITEM		Parent task group, in case task shall belong to a group.	Error if parent task group does not exist. Error if set for SingleTask.
user	ENTITY_ITEM		The user the task will be assigned to.	At least one of the parameters <i>user</i> or <i>userGroup</i> or <i>supplier</i> have to be given. If none of them is given the creating user (see below "creationUser") will be used as assigned user by default. If <i>user</i> and <i>userGroup</i> are given the assignee <i>user</i> has priority. Error if <i>supplier</i> and <i>user/userGroup</i> are given. Error if <i>user</i> specified does not exist.
userGroup	ENTITY_ITEM		The user group the task will be assigned to.	One of the users of the given user group has to accept the task and will be the assigned user then. At least one of the parameters <i>user</i> or <i>userGroup</i> or <i>supplier</i> have to be given. If none of them is given the creating user (see below "creationUser") will be used as assigned user by default. If <i>user</i> and <i>userGroup</i> are given the assignee <i>user</i> has priority. Error if <i>supplier</i> and <i>user/userGroup</i> are given. Error if <i>userGroup</i> specified does not exist.

Properties of the Task object				
Field	Datatype	ReadOnly	Parameter description	Remarks
supplier	ENTITY_ITEM		The supplier the task will be assigned to.	<p>At least one of the parameters <i>user</i> or <i>userGroup</i> or <i>supplier</i> have to be given. If none of them is given the creating user (see below "creationUser") will be used as assigned user by default.</p> <p>Error if <i>supplier</i> and <i>user/userGroup</i> are given.</p> <p>Error if <i>supplier</i> specified does not exist.</p> <p>Error if <i>supplier</i> does not match the supplier of catalog used for task content (container in the case of workflow task).</p>
substitute	ENTITY_ITEM		The substitute replacing the assign user on the given escalation date/time if task is not completed yet.	
responsible	ENTITY_ITEM		The responsible user.	
creationUser	ENTITY_ITEM	yes	The user who created the task.	Set automatically
creationDate	DATE TIME (see page 436)	yes	The date/time the task has been created.	Set automatically

Properties of the Task object				
Field	Datatype	ReadOnly	Parameter description	Remarks
escalationDate	DATE TIME (see page 436)		The date/time the task escalates and the delegate is assigned replacing the originally assigned user.	
deadline	DATE TIME (see page 436)		The date/time the task is scheduled to be fulfilled. If deadline is reached, task will be assigned to responsible.	
finishEstimate	DATE TIME (see page 436)		The date/time the assigned user plans to fulfill the task (pure information).	
finishDate	DATE TIME (see page 436)		The date/time the task has been finished.	
progress	Integer		The progress of the task (pure information).	When creating the progress 0% is used. Error if given value does not exist in progress enum (0-5).
priority	Integer		The priority of that task (pure information).	When creating the priority "normal" is used. Error if given value does not exist in priority enum (0,1,2).

Properties of the Task object				
Field	Datatype	Read Only	Parameter description	Remarks
notificationLevel	Integer		The profile for email notification to be used with that task.	System wide default notification level is used when creating a task.
count	Integer	yes	Number of objects in task.	Set automatically
displayOrder	Integer		Display order for arranging task in UI.	Set automatically when creating a task.
accepted	Boolean	yes	Whether task has been accepted or not.	
acceptanceDate	DATE TIME (see page 436)	yes	The date/time when the task has been accepted by an assignee.	Set automatically when accepting a task.
workflowTaskId	Long		Id of the workflow associated with the task. (Hint: this is not related to InfaBPM workflows!)	Deprecated field. This has been used to combine Tasks with the old Workflow engine which is no longer in use.
complete	Boolean		Calculated parameter reflecting completion status.	If finishDate contradicts the intended completion status, finishDate is updated.

Properties of the Task object				
Field	Datatype	Read Only	Parameter description	Remarks
template	ENTITY_ITEM		UI template the task is bound to	For flexible task UI
workflowServiceEndpoint	String		The endpoint which is to be triggered on Informatica BPM side when a workflow task is marked as finished.	
workflowCorrelationId	String		The correlation id of the workflow instance which is to be triggered on InfaBPM side when a workflow task is marked as finished.	
workflowCommunicationMode	QUEUE / REST		The communication mode which can define message queue or REST communication to BPM	
workflowQueueId	String		The queue id from the server.properties which is defined by the syntax "queue.[queueId].name" which is used for the response message	

Properties of the Task object

Field	Datatype	ReadOnly	Parameter description	Remarks
taskLanguages	Map<String, String>		The language specific data of a task.	Only if feature 'Language specific Task data' is enabled

Properties of the TaskLang object (only if feature 'Language specific Task data' is enabled)

Field	Datatype	ReadOnly	Parameter description	Remarks
language	String		The language of the name and description. Valid values include all synonyms of Enum.Language.WithLanguageIndependent	
name	String		The name of the task. Ideally a meaningful short description of what is to do.	
description	String		Optional long description of the task.	

Result

An object reference of the new task.

Field	Data type	Description
id	Integer	The task id

Field	Data type	Description
label	String	The label of the task

Update Task

URL Pattern	/manage/task/{task-id}
Method	POST
Content types	application/json

Updates task properties (fields). To update the list of entities associated with the task use [Update Task Content](#) (see page 792) .

Content

The content has to be a Task JSON object (see above).

Get Task

URL Pattern	/manage/task/{task-id}
Method	GET
Media types	application/json
Result	Task object

Delete Task

URL Pattern	/manage/task/{task-id}
-------------	------------------------

Method	DELETE
Media types	application/json
Result	'true'/'false' string whether object could be deleted

Update Task Content

URL Pattern	/manage/task/{task-id}/content/{entity}
Method	POST
Parameters	contentMode: add, remove, set - default: set queryMode: static, dynamic - default: static.
Content types	application/json
Media types	application/json
Result	ENTITY_ITEM object of the updated task.

A task with static contents has a list of *entity items* (Products, Items etc) associated with it while a task with dynamic contents has an associated query. Setting task contents changes task *dynamic* property accordingly. Static contents can be adjusted by adding and removing items with the corresponding *contentMode* and static *queryMode*. Adding to and removing from a dynamic task is not possible. The corresponding task will be updated (which can be time consuming) before returning the result.

Content

The content has to be a valid ReportQuery JSON object (see above). Content of tasks created for supplier must contain objects from catalogs of assigned supplier only.

Get Task Content

URL Pattern	/manage/task/{task-id}/content
Method	GET
Parameters	updateTask: updates the task if set to true, by default - false.
Content types	-
Media types	application/json
Result	report result object

Returns report result object, which contains all information from internal report result + entity identifier.

If `updateTask` is set, corresponding task will be updated (which can be time consuming) before returning the result.

[List Management API](#) (see page 471) should be used to obtain actual report items, for example

```
http://localhost:1501/rest/V1.0/list/Article/byReportId?
reportId=<report_id>&datasource=PCM_MASTER
```

Workflow callback

If the `workflowServiceEndpoint` and `workflowCorrelationId` parameters are set, the application server will send a callback message to the Informatica BPM server as soon as the task gets marked as complete.

The callback message posts e.g. following XML body:

```
<taskCompletedRequest>
  <taskId>10146</taskId>
  <task>
    <taskType>SingleTask</taskType>
    <name>Test task callback</name>
```

```

    <priority>1</priority>
    <notificationLevel>1</notificationLevel>
    <creationDate>2015-10-21T14:38:40:580+0200</creationDate>
    <escalationDate/>
    <deadline/>
    <finishEstimate/>
    <finishDate/>
    <progress>0</progress>
    <dynamic>false</dynamic>
    <count>2</count>
    <entity>Article</entity>
    <displayOrder>3125</displayOrder>
    <creationUser>restuser</creationUser>
    <user>Administrator</user>
    <responsible>restuser</responsible>
    <accepted>true</accepted>
    <complete>true</complete>
  </task>
</taskCompletedRequest>

```

The <user> tag will carry the user who accepted the task before completion.

Information Page

A short version of the API description is available at </manage/task/info> .

Examples

Create a task for all items in the catalog *TOOLS*

Rest Client Java Code

```

EntityItemReference catalog = EntityItemReferenceFactory.createByIdentifier(
"TOOLS" );
ReportQuery reportQuery = new ReportQuery( "byCatalog" );
reportQuery.addParameterValue( "catalog", catalog );

Task task = new Task();
task.setName( "Verify all tools" );
task.setTaskType( TaskType.SINGLE.classifier() );
task.setEntity( "Article" );
TaskCreationProfile taskProfile = new TaskCreationProfile();
taskProfile.setTask( task );
taskProfile.setContent( reportQuery );

TaskRequest taskRequest = getRestClient().createTaskRequest();

```

```
EntityItemReference newTask = taskRequest.createTask( taskProfile );
Long taskId = Long.valueOf( newTask.getId() );
```

JSON

```
//POST to http://localhost:1501/rest/V1.0/manage/task
{
  "task":{"name":"Verify all tools.",
    "taskType":"SingleTask",
    "entity":"Article"
  },
  "content":{"identifier":"byCatalog",
    "parameterList":[{"key":"Catalog","value":"'TOOLS'"}]
  }
}
```

Create a task for supplier

Use POST request like `http://<server>:<port>/rest/V1.0/manage/task`

Create a task for Supplier

```
{
  "task":{
    "taskType" : "SingleTask",
    "name" : "Task for supplier created via Service API",
    "supplier" : "SUPPLIER_FOR_TASKS",
    "responsible" : "Administrator"
  },
  "content":{
    "identifier":"byCatalog",
    "parameterList":[{"key":"Catalog","value":"'CATALOG_OF_SUPPLIER_FOR_TAS
KS'"}]
  }
}
```

Update a task: Change name and description

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" "Content-Type: application/json" -X
POST http://localhost:1501/rest/V1.0/manage/task/1/
```

JSON

```
{
  "name": "Check short descriptions",
  "description" : "Check short descriptions in German and English"
}
```

Update a task's content: Add another item

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" "Content-Type: application/json" -X
POST http://localhost:1501/rest/V1.0/manage/task/1/content/Article?
contentMode=add&queryMode=static
```

JSON

```
{
  "identifier": "byItems",
  "parameterList": [
    {
      "key": "items",
      "value": "'I-3'@1"
    }
  ]
}
```

13.6.7.12 REST Workflow API

The Rest Workflow API allows to manage workflows, workflow status and process status. This API is used for the Informatica BPM workflow engine integration.

We strongly recommend to read the Business Process Management overview article in the knowledge base prior to this as it gives a good general overview on the workflow integration.

- [Create / Update Workflow Details](#) (see page 797)
- [Get Workflow Details](#) (see page 802)
- [Enter Workflow Process Status](#) (see page 804)
- [Leave Workflow Process Status](#) (see page 806)
- [End Workflow Process](#) (see page 808)
- [Get Workflow Process Status of an entity item per process](#) (see page 809)
- [Add error message to the BPM perspective protocol view](#) (see page 810)

Create / Update Workflow Details

Creates / updates a workflow definition.

General Info

URL Pattern	/manage/workflow
Method	POST
Parameters	-
Content types	application/json
Media types	application/json
Result	Workflow creation / update optional: Workflow status creation / update

Content

The content has to be a JSON object which includes the properties listed below.

Workflow

Field	Required	Default	Datatype	Parameter description	Example
identifier	yes	n/a	String	Unique identifier of the workflow. Has to include a version number in case of changes in the list of available workflow states	Workflow1 Workflow1-V2
label	no	n/a	String	Short description of the workflow and its purpose.	

Field	Required	Default	Datatype	Parameter description	Example
status	no	n/a	List of Status	List of status definitions	
version	no, but should be given	0.0	Decimal	This gives a version to the current workflow stored. If a version is stored once, it cannot be modified. To add or delete new status entries e.g. you have to send the workflow data with a newer version!	1.0

Status

Field	Required	Default	Datatype	Parameter description	Example
status	yes	n/a	String	Unique identifier of the status.	Status1
displayOrder	no	n/a	Integer	Position of the status within the workflow. If omitted, the order of the status in the request list is used	1
workflowTask	no	n/a	Task	Workflow task that should be generated for this workflow status. See the REST Workflow Task (see page 811) API for details	<pre>{ "container": "1", "containerEntityId": "MasterCatalog", "entity": "Article" }</pre>

Field	Required	Default	Datatype	Parameter description	Example
rejectDecision	no	n/a	Workflow Decision	WorkflowDecision containing all reject possibilities and also the content for the corresponding reject dialog that will be shown to the user. Relevant for approval workflows where an user can approve/reject at a specific point in the workflow process.	<pre> { "label": "Title of the dialog", "singleChoice": false, "decisions": [{ "label": "Classification", "id": "executeClassification" }, { "label": "Spanish translation", "id": "executeSpanishTr anslation" }] }</pre>

WorkflowDecision

Field	Required	Default	Datatype	Parameter description	Example
label	yes	n/a	String	The text for the title of the dialog	

Field	Required	Default	Datatype	Parameter description	Example
<code>singleChoice</code>	no	false	Boolean	Defines if the children workflow decisions are interpreted in the dialog as radio button group (true), or check boxes (false)	
<code>id</code>	no	n/a	String	This identifier will be transmitted to the Informatica BPM if the corresponding decision is marked as rejected by the user. The existence of this identifier can then be used on Informatica BPM side to enter a different path in the workflow process for example.	
<code>decisions</code>	no	n/a	<code>ArrayList<WorkflowDecision></code>	Possibility of defining a hierarchical reject dialog, combining for example radio button groups with sub ordinary check boxes.	

Result

Properties

Field	Data type	Description
<code>List</code>	<code>ResultSummary[]</code>	<p>Counters indicating the number of created, updated and failed entities.</p> <p>NOTE: Successful Workflow status creations are reported as updated entities</p>

Example

Create workflow

Request

POST http://localhost:1512/rest/V1.0/manage/workflow

```
{
  "identifier": "Workflow1",
  "label": "First Workflow",
  "status": [
    { "status": "Status1",
      "workflowTask" : { "container" : "1", "containerEntityId" : "MasterCatalog",
        "entity" : "Article" }
    },
    { "status": "Status2" },
  ]
}
```

Result

```
{
  createdCounter: 2
  updatedCounter: 2
  failedCounter: 0
}
```

Create workflow with tasks for supplier and P360 user

Use POST request like `http://<server>:<port>/rest/V1.0/manage/workflow`

Create workflow with tasks for supplier and P360 user

```
{
  "identifier": "Workflow with Supplier Task",
  "label": "Workflow with Supplier Task",
  "status": [
    {
      "status": "Enrich data",
      "displayOrder": 1,
      "workflowTask": [
        {
          "name": "Enrich data: Task for Supplier",
          "entity": "Article",
          "supplier": "SUPPLIER_FOR_TASKS",
          "workflowServiceEndpoint" :
            "Enrich_Data_Completed",
          "container" : { "id" :
            "'CATALOG_OF_SUPPLIER_FOR_TASKS'" }
        }
      ]
    }
  ]
}
```

```

        }
      ]
    },
    {
      "status": "Approve supplier data",
      "displayOrder": 2,
      "workflowTask": [
        {
          "name": "Approve supplier data: Task for P360 User",
          "entity": "Article",
          "user": "P360_USER",
          "workflowServiceEndpoint" :
"Approve_Data_Completed",
          "container" : { "id" :
"'CATALOG_OF_SUPPLIER_FOR_TASKS'" }
        }
      ]
    }
  ]
}

```

Get Workflow Details

Get a workflow definition.

General Info

URL Pattern	/manage/workflow/{workflowIdentifier}
Method	GET
Parameters	-
Content types	application/json
Media types	application/json
Result	Workflow details Workflow status list

Parameters

Workflow

Field	Required	Default	Datatype	Parameter description	Example
<code>workflowIdentifier</code>	yes	n/a	String	Unique identifier of the workflow. Has to include a version number in case of changes in the list of available workflow states	Workflow1 Workflow1-V2
<code>resolveWorkflowTasks</code>	no	false	boolean	If set to true, the workflow task information is resolved, too	
<code>includeObsoletes</code>	no	false	boolean	If true, status entries which are not available in the newest version of the workflow, will be resolved, too. Otherwise obsolete status entries will not be shown	

Example

Request

```
GET http://localhost:1512/rest/V1.0/manage/workflow/Workflow1
```

Result

```
{
  "identifier": "Workflow1",
  "label": "First Workflow",
  "status": [
    { "status": "Status1", "displayOrder": 1 },
    { "status": "Status2", "displayOrder": 2 }
  ]
}
```

Enter Workflow Process Status

Enters a specific status within the workflow process. If a WorkflowTask is related to the status, this call adds the entity to it.



Starting from 10.5.0.01 EBF3 customers still using the *deprecated* request with only query parameters & no JSON/XML payload will need to explicitly add the request header - "**Content-Type: application/octet-stream**" in the requests.

General Info

URL Pattern	/manage/workflow/status/enter
Method	POST
Parameters	-
Content types	application/json
Media types	application/json
Result	success result

Content

The content has to be a JSON object which includes the properties listed below.

Field	Required	Default	Datatype	Parameter description	Example
processId	yes	n/a	String	Unique identifier of the workflow instance. Within Informatica BPM workflow instances are called processes.	'1234567'

Field	Required	Default	Datatype	Parameter description	Example
<code>workflowId</code>	yes	n/a	String	Unique identifier of the workflow itself	'workflowXYZ'
<code>status</code>	yes	n/a	String	A status name	'stateA', 'stateB'
<code>entity</code>	yes	n/a	String	The entity name	'Article', 'Variant' or 'Product2G'
<code>itemId</code>	yes	n/a	String	The itemids in the typical service API syntax	'1234@1' (see page 796)
<code>resetAcceptedUser</code>	no	true	boolean	If the item was previously accepted by an user, the user is reset and the item available to the usergroup again (since 8.0.01)	'true'
<code>hint</code>	no	n/a	String	A comment field, which can be sent to the server, when this status has been entered	

Example

Request

POST http://localhost:1512/rest/V1.0/manage/workflow/status/enter

```
{
  "processId": "4711",
  "workflowId": "Workflow1",
  "status": "Status1",
  "entity": "Article",
  "hint" : "Comment for Status1",
```

```
{
  "itemId": [ "123@1" ]
}
```

Result

```
{
  "successful":true,
  "canNotFindItem":false,
  "canNotFindWorkflowId":false,
  "workflowTaskNotFound":false
}
```

Leave Workflow Process Status

Leaves a specific status within the workflow process. This call is only required in case no Workflow Task is assigned to the status. In this case PIM leaves the workflow process status implicitly.



Starting from 10.5.0.01 EBF3 customers still using the *deprecated* request with only query parameters & no JSON/XML payload will need to explicitly add the request header - "**Content-Type: application/octet-stream**" in the requests.

General Info

URL Pattern	/manage/workflow/status/leave
Method	POST
Parameters	-
Content types	application/json
Media types	application/json
Result	success result

Content

The content has to be a JSON object which includes the properties listed below.

Field	Required	Default	Datatype	Parameter description	Example
processId	yes	n/a	String	Unique identifier of the workflow instance	'1234567'
status	yes	n/a	String	A status name	'stateA', 'stateB'
entity	yes	n/a	String	The entity name	'Article', 'Variant' or 'Product2G'
itemId	yes	n/a	String	The itemids in the typical service API syntax	'1234@1' (see page 796)
hint	no	n/a	String	A comment field, which can be sent to the server, when this status has been left	

Example

Request

POST http://localhost:1512/rest/V1.0/manage/workflow/status/leave

```
{
  "processId": "4711",
  "status": "Status1",
  "entity": "Article",
  "hint" : "Comment for Status1",
  "itemId": [ "123@1" ]
}
```

Result

```
{
  "successful" : true,
```

```
"canNotFindItem" : false
}
```

End Workflow Process

Ends a workflow process and soft deletes all related status entries

General Info

URL Pattern	/manage/workflow/process
Method	DELETE
Parameters	-
Content types	application/json
Media types	application/json
Result	-

Parameters

Field	Required	Default	Datatype	Parameter description	Example
processId	yes	n/a	String	Unique identifier of the workflow instance	'1234567'

Example

Request

DELETE

```
http://localhost:1512/rest/V1.0/manage/workflow/process?processId=9375
```

Get Workflow Process Status of an entity item per process

Resolves information about the current status entries related to a given workflow process

General Info

URL Pattern	/manage/workflow/status
Method	GET
Parameters	-
Content types	application/json
Media types	application/json
Result	success result and status information

Parameters

Field	Required	Default	Datatype	Parameter description	Example
processId	yes	n/a	String	Unique identifier of the workflow instance	'1234567'
entity	yes	n/a	String	The entity name	'Article', 'Variant' or 'Product2G'

Field	Required	Default	Datatype	Parameter description	Example
itemId	yes	n/a	String	The itemids in the typical service API syntax	'1234@1' (see page 796)

Example

Request

GET

```
http://localhost:1512/rest/V1.0/manage/workflow/status?
itemId=30025@1&entity=Article&processId=9375
```

Result

```
{
  "noItemFound": false,
  "statusList": [
    {
      "status": "teststatusX",
      "visits": 1,
      "duration": 0,
      "startTime": "2015-04-17T16:41:41:950+0200",
      "endTime": "2015-04-17T16:41:41:957+0200"
    }
  ]
}
```

Add error message to the BPM perspective protocol view

This can be used within catch exception events within a workflow, to add information to the bpm perspectives protocol log view

General Info

URL Pattern	/manage/workflow/message
-------------	--------------------------

Method	PUT
Parameters	-
Content types	application/json
Media types	application/json
Result	HTTP 201

Parameters

Field	Required	Default	Datatype	Parameter description	Example
message	yes	n/a	String	Error message	'Workflow 1 - Process 1234 - Assign task to user xyz failed'
severity	no	ERROR	String	The entity name	ERROR, INFO, WARNING, CANCEL, OK

REST Workflow Task

This page describes the data structure of a workflow task as it's used in combination with the [REST Workflow API](#) (see page 796). A workflow task has a limited number of attributes compared to a static task as most of it's attributes are defined by the workflow status to which it belongs.

- [Example](#) (see page 811)
- [WorkflowTask](#) (see page 813)
- [Task Callback](#) (see page 815)
 - [Callback Payload for Finished state](#) (see page 815)
 - [Callback Payload when the process must be terminated](#) (see page 820)

Example

This is an example workflow which has multiple workflow tasks. In this example only typical and mandatory attributes are provided for the workflow task. Please see the table below for the full list of attributes which can be provided here.

```

<rest:RESTRequest
  xmlns:rest="http://schemas.activebpel.org/REST/2007/12/01/aeREST.xsd">
  <rest:method>POST</rest:method>
  <rest:pathInfo>rest/V1.0/manage/workflow</rest:pathInfo>
  <rest:headers>
    <rest:header name="authorization" value="{ $Authorization }"/>
    <rest:header name="bpm-correlation-id" value="{ $processId }"/>
    <rest:header name="Content-Type" value="application/json"/>
    <rest:header name="accept" value="application/json"/>
  </rest:headers>
  <rest:payload contentType="application/json">
  {{
    "identifier": "{ $workflowName }",
    "label": "{$workflowName}",
    "version": "{$workflowVersion}",
    "status": [
      {{ "status": "Classification",
        "workflowTask" : {{ "container" : "'{$catalogIdentifier}'", "entity" :
"Article", "template" : "Item classification UI", "workflowServiceEndpoint" :
"ApprovalWorkflowTaskClassification-Finish", "userGroup" : "AllRights" }} }},
      {{ "status": "TranslateSpanish",
        "workflowTask" : {{ "container" : "'{$catalogIdentifier}'", "entity" :
"Article", "template" : "Item translation UI", "workflowServiceEndpoint" :
"ApprovalWorkflowTaskTranslateSpanish-Finish", "userGroup" : "AllRights" }} }},
      {{ "status": "TranslateItalian",
        "workflowTask" : {{ "container" : "'{$catalogIdentifier}'", "entity" :
"Article", "template" : "Item translation UI", "workflowServiceEndpoint" :
"ApprovalWorkflowTaskTranslateItalian-Finish", "userGroup" : "AllRights" }} }},
      {{ "status": "TranslateFrench",
        "workflowTask" : {{ "container" : "'{$catalogIdentifier}'", "entity" :
"Article", "template" : "Item translation UI", "workflowServiceEndpoint" :
"ApprovalWorkflowTaskTranslateFrench-Finish", "userGroup" : "AllRights" }} }},
      {{ "status": "Approve",
        "workflowTask" : {{ "container" : "'{$catalogIdentifier}'", "entity" :
"Article", "template" : "Item approve UI", "workflowServiceEndpoint" :
"ApprovalWorkflow-Finish", "userGroup" : "AllRights" }},
        "rejectDecision": {{ "label": "Title of Reject Dialog", "singleChoice": true,
          "decisions": [{{ "label": "Classification", "id": "executeClassification"
}},
          {{ "label": "Translation", "singleChoice": false, "decisions":
            [{{ "label": "Translate texts into French", "id":
"executeTranslationFrench" }},
            {{ "label": "Translate texts into Spanish", "id":
"executeTranslationSpanish" }},
            {{ "label": "Translate texts into Italian", "id":
"executeTranslationItalian" }}
            ]
          }}
        ]
      }}
    ]
  }}
  }}

```



```

    ]
  }}
</rest:payload>
  <rest:locales>
    <rest:locale country="US" lang="en">en_US</rest:locale>
  </rest:locales>
</rest:RESTRequest>

```

WorkflowTask

Field	Datatype	Description
container	ENTITY_ITEM	The container, items of the workflow task are contained in. In case of Products, Variants or Items this would be a Catalog for example.
entity	String	Entity identifier of the task's content type.
workflowServiceEndpoint	String	The endpoint which is to be triggered on Informatica BPM side when a workflow task is marked as finished.
workflowCommunicationMode	QUEUE/REST	The communication mode which can define message queue or REST communication to BPM
workflowQueueId	String	The queue id from the server.properties which is defined by the syntax "queue.[queueid].name" which is used for the response message

Field	Datatype	Description
<code>userGroup</code>	ENTITY_ITEM	<p>The user group the task will be assigned to.</p> <p>Error if specified <i>userGroup</i> does not exist.</p> <p>At least one of the parameters <i>user</i> or <i>userGroup</i> or <i>supplier</i> have to be given.</p> <p>If none of them is given the creating user (see below "creationUser") will be used as assigned user by default. If <i>user</i> and <i>userGroup</i> are given the assignee <i>user</i> has priority.</p> <p>Error if <i>supplier</i> and <i>user/userGroup</i> are given.</p>
<code>user</code>	ENTITY_ITEM	<p>The user the task will be assigned to.</p> <p>Error if specified <i>user</i> does not exist.</p> <p>At least one of the parameters <i>user</i> or <i>userGroup</i> or <i>supplier</i> have to be given.</p> <p>If none of them is given the creating user (see below "creationUser") will be used as assigned user by default. If <i>user</i> and <i>userGroup</i> are given the assignee <i>user</i> has priority.</p> <p>Error if <i>supplier</i> and <i>user/userGroup</i> are given.</p>
<code>supplier</code>	ENTITY_ITEM	<p>The supplier the task will be assigned to.</p> <p>Error if specified <i>supplier</i> does not exist.</p> <p>Error if <i>supplier</i> does not match the supplier of catalog used for task content (container in the case of workflow task).</p> <p>At least one of the parameters <i>user</i> or <i>userGroup</i> or <i>supplier</i> have to be given. If none of them is given the creating user (see below "creationUser") will be used as assigned user by default.</p> <p>Error if <i>supplier</i> and <i>user/userGroup</i> are given.</p>
<code>template</code>	ENTITY_ITEM	<p>Identifier of the UI template the task is bound to (aka Flexible Task UI).</p> <p>Error in case the template for this identifier does not exist</p>

Field	Datatype	Description
maximumDuration	Long	The maximum amount of time in milliseconds an item residing in a workflow task can be worked on (after accepting) until the deadline for this item is reached. No functional impact as of now (only for information purposes).
name	String	The name of the task. Ideally a meaningful short description of what is to do.
description	String	Optional long description of the task.
priority	Integer	The priority of that task (pure information). When creating the priority "normal" is used. Error if given value does not exist in priority enumeration (0,1,2).

Task Callback

As soon as the user decides that the task is finished for a specific list of items of a workflow task, Informatica BPM will be called from Product 360 so BPM can move the affected workflow instances to the next step.

For this, you need to provide the `workflowServiceEndpoint` when the workflow task is created, and the `processId` when the item will enter the corresponding workflow status. Informatica Product 360 will call BPM for each workflow instance of this status so the workflow can be moved forward to the next action. Please note that Informatica BPM will be called for each process instance for this workflow.

Additionally to this pure "signal" we provide some payload data on the current process status the item was in when the user finished the task for the item. This way the workflow can decide what to do next without having to fetch this again from Product 360.

Callback Payload for Finished state

Here is an example of the payload which is given in the callback. Note that either user, or userGroup or supplier is given here. (TODO: correct!?)

Note: This extended callback payload is available with 8.1.1.00.05 and after.

```
<rest:RESTRequest xmlns:rest="http://schemas.activebpel.org/REST/2007/12/01/aeREST.xsd">
```

```

<rest:method>POST</rest:method>
<rest:headers>
  <rest:header name="bpm-correlation-id" value="8002"/>
  <rest:header name="connection" value="keep-alive"/>
  <rest:header name="host" value="localhost:8080"/>
  <rest:header name="Content-Length" value="1547"/>
  <rest:header name="accept" value="text/html, image/gif, image/jpeg, *; q=.2, */
*; q=.2"/>
  <rest:header name="Content-Type" value="text/xml"/>
</rest:headers>
<rest:payload contentType="text/xml">
  <decisions>
    <decision id="p360.bpm.finish"/>
    <processStatusInfo>
      <processIdentifier>8002</processIdentifier>
      <itemEntity>Article</itemEntity>
      <item>2401@1</item>
      <statusEntry>
        <attribute>
          <name>workflow</name>
          <value>905</value>
        </attribute>
        <attribute>
          <name>status</name>
          <value>Test Status1</value>
        </attribute>
        <attribute>
          <name>startTime</name>
          <value>2018-10-02T14:50:38:440+0200</value>
        </attribute>
        <attribute>
          <name>lastStartTime</name>
          <value>2018-10-02T14:50:38:443+0200</value>
        </attribute>
        <attribute>
          <name>user</name>
          <value>100</value>
        </attribute>
        <attribute>
          <name>userGroup</name>
        </attribute>
        <attribute>
          <name>supplier</name>
        </attribute>
        <attribute>
          <name>endTime</name>
          <value>2018-10-02T14:51:16:486+0200</value>
        </attribute>
        <attribute>
          <name>duration</name>
          <value>0</value>
        </attribute>
      </statusEntry>
    </processStatusInfo>
  </decisions>
</rest:payload>

```

```

    <attribute>
      <name>visitCounter</name>
      <value>1</value>
    </attribute>
    <attribute>
      <name>errorMessage</name>
      <value>No error</value>
    </attribute>
    <attribute>
      <name>acceptDate</name>
      <value>2018-10-02T14:51:08:521+0200</value>
    </attribute>
    <attribute>
      <name>deadline</name>
      <value>2018-10-02T14:51:08:440+0200</value>
    </attribute>
    <attribute>
      <name>creationDate</name>
      <value>2018-10-02T14:50:38:480+0200</value>
    </attribute>
    <attribute>
      <name>creationUser</name>
      <value>300</value>
    </attribute>
    <attribute>
      <name>modificationDate</name>
      <value>2018-10-02T14:51:08:520+0200</value>
    </attribute>
    <attribute>
      <name>modificationUser</name>
      <value>100</value>
    </attribute>
    <attribute>
      <name>hint</name>
    </attribute>
  </statusEntry>
</processStatusInfo>
</decisions>
</rest:payload>
<rest:ssl>false</rest:ssl>
<rest:contextPath>/active-bpel</rest:contextPath>
<rest:requestURI>/active-bpel/services/REST/TestStatus1-Finished</rest:requestURI>
<rest:locales>
  <rest:locale country="US" lang="en">en_US</rest:locale>
</rest:locales>
</rest:RESTRequest>

```

Field	Datatype	Description
processIdentifier	String	The process identifier, aka correlationId of the workflow process
itemEntity	String	The repository entity of the item which has been finished (e.g. Article, or Product2G etc.)
item	ENTITY_ITEM	The object for which the workflow task has been finished in the String Syntax of entity items
statusEntry	StatusEntry[]	The status entry object of the status the user just finished. Note: Although the model defines an array here, it can actually be only a single status entry each status represents a task the user finished working on.

The StatusEntry object

Field	Data Type	Description
workflow	ENTITY_ITEM (in String syntax)	The workflow object this status entry belongs to
status	String	The identifier of the status
user	ENTITY_ITEM (in String syntax)	The user which finished the work on the item. It's either a supplier, or a user, never both.

The StatusEntry object		
Field	DataType	Description
supplier	ENTITY_ITEM (in String syntax)	The supplier which finished the work on the item. It's either a supplier, or a user, never both.
errorMessage	String	
hint	String	Optional hint provided by the user when finishing the work. Example would be an approval task in which the approving user gives a hint as to what is not correct.
startTime	Timestamp	The first time the item started to be part of the workflow task. In other words, the first time the item entered this workflow state
lastStartTime	Timestamp	The last time the item started to be part of the workflow task. In other words, the last time the item entered this workflow state
endTime	Timestamp	The last time when the item finished this workflow task. In other words, the timestamp the item left this workflow state the very last
acceptDate	Timestamp	The time the user accepted the workflow task for this item the last time
deadline	Timestamp	The calculated deadline of the item in this workflow task. Based on the time the item entered the task (startTime) and the given maximumDuration for this status. (TODO: is this correct?)

The StatusEntry object

Field	DataType	Description
visitCounter	Long	The number of times this state has been entered
duration	Long	The total duration the item was part of this workflow state. This means for each recurring enter/leave the time is summed up here.

Typically a workflow is designed to not trust the users very much. In case a data quality result will trigger the item to be part of a manual task, it's normal that the workflow will check the data quality again after the task has been finished. In case the data quality is still not correct, the workflow will loop back and enter the same workflow task again. So a user which "just finishes" a task, without actually solving the data quality issue of the item, will get this item back.

In other words, the same item can be multiple time in the same status of the same workflow instance. Therefore we do record how many times the item was in this state (`visitCounter`) and we sum up the total `duration` of the item in this state (duration).

Callback Payload when the process must be terminated

In case a workflow id deleted in Product360 while there are still instances running, or because of other scenarios, Product 360 will contact BPM with a predefined terminate process endpoint. The identifier of this endpoint is `P360-TerminateProcessInstance` .

Even in this case, in which the user did not finish a task by himself, but the system terminated the workflows we provide as much information as possible to the workflow.

Field	Datatype	Description
processIdentifier	String	The process identifier, aka <code>correlationId</code> of the workflow process

Field	Datatype	Description
item	ENTITY_ITEM	The object for which the workflow task has been finished
statusEntry	StatusEntry[]	An array of status entry objects of all the status the item was currently in at the time the workflow got terminated. In contrary to the payload when the user finishes his work for a specific status (above) this payload contains the entries of all status - as all of them terminated at once. For details on the StatusEntry object please see above

13.6.8 REST Enumeration API

The Enumeration API provides read-only access to all enumerations in Product 360. The enum entry keys are used in field values as well as for the qualification of fields.

- [All Enumerations](#) (see page 821)
 - [Result](#) (see page 822)
- [Meta Data and Enumeration Entries](#) (see page 822)
 - [Result](#) (see page 823)
- [Examples](#) (see page 825)
 - [Retrieving all enumerations](#) (see page 825)
 - [Retrieving the entries of enumeration Enum.Languages](#) (see page 825)
 - [Retrieving the entries of enumeration Enum.SupplierWithMainSupplier](#) (see page 826)

13.6.8.1 All Enumerations

Returns the list of all available enumerations.

URL Pattern	/enum/info
Method	GET
Parameters	No parameters are available

Media type	text/html, application/json, application/xml
Result	A list of all enumerations defined in the repository

Result

For each enumeration the following properties are returned:

Key	Data type	Description
identifier	String	Unique identifier of the enumeration
dataType	String	Data type of the enumeration entry's key, see also REST Datatypes (see page 435).
name	String	Localized name of the enumeration
description	String	Localized description of the enumeration

13.6.8.2 Meta Data and Enumeration Entries

Returns the entries of the specified enumeration.

URL Pattern	/enum/{enumeration-identifier}
Method	GET

Parameters	Parameter	Required	Default	Datatype	Description
	container	no	none	ENTITY_ITEM	<p>In case the enum requires a container it must be specified with this parameter and the ENTITY_ITEM syntax. So either with the internal ID like</p> <pre>container=4711</pre> <p>or the identifier like</p> <pre>container='MyLookup'</pre> <p>If the parameter is missing but the enumeration needs one, a corresponding status and error message is returned.</p>
Media type	text/html, application/json, application/xml				
Result	The meta data of the enumeration (like the localized name) and all entries of the enumeration				

Result

The following properties are returned for an enumeration:

Enumeration properties			
	Field	Data type	Description
	identifier	String	Unique identifier of this enumeration

	name	String	Localized name of this enumeration
	description	String	Localized description of this enumeration
	dataType	String	Data type of the enumeration entry's key, see also REST Datatypes (see page 435).
	entries	Array	List of all entries of this enumeration

Each enumeration entry has the following properties:

Properties of a member of <code>entries</code>			
	Field	Data type	Description
	label	String	Unique label of this enumeration entry
	key	String or Object	<p>Unique key of this enumeration entry, not necessarily human readable.</p> <p>In case the key is of type ENTITY_ITEM, the key is represented as proxy object containing the properties <code>id</code>, <code>url</code> and <code>label</code>. If XML is specified as output format, a proxy object is embedded in the tag <code>object</code>.</p> <p>When creating a qualified field, this value should be used. If it is a proxy object, the value of the property <code>id</code> should be used.</p>
	externalCode	String	External code of the enumeration entry.

	keySynonyms	Array of Strings	List synonyms for the entry
--	-------------	------------------	-----------------------------

13.6.8.3 Examples

Retrieving all enumerations

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/enum/info
```

The following JSON object is returned:

```

1  {
2    "enumerations": [
3      {
4        "identifier": "Enum.Language",
5        "dataType": "INTEGER",
6        "name": "All languages",
7        "description": ""
8      },
9      ...
10 ]}
```

Rest Client Java Code

```
EnumerationInfoRequest enumInfoRequest = restClient.createEnumerationInfoRequest();
EnumerationInfos allEnumInfos = enumInfoRequest.getAllEnumerations();
for (EnumerationInfo enumInfo : allEnumInfos)
{
    // ...
}
```

Retrieving the entries of enumeration Enum.Languages

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/enum/Enum.Language
```

The following JSON object is returned:

```

1  {
2      "identifier": "Enum.Language",
3      "name": "All languages",
4      "description": "",
5      "dataType": "INTEGER",
6      "entries": [
7          {
8              "label": "Italian",
9              "key": "16",
10             "externalCode": "ita",
11             "synonyms": [
12                 "italian",
13                 "it",
14                 "ita",
15                 "ita_it"
16             ]
17         },
18         ...
19     ]
20 }
```

Rest Client Java Code

```

EnumerationRequest enumRequest = restClient.createEnumerationRequest();
Enumeration enumeration = enumRequest.getEnumeration( "Enum.Language" );
List< EnumerationEntry > enumEntries = enumeration.getEntries();
```

Retrieving the entries of enumeration Enum.SupplierWithMainSupplier

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/enum/Enum.SupplierWithMainSupplier
```

The following JSON object is returned:

(Note that the keys are in this case returned as proxy objects and the special object *<main supplier>* does not really exist but is a placeholder for the main supplier as specified within the HPM.)

```


1  {
2      "identifier": "Enum.SupplierWithMainSupplier",
3      "name": "Suppliers (inc. main supplier)",
4      "description": "",
5      "dataType": "ENTITY_ITEM",
```

```

6      "entries": [
7          {
8              "label": "<Main supplier>",
9              "key": {
10                 "id": "-10",
11                 "label": ""
12             },
13             "externalCode": "",
14             "synonyms": []
15         },
16         {
17             "label": "TestProvider",
18             "key": {
19                 "id": "101",
20                 "label": "TestProvider"
21             },
22             "externalCode": "TestProvider",
23             "synonyms": []
24         },
25         {
26             "label": "Heiler Product Manager",
27             "key": {
28                 "id": "3",
29                 "label": "Heiler Product Manager"
30             },
31             "externalCode": "Heiler Product Manager",
32             "synonyms": []
33         }
34     ]
35 }

```

13.6.9 REST Media API

 The REST Media API provides access to all kinds of media assets in the Product Manager.

- [Single Media Asset File](#) (see page 828)
- [Preview of Media Asset File](#) (see page 828)
- [Media Asset File Location](#) (see page 829)
- [Upload single Media Asset File](#) (see page 829)
- [Examples](#) (see page 830)
 - [Retrieve single media asset file in 36dpi JPEG quality with identifier D123456](#) (see page 830)
 - [Retrieve media asset file preview](#) (see page 831)
 - [Retrieve media asset file location](#) (see page 831)
 - [Upload media asset file to media asset server](#) (see page 831)

13.6.9.1 Single Media Asset File

URL Pattern	<code>/media/{quality/derivativeDefinition}/{identifier}</code>	
Method	GET	
Parameters	<code>lastModified</code>	DATETIME
Content types	application/octet-stream	
Returns	the media asset file as binary data stream	

13.6.9.2 Preview of Media Asset File

URL Pattern	<code>/media/{identifier}/preview</code>	
Method	GET	
Parameters	<code>size</code>	All allowed values: small medium large
Content types	image/jpg	
Returns	the media asset preview as binary jpeg	

13.6.9.3 Media Asset File Location

URL Pattern	/media/{quality/derivativeDefinition}/{identifier}
Method	GET
Parameters	-
Content types	text/plain
Returns	the URL to the media asset file

13.6.9.4 Upload single Media Asset File

To upload a local image to the media asset server, the following two steps should be performed:

1. call the [REST File API \(see page 738\)](#) to upload a local file to the PIM storage, which returns a result contains id for the uploaded file.
2. call the following REST API to add the uploaded file to the media manager. The path parameter "fileId" is the id of the returned file object of the first step.

URL Pattern	/media/{fileId}	
Method	POST	
Parameters	categoryId	optional parameter indicates the id of the category
Content types	text/plain	
Returns	the identifier of the uploaded media asset file	



The query parameter "categoryId" is optional, it indicates the id of the category to which the added media asset file should belong. If it is not given, the media asset file will be assigned to the default category.

13.6.9.5 Examples

Retrieve single media asset file in 36dpi JPEG quality with identifier D123456

Rest Call

```
curl -u rest:heiler -H "Accept: application/octet-stream" -X GET http://localhost:1501/rest/V1.0/media/JPEG 36dpi/D123456
```

Rest Client Java Code

```
MediaRequest mediaAssetRequest = restClient.createMediaRequest();

InputStream result = mediaAssetRequest.getMediaAsset( "D123456", "JPEG 36dpi", null );
```

Return media asset file only if modified after given time (otherwise empty stream is returned):

Rest Call

```
curl -u rest:heiler -H "Accept:application/octet-stream" -X GET http://localhost:1501/rest/V1.0/media/JPEG 36dpi/D123456?lastModified=2012-09-26T15:33:12
```

Rest Client Java Code

```
MediaRequest mediaRequest = restClient.createMediaRequest();

Timestamp lastModified = new Timestamp( calendar.getTime().getTime() );

InputStream result = mediaRequest.getMediaAsset( "D123456", "JPEG 36dpi", lastModified );
```

Retrieve media asset file preview

Rest Call

```
curl -u rest:heiler -H "Accept: image/jpg" -X GET http://localhost:1501/rest/V1.0/media/D123456/preview?size=small
```

Rest Client Java Code

```
MediaRequest mediaRequest = restClient.createMediaRequest();

PreviewSize size = PreviewSize.BIG;

InputStream result = mediaAssetRequest.getMediaAssetPreview( "D123456", size );
```

Retrieve media asset file location

```
curl -u rest:heiler -H "Accept: text/plain" -X GET http://localhost:1501/rest/V1.0/media/JPEG 36dpi/D123456
```

Rest Client Java Code

```
MediaRequest mediaRequest = restClient.createMediaRequest();

URL mediaAssetUrl = mediaRequest.getMediaAssetLocation( "D123456", "JPEG 36dpi");
```

Upload media asset file to media asset server

call [REST File API](#) (see page 738) to upload a local file to the PIM storage

```
curl -u rest:heiler -H "Accept: application/json" -H "Content-Type:application/octet-stream" --data-binary "@Test.jpg" -X POST http://localhost:1501/rest/V1.0/manage/file?originalFilename=Test.jpg
```

Rest Client Java Code

```
FileUploadRequest fileUploadRequest = restClient.createFileUploadRequest();
```

```
FileReference result = fileUploadRequest.uploadFile( "Test.jpg", inputStream );
```

An object including an ID and an URL is returned:

```
{
  "id": "feb3cd45-d27e-4700-912e-26014512a6e3",
  "originalFilename": "Test.jpg"
}
```

Then call following Rest Media API to add the uploaded file to the media asset server(in the default category) .

Rest Call

```
curl -u rest:heiler -H "Accept: text/plain" -X POST http://localhost:1501/rest/V1.0/media/feb3cd45-d27e-4700-912e-26014512a6e3
```

Rest Client Java Code

```
MediaRequest mediaRequest = restClient.createMediaRequest();

String mediaAssetIdentifier = mediaRequest.addMediaAsset( "feb3cd45-  
d27e-4700-912e-26014512a6e3");
```

Or call the same API with query parameter "categoryId" to add the uploaded file in desired category

```
curl -u rest:heiler -H "Accept: text/plain" -X POST http://localhost:1501/rest/V1.0/  
media/feb3cd45-d27e-4700-912e-26014512a6e3?  
categoryId=0099000000000000000000000000000000000000000000000000000000000000  
0000000000000000000000000000000000000000
```

Rest Client Java Code

[illegible]

13.6.9.6 REST List API for MediaAssetFile entity

i The REST List API for MediaAssetFile entity provides searching, reporting, navigation, deletion and finally result listing of MediaAssetFile like other general objects in system.

Since PIM 8.0 the MediaAssetFile entity is redefined for the media asset object, but currently it is only supported by PIM Core with Media Manager.

The data access for the MediaAssetFile entity is not per PIM database, but per the connector API, so that the corresponding persistence logic like fragments, mediators and reports are adjusted in a special way, therefore it is meaning to introduce the REST List API for that with this additional chapter.

- [Adjustments for MediaAssetFile entity](#) (see page 833)
 - [Internal id](#) (see page 833)
 - [Special reports](#) (see page 833)
 - [byCategory](#) (see page 833)
 - [bySearch](#) (see page 834)
 - [Search with query](#) (see page 835)
 - [Search with mediaQuery](#) (see page 835)
- [Examples](#) (see page 835)
 - [Retrieve media asset file with "byIdentifiers" report for external id](#) (see page 835)
 - [Retrieve media asset file with "byItems" report for internal id](#) (see page 836)
 - [Retrieve media asset files which belong \(be assigned\) to a category and are used in PIM CORE](#) (see page 837)
 - [Retrieve all media asset files which belong to a category or its child categories](#) (see page 838)
 - [Update media asset file \(Root Entity\)](#) (see page 839)
 - [Update media asset file language special attribute \(Sub Entity\)](#) (see page 840)
 - [Delete media asset file with identifier D11000100112532](#) (see page 842)
 - [Create media asset file](#) (see page 842)

Adjustments for MediaAssetFile entity

The REST List API access for general object is already described in [REST List API](#) (see page 462), which is also available for the MediaAssetFile entity with some adjustments.

Internal id

Each PIM entity needs an unique internal id which should be a long value and usually comes from the database. For the MediaAssetFile entity it is converted from the media asset identifier (PKOM_PNR) of Media Manager in the way which replaces the first character "D" with the "9".

For example the media asset with identifier D110001123456 has then an internal id 9110001123456.


Special reports

byCategory

Search media asset files which belong (be assigned) to a category

Identifier	Name	Description	Mandatory	Data type	Collection	Entity	Enumeration
categoryId	Category ID	Category id defined in Media Manager	true	STRING	false		
usageFilter	Usage filter	Filter all/used/unused media asset files	false	INTEGER	false		Media asset files usage filter

bySearch

Identifier	Name	Description	Mandatory	Data type	Collection	Entity	Enumeration
query <div> available since 8.0.03 Hotfixes 1</div>	Search query	The query condition for the search. For details of the syntax to be used, refer to the documentation. If this parameter is not specified, then the parameter "mediaQuery" should be set.	false	STRING	false		
mediaQuery	Special media search query	Special search query for media asset file. If this parameter is not specified, then the parameter "query" should be set.	false	STRING	true		

Identifier	Name	Description	Mandatory	Data type	Collection	Entity	Enumeration
usageFilter	Usage filter	Filter all/used/unused media asset files	false	INTEGER	false		Media asset files usage filter

Search with query



This search is available since the 8.0.03 Hotfixes 1

For detailed information about the search with the parameter "query" please visit the page [Search query for MediaAssetFile entity](#) (see page 848).

Search with mediaQuery

For detailed information about the search with the parameter "mediaQuery" please visit the page [Media search query for MediaAssetFile entity](#) (see page 842).

Examples

Retrieve media asset file with "byIdentifiers" report for external id

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -H -X GET http://localhost:1512/rest/V1.0/list/MediaAssetFile/byIdentifiers?identifiers=D11000100112129&fields=MediaAssetFile.Identifier,MediaAssetFileAttribute.Filename,MediaAssetFileAttributeLang.Memo(en),MediaAssetFileAttributeLang.Memo(german)
```

A list model result that contains the desired media asset files and attributes is returned:

```
{
  "cacheId": "20150813_172919_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 1,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 1,
  "columnCount": 0,
```

```

"columns": [],
"rows": [
  {
    "object": {
      "id": "911000100112129",
      "label": "Media asset file 911000100112129"
    },
    "values": [
      "D11000100112129",
      "Lighthouse.jpg",
      "Lighthouse_memo_english",
      "Lighthouse_memo_german"
    ]
  }
]
}

```

Retrieve media asset file with "byItems" report for internal id

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -H -X GET http://localhost:1512/
rest/V1.0/list/MediaAssetFile/byItems?
items=911000100112129,911000100112211&fields=MediaAssetFile.Identifier,MediaAssetFile
Attribute.Filename,MediaAssetFileAttributeLang.Memo(en),MediaAssetFileAttributeLang.M
emo(german)

```

A list model result that contains the desired media asset files and attributes is returned:

```

{
  "cacheId": "20150812_164055_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 2,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 2,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "911000100112129",
        "label": "Media asset file 911000100112129"
      },
      "values": [
        "D11000100112129",
        "Lighthouse.jpg",
        "Lighthouse_memo_english",

```



```

    "Lighthouse_memo_german"
  ],
  {
    "object": {
      "id": "911000100112211",
      "label": "Media asset file 911000100112211"
    },
    "values": [
      "D11000100112211",
      "Koala.jpg",
      "Koala_memo_english",
      "Koala_memo_german"
    ]
  }
]
}

```

Retrieve media asset files which belong (be assigned) to a category and are used in PIM core

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/V1.0/list/MediaAssetFile/byCategory?usageFilter=used&CategoryId=0099000000000000000000000000000000000000000000000000000000000000&fields=MediaAssetFile.Identifier,MediaAssetFileAttribute.Filename
```

The following JSON object is returned:

```
{
  "cacheId": "20150812_170313_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 3,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 3,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "911000100112048",
        "label": "Media asset file 911000100112048"
      },
      "values": [
        "D11000100112048",
        "text.txt"
      ]
    }
  ]
}
```

```

    ],
    },
    ...
  ]
}

```

Retrieve all media asset files which belong to a category or its child categories

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/list/MediaAssetFile/byCategory?
CategoryId=0099%&fields=MediaAssetFile.Identifier,MediaAssetFileAttribute.Filename

```

The following JSON object is returned:

```

{
  "cacheId": "20160204_181155_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 191,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 100,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "911000100111676",
        "label": "Media-Asset-Datei 911000100111676",
        "entityId": 2500
      },
      "values": [
        "D11000100111676",
        "testtest.jpg"
      ]
    }
    ...
  ]
}

```

Update media asset file(Root Entity)

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X POST http://localhost:1512/rest/V1.0/list/MediaAssetFile
```

The following JSON object is provided as content:

```
{
  "columns": [
    {
      "identifier": "MediaAssetFileAttribute.AgencyId"
    },
    {
      "identifier": "MediaAssetFileAttribute.Level"
    },
    ,
    {
      "identifier": "MediaAssetFileAttribute.State"
    },
    ,
    {
      "identifier": "MediaAssetFileAttribute.Status"
    },
    ,
    {
      "identifier": "MediaAssetFileAttribute.Finished"
    },
    ,
    {
      "identifier": "MediaAssetFileAttribute.InProgress"
    },
    ,
    {
      "identifier": "MediaAssetFileAttribute.Class"
    },
    ,
    {
      "identifier": "MediaAssetFileAttribute.Categories"
    }
  ],
  "rows": [
    {
      "object": {
        "id": "'D11000100112038'"
      },
      "values": [
```

[illegible]

The following JSON object is returned:

```
{
  "counters": {
    "errors": 0,
    "warnings": 0,
    "createdObjects": 0,
    "updatedObjects": 1,
    "objectsWithErrors": 0,
    "objectsWithWarnings": 0
  },
  "entries": [],
  "objects": [
    {
      "row": 0,
      "object": {
        "id": "911000100112038",
        "label": "Media asset file 911000100112038"
      },
      "status": [
        "UPDATED"
      ]
    }
  ]
}
```

Update media asset file language special attribute(Sub Entity)

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X POST http://localhost:1512/rest/V1.0/list/MediaAssetFile/MediaAssetFileAttributeLang
```

The following JSON object is provided as content:

```
{
  "columns": [
    {
```

```

    "identifier": "MediaAssetFileAttributeLang.Name"
  }
],
"rows": [
  {
    "object": {
      "id": "911000100112038"
    },
    "qualification": {
      "language": "English"
    },
    "values": [
      "updatedName en"
    ]
  },
  {
    "object": {
      "id": "'D11000100112013'"
    },
    "qualification": {
      "language": "de"
    },
    "values": [
      "updated Name (deu)"
    ]
  }
]
}

```

The following JSON object is returned:

```

{
  "counters": {
    "errors": 0,
    "warnings": 0,
    "createdObjects": 0,
    "updatedObjects": 2,
    "objectsWithErrors": 0,
    "objectsWithWarnings": 0
  },
  "entries": [],
  "objects": [
    {
      "row": 0,
      "object": {
        "id": "911000100112038",
        "label": "Media asset file 911000100112038"
      },
      "status": [
        "UPDATED"
      ]
    }
  ]
}

```

```

    },
    {
      "row": 1,
      "object": {
        "id": "911000100112013",
        "label": "Media asset file 911000100112013"
      },
      "status": [
        "UPDATED"
      ]
    }
  ]
}

```

Delete media asset file with identifier D11000100112532

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X DELETE http://localhost:1512/rest/V1.0/list/MediaAssetFile/byIdentifiers?identifiers=D11000100112532
```

or


Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X DELETE http://localhost:1512/rest/V1.0/list/MediaAssetFile/byItems?Items=911000100112532
```

Create media asset file

Please visit the page [Upload single Media Asset File](#).

Media search query for MediaAssetFile entity

 This search with the parameter "mediaQuery" for MediaAssetFile entity is available since the Version 8.0.00.00.

The report "bySearch" for the entity MediaAssetFile provides searching and finally result listing of the data entity MediaAssetFile. Its specific parameter "mediaQuery" is used to search the result directly with the search fields defined in Media Manager, while the specific parameter "query" is for the usual search with the Field entity defined in PIM repository, just like other general objects in system.

This chapter introduces the report "bySearch" with the parameter "mediaQuery".

Currently it is only supported by PIM Core with Media Manager and the data access is not per PIM database, but per the connector API.

- [Syntax for mediaQuery](#) (see page 843)
- [Examples](#) (see page 847)
 - [Search all media asset files whose filename contain "test"](#) (see page 847)

Syntax for mediaQuery

The value for the parameter **mediaQuery** contains three components: FIELD, OPERATOR, CONTENT.

Syntax
mediaQuery: <FIELD> <OPERATOR> <CONTENT>

The following table indicates all possible value for FIELD, OPERATOR and corresponding CONTENT.

FIELD	OPERATOR	CONTENT	Example search content	Corresponding PIM Desktop search fieldname (en)
*	contain	Character	My search value.	Quick search
F_IMGKOH.IMHL_IHIE_ID	is	Category ID	03110000%	Category
F_IMGKOMP.PKOM_BEZ	is/contain/begin/empty	Character	My search value (If 'empty' operator is used no search value is needed)	Name
FULLTEXT	contains	Character	My search value	(Text search) Document
FILENAME	is/contain/begin/empty	Character	My search value (If 'empty' operator is used no search value is needed)	File name

FIELD	OPERATOR	CONTENT	Example search content	Corresponding PIM Desktop search fieldname (en)
F_ATTRIBUT.ATTR_NAME	is/contain/begin/empty	Character	My custom defined state (If 'empty' operator is used no search value is needed)	State
F_IMGKOMP.PKOM_STATUS	is	-/s/a	s -> Locked a -> Archived - -> Free	Status
F_IMGKOMP.PKOM_TYPE	is	9 values	docus images videos sounds fonts internet profiles common drawings	Medias type
F_IMGKOMP.PKOM_FERTIG	is	True/False	false	Finished flag
F_IMGKOMP.PKOM_INARBEIT	is	True/False	true	In progress
F_IMGKOMP.PKOM_PSIZE	equal/less/greater	Number	10000000	Size in bytes

FIELD	OPERATOR	CONTENT	Example search content	Corresponding PIM Desktop search fieldname (en)
F_IMGKOMP.PKOM_RESOL	equal/less/greater	Number	96	Resolution
F_IMGKOMP.PKOM_COLS	equal/less/greater	Number	24	Color depth
F_IMGMEMO.PIMG_MEMO	contain/begin	Character	My search value (If 'empty' operator is used no search value is needed)	Memo
F_IMGKOMP.PIMG_FARBRAUM	is/contain/begin	Character	rgb	Color space
F_IMGKOMP.PIMG_VERS_NR	equal/less/greater	Number	2	Version number
F_IMGKOMP.PIMG_ANG_AM	is/before/after	Date	20.11.2008 (dd.MM.yyyy)	Media asset created
F_IMGKOMP.PIMG_LASTMOD_TIMESTAMP	is/before/after	Date	20.11.2008 (dd.MM.yyyy)	Media asset last modification date
F_IMGKOMP.PKOM_LASTDATE	is/before/after	Date	20.11.2008 (dd.MM.yyyy)	Last modification date of the source file of the media asset. Last file modification date.
F_IMGKOMP.PKOM_PNR	is/contain/begin	Character	D030001315183	Id

FIELD	OPERATOR	CONTENT	Example search content	Corresponding PIM Desktop search fieldname (en)
F_IMGKOMP.PIMG_PREVIEW_STATE	equal/less	4 values	<p>Special case only this 4 combinations are allowed:</p> <p>No preview: OPERATOR 'less' with '5'</p> <p>No read only preview: OPERATOR 'less' with '9'</p> <p>Error creating preview: OPERATOR 'equal' with '0 or 6'</p> <p>Preview generation queued: OPERATOR 'equal' with CONTENT '3 or 4 or 7 or 8'</p>	Preview version
NUMBER_OF_GROUP_ASSIGNMENT	equal/greater	Number	1	Number of group assignments
F_IMGITEM.IML_ITEM_Mxx (text type property field)	is/contain/begin	Character	My search value	Itemfield type text
F_IMGITEM.IML_ITEM_Mxx (list type property field)	is	List values	My selected list value	Itemfield type list
F_IMGITEM.IML_ITEM_Mxx (bool type property field)	is	True/False	true	Itemfield type bool

FIELD	OPERATOR	CONTENT	Example search content	Corresponding PIM Desktop search fieldname (en)
F_IMGITEM.IMI_ITE Mxx (date type property field)	equal/less/ greater	Date	09.08.2013	Itemfield type date
F_IMGITEM.IMI_ITE Mxx (int type property field)	equal/less/ greater/ lessequal/ greaterequal	Number	4	Itemfield type int
F_IMGITEM.IMI_ITE Mxx (real type property field)	equal/less/ greater/ lessequal/ greaterequal	Number	3.5	Itemfield type real
F_IMGITEM.IMI_ITE Mxx (multiselect list type property field)	contain	Listvalues	My selected list value	Itemfield type multiselect list

Examples

Search all media asset files whose filename contain "test"

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/V1.0/list/MediaAssetFile/bySearch?mediaQuery= FILEName contain test
&fields=MediaAssetFile.Identifier,MediaAssetFileAttribute.FileName
```

The following JSON object is returned:

```
{
  "cacheId": "20150812_171149_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 79,
  "startIndex": 0,
  "pageSize": 100,
```

```

"rowCount": 79,
"columnCount": 0,
"columns": [],
"rows": [
  {
    "object": {
      "id": "911000107072",
      "label": "Media asset file 911000107072"
    },
    "values": [
      "D11000107072",
      "test2.png"
    ]
  },
  ...
]
}

```

Search query for MediaAssetFile entity

i This search with the parameter "query" for MediaAssetFile entity is available since the 8.0.03 Hotfixes 1

The report "bySearch" for the entity MediaAssetFile provides searching and finally result listing of the data entity MediaAssetFile. Its specific parameter "mediaQuery" is used to search the result directly with the search fields defined in Media Manager, while the specific parameter "query" is for the usual search with the Field entity defined in PIM repository, just like other general objects in system.

This chapter introduces the report "bySearch" with the parameter "query".

Currently it is only supported by PIM Core with Media Manager and the data access is not per PIM database, but per the connector API.

- [Syntax for query](#) (see page 848)
- [Examples](#) (see page 853)
 - [Retrieve all media asset files whose filename contain "test"](#) (see page 853)
 - [Retrieve all media asset files which belong to a category or its child categories](#) (see page 854)
 - [Retrieve all media asset files whose version is greater or equals 2](#) (see page 855)
 - [Retrieve all media asset files who are changed on the 2015-11-12 or later](#) (see page 856)
 - [Retrieve all media asset files whose medias type equals "images"](#) (see page 857)
 - [Retrieve all media asset files whose english memo contains "test"](#) (see page 858)
 - [Retrieve all media asset files whose english property field\(of MultiSelectList type\) contains "First"](#) (see page 859)
 - [Retrieve all media asset files whose name start with "test" or whose english memo equals "test"](#) (see page 860)

Syntax for query

The syntax for the search with the parameter "query" is same as the other general objects in PIM, please visit the following page for the corresponding information [REST Search Query Language](#) (see page 447).

While the data access of the search query is not per PIM database, but per the connector API, there are several limitations by search with the parameter "query":

- Search value is always case insensitive, it means that upper and lower case differences will be ignored, so that the result from "equals" is same as the one from "equalsIC", also for "contains" and "containsIC", "startsWith" and "startsWithIC".
- Search with mixed language in one query is not allowed. For example, a search with combination of the different language specific attributes like following is not possible:

ILLEGAL REST CALL

illegal REST call with mixed language

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/V1.0/list/MediaAssetFile/bySearch?query=MediaAssetFileAttributeLang.Memo(English) startsWith "test" or MediaAssetFileAttributeLang.Memo(German) startsWith "test"&fields=MediaAssetFile.Identifier
```

The following table indicates search information for all searchable MediaAssetFile fields defined in repository

ENTITY FIELD	OPERATOR	CONTENT	Example search content	Notice
MediaAssetFile.Identifier	=, != / <>, equals(IC), contains(IC), startsWith(IC), in, is empty	Media asset file identifier (STRING)	D1100010561	
MediaAssetFile.Attribute.Categories	=, != / <>, equals(IC)	<ul style="list-style-type: none"> • exact category id (120-digit-STRING) • id for category and all sub categories (STRING with suffix %) 	<ul style="list-style-type: none"> • "00990010000..." • "0099%" 	

ENTITY FIELD	OPERATOR	CONTENT	Example search content	Notice
MediaAssetFile Attribute.FileName	=, != / <>, equals(IC), contains(IC), startsWith(IC), in, is empty	File name(String)	test.jpg	
MediaAssetFile Attribute.Resolution	=, != / <>, <, <=, >, >=, equals	Resolution value (INTEGER)	100	
MediaAssetFile Attribute.ColorDepth	=, != / <>, <, <=, >, >=, equals	Color depth value (INTEGER)	24	
MediaAssetFile Attribute.ColorSpace	=, != / <>, equals(IC), contains(IC), startsWith(IC), in, is empty	Color space value (STRING)	RGB	
MediaAssetFile Attribute.ColorProfile	=, != / <>, equals(IC), contains(IC), startsWith(IC), in, is empty	Color profile value (STRING)	sRGB	
MediaAssetFile Attribute.Version	=, != / <>, <, <=, >, >=, equals	Version value (INTEGER)	2	
MediaAssetFile Attribute.CreationDate	=, != / <>, <, <=, >, >=, equals	Creation date value (DATETIME)	2011-10-21T 10:00:00	Only the date of the search content is considered for search, time information will be ignored.

ENTITY FIELD	OPERATOR	CONTENT	Example search content	Notice
MediaAssetFile Attribute.LastModified	=, != / <>, <, <=, >, >=, equals	Last modified value (DATETIME)	2011-10-21T 10:00:00	Only the date of the search content is considered for search, time information will be ignored.
MediaAssetFile Attribute.Type	=, != / <>, equals(IC), in	Only the following string values are allowed: <ul style="list-style-type: none"> • docus • images • videos • sounds • fonts • internet • profiles • common • drawings 	images	<p>The allowed search contents are the labels of the Media Manager Enum MediaDataTypes whose id and the detailed medias type name are stored as Class for this entity field in PIM.</p> <p>E.g. a search with the "query=MediaAssetFileAttribute.Type = images" may return media asset files with field value "2#####JPEG" and "2#####Graphics Interchange Format", in which "2" is the id of the MediaDataTypes enum entry, "JPEG" and "Graphics Interchange Format" are detailed medias type names.</p>
MediaAssetFile Attribute.State	=, equals, is empty	entry of Enum.MediaAsset FileState	"Job is finished" 2	
MediaAssetFile Attribute.Status	=, equals	entry of Enum.MediaAsset FileStatus	Free	

ENTITY FIELD	OPERATOR	CONTENT	Example search content	Notice
MediaAssetFile Attribute.Finishe d	=, equals	True False	True	
MediaAssetFile Attribute.InProgr ess	=, equals	True False	False	
MediaAssetFile AttributeLang.N ame	=, != / <>, equals(IC), contains(IC), startsWith(IC), in, is empty	Name value (STRING)	nameTest	
MediaAssetFile AttributeLang.M emo	=, != / <>, equals(IC), contains(IC), startsWith(IC), is empty	Name value (STRING)	memoTest	
Property Field of Boolean type	=, equals	True False	True	
Property Field of Text type	=, != / <>, equals(IC), contains(IC), startsWith(IC), in, is empty	Text value (STRING)	test	
Property Field of Integer type	=, != / <>, <, <=, <, >=, equals, in	Integer value (INTEGER)	2	
Property Field of Decimal type	=, != / <>, <, <=, <, >=, equals	Decimal value (DECIMAL)	11.0	

ENTITY FIELD	OPERATOR	CONTENT	Example search content	Notice
Property Field of Date type	=, != / <>, <, <=, >, >=, equals	Date value (DATE)	2016-05-03	
Property Field of SelectionList type	=, equals	entry of the corresponding enumeration (STRING)	one	
Property Field of MultiSelectionList type	=, equals, contains	entry of the corresponding enumeration (STRING)	First	
Property Field of Long Text type	=, != / <>, equals(IC), contains(IC), startsWith(IC), is empty	Long text value (STRING)	long test test	

Examples

Retrieve all media asset files whose filename contain "test"

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/V1.0/list/MediaAssetFile/bySearch?query=MediaAssetFileAttribute.Filename contains test&fields=MediaAssetFile.Identifier,MediaAssetFileAttribute.Filename
```

The following JSON object is returned:

```
{
  "cacheId": "20160204_110314_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 79,
  "startIndex": 0,
```

```

"pageSize": 100,
"rowCount": 79,
"columnCount": 0,
"columns": [],
"rows": [
  {
    "object": {
      "id": "911000107072",
      "label": "Media asset file 911000107072"
    },
    "values": [
      "D11000107072",
      "test2.png"
    ]
  },
  ...
]
}

```

Retrieve all media asset files which belong to a category or its child categories

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/list/MediaAssetFile/bySearch?query=MediaAssetFileAttribute.Categories =
"0099%"&fields=MediaAssetFile.Identifier,MediaAssetFileAttribute.Filename,
MediaAssetFileAttribute.Categories

```

The following JSON object is returned:

```

{
  "cacheId": "20160204_172920_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 191,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 100,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "911000100111676",
        "label": "Media asset file 911000100111676",
        "entityId": 2500
      },
      "values": [

```

[illegible]

Retrieve all media asset files whose version is greater or equals 2

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/V1.0/list/MediaAssetFile/bySearch?query=MediaAssetFileAttribute.Version >=2&fields=MediaAssetFile.Identifier,MediaAssetFileAttribute.Version
```

The following JSON object is returned:

```
{
  "cacheId": "20160204_173549_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 650,
  "startIndex": 0,
  "pageSize": 100,
}
```

```

"rowCount": 100,
"columnCount": 0,
"columns": [],
"rows": [
  {
    "object": {
      "id": "911000107070",
      "label": "Media asset file 911000107070",
      "entityId": 2500
    },
    "values": [
      "D11000107070",
      "2"
    ]
  },
  {
    "object": {
      "id": "9110001070102",
      "label": "Media asset file 9110001070102",
      "entityId": 2500
    },
    "values": [
      "D110001070102",
      "4"
    ]
  },
  ...
]
}

```

Retrieve all media asset files who are changed on the 2015-11-12 or later

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/list/MediaAssetFile/bySearch?query=MediaAssetFileAttribute.LastModified >=
2015-11-12T10:00:00&fields=MediaAssetFile.Identifier,MediaAssetFileAttribute.LastModi
fied

```

The following JSON object is returned:

```

{
  "cacheId": "20160204_173754_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 181,
  "startIndex": 0,
  "pageSize": 100,

```

```

"rowCount": 100,
"columnCount": 0,
"columns": [],
"rows": [
  {
    "object": {
      "id": "911000100112281",
      "label": "Media asset file 911000100112281",
      "entityId": 2500
    },
    "values": [
      "D11000100112281",
      "2015-11-12T18:59:56:623+0100"
    ]
  },
  {
    "object": {
      "id": "911000100112290",
      "label": "Media asset file 911000100112290",
      "entityId": 2500
    },
    "values": [
      "D11000100112290",
      "2015-11-13T12:44:44:543+0100"
    ]
  },
  ...
]
}

```

Retrieve all media asset files whose medias type equals "images"

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/list/MediaAssetFile/bySearch?query=MediaAssetFileAttribute.Type =
images&fields=MediaAssetFile.Identifier,MediaAssetFileAttribute.Filename,
MediaAssetFileAttribute.Type

```

The following JSON object is returned:

```

{
  "cacheId": "20160204_174021_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 7535,
  "startIndex": 0,
  "pageSize": 100,

```

```

"rowCount": 100,
"columnCount": 0,
"columns": [],
"rows": [
  {
    "object": {
      "id": "91100010561",
      "label": "Media asset file 91100010561",
      "entityId": 2500
    },
    "values": [
      "D1100010561",
      "background.jpg",
      "2####JPEG"
    ]
  },
  {
    "object": {
      "id": "91100010702",
      "label": "Media asset file 91100010702",
      "entityId": 2500
    },
    "values": [
      "D1100010702",
      "joerg_bundesjogi_2.png",
      "2####Portable Network Graphics"
    ]
  },
  ...
]
}

```

Retrieve all media asset files whose english memo contains "test"

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/list/MediaAssetFile/bySearch?query=MediaAssetFileAttributeLang.Memo(English)
contains
"test"&fields=MediaAssetFile.Identifier,MediaAssetFileAttributeLang.Memo(English),
MediaAssetFileAttributeLang.Memo(German)

```

The following JSON object is returned:

```

{
  "cacheId": "20160204_174304_0",
  "entityIdentifier": "MediaAssetFile",

```

```

"totalSize": 5,
"startIndex": 0,
"pageSize": 100,
"rowCount": 5,
"columnCount": 0,
"columns": [],
"rows": [
  {
    "object": {
      "id": "911000107063",
      "label": "Media asset file 911000107063",
      "entityId": 2500
    },
    "values": [
      "D11000107063",
      "test",
      "no"
    ]
  },
  {
    "object": {
      "id": "91100010701194",
      "label": "Media asset file 91100010701194",
      "entityId": 2500
    },
    "values": [
      "D1100010701194",
      "memo_engtest",
      "memo_ger"
    ]
  },
  ...
]
}

```

Retrieve all media asset files whose english property field(of MultiSelectList type) contains "First"

Rest Call

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/list/MediaAssetFile/bySearch?
query=MediaAssetFileAttributeLang.MetadataMultiSelectionList(en) contains
"First"&fields=MediaAssetFile.Identifier,MediaAssetFileAttributeLang.MetadataMultiSel
ectionList(en),MediaAssetFileAttributeLang.MetadataMultiSelectionList(de)

```

The following JSON object is returned:

```

{
  "cacheId": "20160204_174522_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 11,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 11,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "91100010702",
        "label": "Media asset file 91100010702",
        "entityId": 2500
      },
      "values": [
        "D1100010702",
        [
          "Second",
          "First"
        ],
        [
          "Fuenfte",
          "Dritte",
          "Zweite"
        ]
      ]
    },
    {
      "object": {
        "id": "911000107066",
        "label": "Media asset file 911000107066",
        "entityId": 2500
      },
      "values": [
        "D11000107066",
        [
          "First",
          "Third"
        ],
        []
      ]
    },
    ...
  ]
}

```

Retrieve all media asset files whose name start with "test" or whose english memo equals "test"


Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/V1.0/list/MediaAssetFile/bySearch?query=MediaAssetFileAttribute.FileName startsWith "test" or (MediaAssetFileAttribute.Lang.Memo(en) = test )&fields=MediaAssetFile.Identifier,MediaAssetFileAttribute.FileName,MediaAssetFileAttribute.Lang.Memo(en)
```

The following JSON object is returned:

```
{
  "cacheId": "20160204_175436_0",
  "entityIdentifier": "MediaAssetFile",
  "totalSize": 47,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 47,
  "columnCount": 0,
  "columns": [],
  "rows": [
    {
      "object": {
        "id": "911000107063",
        "label": "Media-Asset-Datei 911000107063",
        "entityId": 2500
      },
      "values": [
        "D11000107063",
        "garfield_and_odie.jpg",
        "test"
      ]
    },
    {
      "object": {
        "id": "911000107072",
        "label": "Media-Asset-Datei 911000107072",
        "entityId": 2500
      },
      "values": [
        "D11000107072",
        "test2.png",
        "memo"
      ]
    },
    ...
  ]
}
```

13.6.9.7 REST Media Asset Category API

 The REST Media Asset Category API provides access to media asset categories in the PIM Core.

Currently It is only supported by PIM Core with Media Manager.

- [Get Media Asset Category](#) (see page 862)
- [Get top Media Asset Categories](#) (see page 862)
- [Get direct child Media Asset Categories](#) (see page 863)
- [Add Media Asset Category](#) (see page 863)
- [Update Media Asset Category](#) (see page 864)
- [Delete Media Asset Category](#) (see page 864)
- [Examples](#) (see page 865)
 - [Retrieve single media asset category with identifier](#) (see page 865)
 - [Retrieve all top media asset categories](#) (see page 865)
 - [Retrieve all direct child media asset categories of the category with identifier](#) (see page 866)
 - [Add a top media asset category](#) (see page 867)
 - [Add a child media asset category under the category with identifier](#) (see page 868)
 - [Update the name of the media asset category with identifier](#) (see page 868)
 - [Delete a media asset category with identifier](#) (see page 869)

Get Media Asset Category

URL Pattern	/media/categories/{identifier}
Method	GET
Parameters	-
Media types	application/xml, application/json
Returns	the media asset category as MediaAssetCategory object

Get top Media Asset Categories

URL Pattern	/media/categories/
Method	GET

Parameters	-
Media types	application/xml, application/json
Returns	all top media asset categories as a List of MediaAssetCategory objects

Get direct child Media Asset Categories

URL Pattern	/media/categories/{identifier}/categories
Method	GET
Parameters	-
Media types	application/xml, application/json
Returns	all direct child media asset categories as a List of MediaAssetCategory objects

Add Media Asset Category

URL Pattern	/media/categories
Method	POST
Parameters	parentId
Media types	application/xml, application/json
Returns	created media asset category as a MediaAssetCategory object



The query parameter "parentId" is optional: if it is not set, then a new top category will be created, otherwise a child category will be created under the category with parentId.

Update Media Asset Category

URL Pattern	/media/categories/{identifier}
Method	PUT
Parameters	-
Media types	application/xml, application/json
Returns	updated media asset category as a MediaAssetCategory object

Delete Media Asset Category

URL Pattern	/media/categories/{identifier}
Method	DELETE
Parameters	-
Media types	application/xml, application/json
Returns	string if media asset category is successfully deleted

[illegible]

Rest Client Java Code

```
MediaAssetCategoryRequest mediaAssetCategoryRequest =
restClient.createMediaRequest().createCategoryRequest();
List< MediaAssetCategory > topCategories =
mediaAssetCategoryRequest.getCategories( );
```

Retrieve all direct child media asset categories of the category with identifier

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/  
V1.0/media/categories/  
0001000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000/cATEGORIES
```

Rest Client Java Code

[illegible]

Add a top media asset category

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X POST http://localhost:1501/rest/V1.0/media/categories
```

The following JSON object is provided as content:

```
{
  "name": "newTopCategory"
}
```

The following JSON object is returned:

[illegible]

Rest Client Java Code

```
MediaAssetCategoryRequest mediaAssetCategoryRequest =
restClient.createMediaRequest().createCategoryRequest();
MediaAssetCategoryAttributes categoryAttributes = new
MediaAssetCategoryAttributes( );
```

```
categoryAttributes.setName( "newTopCategory" );
MediaAssetCategory topCategory =
mediaAssetCategoryRequest.addCategory( categoryAttributes );
```

Add a child media asset category under the category with identifier

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X POST http://localhost:1501/rest/
V1.0/media/categories/?
parentId=0001000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
```

Rest Client Java Code

```
MediaAssetCategoryRequest mediaAssetCategoryRequest =
restClient.createMediaRequest().createCategoryRequest();
String parentId =
"0001000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000";
MediaAssetCategoryAttributes categoryAttributes = new
MediaAssetCategoryAttributes( );
categoryAttributes.setName( "newSubCategory" );
MediaAssetCategory subCategory= mediaAssetCategoryRequest.addCategory( parentId,
categoryAttributes );
```

Update the name of the media asset category with identifier

Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X PUT http://localhost:1501/rest/
V1.0/media/categories/
0001000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
```

The following JSON object is provided as content:

```
{
  "name": "newCategoryName"
}
```

The following JSON object is returned:

[illegible]

Rest Client Java Code

[illegible]

Delete a media asset category with identifier

Rest Call

[illegible]

Rest Client Java Code

[illegible]

13.6.10 REST Meta API

The [REST Meta API \(see page 870\)](#) provides meta-information on all available Product Manager objects. It provides access not only to the objects, but also to their field including names, data types, constraints, etc. Using this API you can build powerful generic clients which may work with different installations of the Product Manager with different Repository configurations.

- [All Root Entities \(see page 870\)](#)
 - [Result \(see page 871\)](#)
- [Metadata for a Root Entity \(see page 871\)](#)
 - [Parameters \(see page 872\)](#)
 - [Result \(see page 872\)](#)
- [All Fields of a Root Entity \(see page 873\)](#)
 - [Result \(see page 873\)](#)
- [Metadata for a Sub Entity \(see page 874\)](#)
 - [Parameters \(see page 875\)](#)
 - [Result \(see page 875\)](#)
- [Metadata for a Field \(see page 875\)](#)
 - [Parameters \(see page 876\)](#)
 - [Result \(see page 876\)](#)
- [Metadata for a Qualifier \(see page 878\)](#)
 - [Result \(see page 879\)](#)
- [Examples \(see page 880\)](#)
 - [Retrieving all root entities \(see page 880\)](#)
 - [Retrieving meta data of the root entity Article \(see page 880\)](#)
 - [Retrieving all fields of the root entity Article \(see page 881\)](#)
 - [Retrieving meta data of the sub entity ArticlePriceValuePurchase \(see page 882\)](#)
 - [Retrieving metadata for the field ArticleLang.DescriptionLong \(see page 883\)](#)
 - [Retrieving meta data of the field ArticlePriceVa.DescriptionLong with qualification \(see page 883\)](#)
 - [Retrieving meta data of the qualifier Language for entity ArticleLang \(see page 884\)](#)

13.6.10.1 All Root Entities

Returns all root entities defined in the repository.

URL Pattern	/meta
Method	GET
Parameters	No parameters are available
Media type	text/html, application/json, application/xml

Result	All root entities defined in the repository
--------	---

Result

A list of all root entities supporting the REST API is returned. Each entry contains the following properties:

Properties for entities element			
	Field	Data type	Description
	identifier	String	Unique identifier of the entity
	name	String	Localized name of the entity
	description	String	Localized description of the entity

13.6.10.2 Metadata for a Root Entity

Returns the complete metadata for a root entity including all fields and all sub entities with their qualifiers, fields and sub entities.

URL Pattern	/meta/{root-entity-identifier}
Method	GET
Parameters	visibleOnly
Media type	text/html, application/json, application/xml
Result	The metadata for a root entity, like all direct fields, all direct sub entities etc.

Parameters

Available parameters					
	Parameter	Required	Default	Datatype	Parameter description
	visibleOnly	no	false	boolean	Limits the result of the fields depending on the qualification permissions of the authenticated user. If false, all fields will be returned, even when the user doesn't have the read permission on this field.

Result

The following entity properties are returned:

Properties of the returned object			
	Field	Data type	Description
	identifier	String	Unique identifier
	name	String	Localized name
	qualifiedName	String	Qualified and localized name (in case of a root entity always identical to <i>name</i>)
	description	String	Localized description
	qualifiers	Array	List of qualifiers, always empty in case of a root entity). The available properties for each field are described at qualifier properties (see page 879) .
	fields	Array	List of direct fields. The available properties for each field are described at field properties (see page 876) .

	entities	Array	List of direct sub entities. The available properties are identical to those of the root entity.
--	----------	-------	--

Remarks

- If the media type `text/html` is requested, only the direct fields and sub entities are returned to ensure the HTML page has a manageable size.

13.6.10.3 All Fields of a Root Entity

Returns all fields of a root entity including fields of sub entities. The fields are grouped by category.

URL Pattern	/meta/{root-entity-identifier}/fields
Method	GET
Parameters	No parameters are available
Media type	text/html, application/json, application/xml
Result	All fields (including fields of sub entities) of the specified entity grouped by category. The returned structure contains a list of categories whereby the corresponding fields are included for each category.

Result

The following properties for each field are returned:

Properties of the each <code>categories</code> element			
	Field	Data type	Description
	identifier	String	Unique identifier of the category

	qualifiedName	String	Localized name of category.
	fields	Array	Localized description of the field

Properties of the each `fields` element

	Field	Data type	Description
	resource	URL	Link to the meta data of the field
	identifier	String	Unique identifier of the field
	qualifiedName	String	Qualified and localized name of the field. The default qualifiers are used.
	description	String	Localized description of the field

13.6.10.4 Metadata for a Sub Entity

Returns the metadata for a sub entity. The returned metadata includes a list of qualifiers, a list of direct fields, and a list of direct sub entities.

URL Pattern	/meta/{root-entity-identifier}/{sub-entity-identifier}
Method	GET
Parameters	visibleOnly
Media type	text/html, application/json, application/xml

Result	The metadata for a sub entity. The returned metadata includes information like all direct fields, all sub entities etc. The sub entity does not need to be a direct sub entity of the root entity.
--------	---

Parameters

Available parameters					
	Parameter	Required	Default	Datatype	Parameter description
	visibleOnly	no	false	boolean	Limits the result of the fields depending on the qualification permissions of the authenticated user. If false, all fields will be returned, even when the user doesn't have the read permission on this field.

Result

The returned object has the same properties as the result object for a root entity. See [Properties of a root entity](#) (see page 872).

13.6.10.5 Metadata for a Field

Returns the available metadata for a field.

URL Pattern	/meta/{root-entity-identifier}/{field-identifier}
Method	GET
Parameters	qualification visibleOnly
Media type	text/html, application/json, application/xml

Result	The metadata for a field. The returned metadata includes information like the data type, the enumeration (if available), the cardinality, the field qualifiers etc. If the parameter <i>qualification</i> is set, the qualified name is generated using the specified qualification.
--------	--

Parameters

Available parameters					
	Parameter	Required	Default	Data type	Parameter description
	qualification	no		String	Sets the qualification which is used to generate the content of the property <code>qualifiedName</code> . The format is described in REST Field Qualification (see page 440) whereby only the qualification part has to be specified (including the parentheses). Qualification example for the field <i>ArticlePriceValueSales.Amount</i> : (1,nrp,USD,US,2010-08-06,1.5)
	visibleOnly	no		boolean	Limits the result of the fields depending on the qualification permissions of the authenticated user. If false, all fields will be returned, even when the user doesn't have the read permission on this field.

Result

The following field properties are returned:

Properties			
	Field	Data type	Description
	identifier	String	Unique identifier of this field

	name	String	Localized name of the field
	qualifiedName	String	Qualified and localized name. The default qualifiers are used if not specified explicitly in the <code>qualification</code> parameter.
	category	String	Category this field belongs to
	description	String	Localized description of this field
	dataType	String	Data type of this field. The possible data types are limited and documented here (see page 0)
	enumeration	Object	Proxy object to the enumeration containing all possible values for this field. The proxy contains the properties <code>name</code> and <code>identifier</code> .
	proposalEnum	Object	Proxy object to the enumeration containing a proposal list of possible values (other values are allowed too). The proxy contains the properties <code>name</code> and <code>identifier</code> .
	lowerBound	Integer	Defines the cardinality (lower bound). 0 means the field is optional, 1 means it is mandatory.
	upperBound	Integer	Defines the cardinality (upper bound). 1 means the field is a single value field, -1 means, this field is a list of values.
	minLength	Integer	Defines the minimum length. Only meaningful if the data type is String (see page 0) .
	maxLength	Integer	Defines the maximum length. Only meaningful if the data type is String (see page 0) .

	richtext	Boolean	Defines if this field can hold formatting markers. Only meaningful if the data type is String (see page 0).
	scale	Integer	Defines the number of digits after the decimal separator. Only meaningful if the data type is Decimal (see page 0).
	pictureClause	String	Defines how the value should be formatted. If specified, the default picture clauses which come from the corresponding datatype is overwritten. See Repository documentation (see page 102) for details.
	value	String	Default value or fixed value in case this field is not editable
	visible	Boolean	Defines if this field is visible for the authenticated user or not
	editable	Boolean	Defines if this field can be edited by the authenticated user or not
	multiline	Boolean	Defines if carriage return line feeds are allowed in the string, also adjusts the visualization of the field. Only meaningful if the data type is String (see page 0).
	qualifiers	Array of objects	All qualifiers of this field. The object contains the properties name, description, data type and enumeration. This property is not provided when this field description is embedded in an entity description.

13.6.10.6 Metadata for a Qualifier

Returns the available metadata for a qualifier.

URL Pattern	/meta/{root-entity-identifier}/{sub-entity-identifier}/{qualifier-identifier}
Method	GET

Parameters	No parameters are available
Media type	text/html, application/json, application/xml
Result	The metadata for a qualifier. The returned metadata includes information like the data type, the enumeration (if available), etc

Result

The following qualifier properties are returned:

Properties			
	Field	Data type	Description
	name	String	Localized name.
	qualifiedName	String	Localized and qualified name.
	qualificationFilterIdentifier	String	Identifier used in the property qualification when executing a REST List API Write for Sub Entities (see page 527).
	description	String	Localized description of this field.
	dataType	String	Data type of this qualifier. The possible data types are limited and documented here (see page 0)
	enumeration	Object	Proxy to the enumeration containing all allowed values for this qualifier.
	proposalEnum	Object	Proxy to the enumeration containing a proposal list of possible values (other values are allowed too)

13.6.10.7 Examples

Retrieving all root entities

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/meta
```

The following JSON object is returned:

```

1  {
2      "entities": [
3          {
4              "identifier": "Article",
5              "name": "Item",
6              "description": ""
7          },
8          ...
9      ]
10 }
```

Retrieving meta data of the root entity *Article*

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/meta/Article
```

The following JSON object is returned:

```

1  {
2      "identifier": "Article",
3      "name": "Item",
4      "qualifiedName": "Item",
5      "description": "",
6      "qualifiers": [],
7      "fields": [
8          {
9              "identifier": "Article.Id",
10             "name": "Internal item number",
11             "qualifiedName": "Internal item number",
12             "category": "",
13             "description": "",
14             "dataType": "INTEGER",
15             "lowerBound": "0",
16             "upperBound": "1",
17             "minLength": "0",
18             "maxLength": "0",
```

```

19         "richtext": "false",
20         "rangeMin": "",
21         "rangeMax": "",
22         "scale": "0",
23         "pictureClause": "",
24         "value": "",
25         "editable": "false",
26         "multiline": "false",
27         "searchable": "false"
28     },
29     ...
30 ],
31 "entities": [
32     {
33         "identifier": "ArticleLogistic",
34         "name": "Logistical data",
35         "qualifiedName": "Logistical data",
36         "description": "",
37         "qualifiers": [
38             {
39                 "name": "Country",
40                 "qualifiedName": "Country",
41                 "qualificationFilterIdentifier": "territory",
42                 "description": "",
43                 "dataType": "STRING",
44                 "enumeration": {
45                     "name": "Countries",
46                     "identifier": "Enum.Territory"
47                 },
48                 "value": "DE"
49             }
50         ],
51         "fields": [ ... ]
52     }
53 ]
54 }

```

Retrieving all fields of the root entity *Article*

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/
V1.0/meta/Article/fields
```

The following JSON object is returned:

```

1  {
2      "categories": [
3          {
4              "identifier": "master-data",
5              "qualifiedName": "Header data",
6              "fields": [

```

```

7      {
8          "identifier": "Article.AclProxy",
9          "qualifiedName": "Object rights",
10         "description": "Defines the rights for an object"
11     },
12     {
13         "identifier": "Article.AclFlag",
14         "qualifiedName": "Object right type",
15         "description": "Type of object right"
16     },
17     ....
18 }
19 ]
20 }
```

Retrieving meta data of the sub entity *ArticlePriceValuePurchase*

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/
V1.0/meta/Article/ArticlePriceValuePurchase
```

The following JSON object is returned:

```

1  {
2      "identifier": "ArticlePriceValuePurchase",
3      "name": "Price tier",
4      "qualifiedName": "Price tier",
5      "description": "",
6      "qualifiers": [
7          {
8              "name": "Supplier",
9              "description": "Name of the supplier",
10             "dataType": "ENTITY_ITEM",
11             "enumeration": {
12                 "name": "Suppliers (inc. main supplier)",
13                 "identifier": "Enum.SupplierWithMainSupplier"
14             }
15         },
16         ...
17     ],
18     "fields": [
19         {
20             "identifier": "ArticlePriceValuePurchase.Amount",
21             "name": "Price (from 1.0000)",
22             "description": "Purchase price level"
23         },
24         ...
25     ],
26     "entities": []
27 }
```

Retrieving metadata for the field ArticleLang.DescriptionLong

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/meta/Article/ArticleLang.DescriptionLong
```

The following JSON object is returned:

```

1  {
2      "identifier": "ArticleLang.DescriptionLong",
3      "name": "Long description",
4      "qualifiedName": "Long description (German)",
5      "category": "Texts",
6      "description": "Detailed description of item",
7      "dataType": "STRING",
8      "lowerBound": "0",
9      "upperBound": "1",
10     "minLength": "0",
11     "maxLength": "2147483647",
12     "richtext": "true",
13     "rangeMin": "",
14     "rangeMax": "",
15     "scale": "0",
16     "pictureClause": "",
17     "value": "",
18     "editable": "true",
19     "multiline": "true",
20     "searchable": "true",
21     "qualifiers": [
22         {
23             "name": "Language",
24             "description": "Unique identifier of the language in which the
25 item has been recorded",
26             "dataType": "INTEGER",
27             "enumeration": {
28                 "name": "All languages",
29                 "identifier": "Enum.Language"
30             }
31         }
32     ]
33 }
```

Retrieving meta data of the field ArticlePriceVa.DescriptionLong with qualification

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/meta/Article/ArticlePriceValueSales.Amount?qualification=(1,nrp,USD,US,2012-08-21,1.0)
```

The following JSON object is returned:

```

1  {
2      "identifier": "ArticlePriceValueSales.Amount",
3      "name": "Price",
4      "qualifiedName": "Non-binding price recommendation (from 1)",
5      "category": "Prices",
6      "description": "Selling price level",
7      "dataType": "DECIMAL",
8      "lowerBound": "0",
9      "upperBound": "1",
10     "minLength": "0",
11     "maxLength": "0",
12     "richtext": "false",
13     "rangeMin": "0",
14     "rangeMax": "9999999999.999999",
15     "scale": "2",
16     "pictureClause": "",
17     "value": "",
18     "editable": "true",
19     "multiline": "false",
20     "searchable": "true",
21     "qualifiers": [
22         {
23             "name": "Customer",
24             "description": "Unique number of purchasing company",
25             "dataType": "ENTITY_ITEM",
26             "enumeration": {
27                 "name": "Customers",
28                 "identifier": "Enum.Buyer"
29             }
30         },
31         ...
32     ]
33 }
```

Retrieving meta data of the qualifier *Language* for entity *ArticleLang*

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/meta/Article/ArticleLang/ArticleLangType.LK.Language
```

The following JSON object is returned:

```

1  {
2      "name": "Language",
3      "qualifiedName": "Language",
4      "qualificationFilterIdentifier": "language",
5      "description": "Unique identifier of the language in which the item
has been recorded",
```



```

6      "dataType": "INTEGER",
7      "enumeration": {
8          "name": "All languages",
9          "identifier": "Enum.Language"
10     },
11     "value": "7"
12 }

```

13.6.10.8 REST Meta Media API

e

i The REST Meta Media API provides access to some meta functionalities for media area.

It is only supported when the Informatica Media Manager is used.

- [Derivative definition](#) (see page 886)
 - [Get single derivative definition](#) (see page 886)
 - [Content](#) (see page 886)
 - [Get all derivative definitions](#) (see page 887)
- [Media Asset File Property field definition](#) (see page 887)
 - [Get single property field definition for one supported language](#) (see page 887)
 - [Content](#) (see page 888)
 - [Get single property field definition for all supported languages](#) (see page 891)
 - [Get all property field definitions for all supported languages](#) (see page 892)
- [Examples](#) (see page 892)
 - [Retrieve single derivative definition with identifier](#) (see page 892)
 - [REST call for JSON](#) (see page 892)
 - [REST call for XML](#) (see page 893)
 - [REST Client Java](#) (see page 893)
 - [Retrieve all derivative definitions](#) (see page 894)
 - [REST call for JSON](#) (see page 894)
 - [REST call for XML](#) (see page 894)
 - [REST Client Java](#) (see page 895)
 - [Retrieve single property field definition for one supported language](#) (see page 895)
 - [REST call for JSON](#) (see page 895)
 - [REST call for XML](#) (see page 896)
 - [REST Client Java](#) (see page 897)
 - [Retrieve single property field definition for all supported languages](#) (see page 897)
 - [REST call for JSON](#) (see page 897)
 - [REST call for XML](#) (see page 898)
 - [REST Client Java](#) (see page 899)
 - [Retrieve property field definitions for all supported languages](#) (see page 899)
 - [REST call for JSON](#) (see page 899)
 - [REST call for XML](#) (see page 900)
 - [REST Client Java](#) (see page 901)

Derivative definition

The following API methods provides the read/list access to the derivative definition which are defined in the Informatica Media Manager. The call result can be used to fetch the corresponding derivative of the desired media asset file, for more details please visit the page [REST Media API \(see page 827\)](#).



Derivative definition is a predefined schema with which the master asset (original image) can be converted to a derivative for certain rules (e.g. extension, resolution).

Get single derivative definition

URL Pattern	/meta/media/derivativeDefinitions/{id}
Method	GET
Path Parameters	id: The id of the desired derivaitve definition, must be an integer value
Media types	application/xml, application/json
Result	Derivative definition as a MediaAssetDerivativeDefinition object

Content

The content has to be a JSON/XML object which includes the properties listed below.

MediaAssetDerivativeDefinition

Field	Required	Datatype	Parameter description	Example
id	yes	INTEGER	Unique id of the derivatrive definition	1
shortDescription	no	STRING	Short description of the derivatrive definition	JPEG 36dpi

Field	Required	Datatype	Parameter description	Example
longDescription	no	STRING	Long description of the derivatrive definition	Derivative with JPEG extension and 36 dpi resolution
volumeld	no	INTEGER	Id of the volumn which is defined in the Media Manager to indicate the file storage where the corresponding derivatives are stored	0

Get all derivative definitions

URL Pattern	/meta/media/derivativeDefinitions
Method	GET
Parameters	-
Media types	application/xml, application/json
Result	all derivative definitions as a List of MediaAssetDerivativeDefinition objects

Media Asset File Property field definition

The following API methods provides the read/list access to the media asset file property field definition which are defined in the Informatica Media Manager.



Property field definition is a schema which defines the corresponding settings for a media asset file property field, e.g. number, type, default value.

Get single property field definition for one supported language

URL Pattern	/meta/media/fields/{field-identifier}/{locale}
-------------	--

Method	GET
Path Parameters	<p>field-identifier: field identifier for the property field definition which seems like 'F_IMGITEM.IMI_ITEM3'</p> <p>locale: The locale for which the desired property field definition is defined in the Media Manager</p>
Media types	text/html, application/xml, application/json
Result	The metadata of a field for property field definition. The returned metadata includes information like the identifier, locale, data type etc.

Content

The content has to be a HTML/JSON/XML object which includes the properties listed below.

Field properties for property field definition

Field	Required	Datatype	Parameter description	Example
identifier	yes	STRING	id of the corresponding property field definition	F_IMGITEM.IMI_ITEM3
locale	yes	STRING	locale for which the corresponding property field definition is defined	en_US
name	no	STRING	name of the property file definition	License Free
category	no	STRING	fix value as "General information(Media asset file attributes)"	General information(Media asset file attributes)

Field	Required	Datatype	Parameter description	Example
dataType	no	DataType	data type of the property field definition which is transferred from the original type in the Media Manager	STRING
lowerBound	no	INTEGER	Possible values are 0 or 1, where 0 means the property field is completely optional, and 1 indicates it defines a mandatory property field	0
upperBound	no	INTEGER	The value can be 1 or -1, where -1 means an unlimited amount of objects, and 1 exactly one child object. Currently only the "Multiple selection list" type property field definition has a -1 upperBound	1
maxLength	no	INTEGER	currently only available for the common "Text" type property field definition	3000
rangeMin	no	STRING	only available for "Integer" and "Dezimal number" type property field definition	-1.0
rangeMax	no	STRING	only available for "Integer" and "Dezimal number" type property field definition	9.9
scale	no	INTEGER	The precision of decimal data types. Defines the number of digits after the comma.	2
value	no	STRING	default value	"test"

Field	Required	Datatype	Parameter description	Example
searchable	no	BOOLEAN	True if the search functionalities can be used to search for this property field content, false if not	TRUE
readLevel	no	INTEGER	read access level which defined in the Media Manager	0
readWriteLevel	no	INTEGER	read/write access level which defined in the Media Manager	0
allowedValues	no	List of STRING	currently only available for "Selection list" and "Multiple selection list"	["One", "Two", "Three"]
regularExpression	no	STRING	only available for the common "Text" type property field definition(not for "Long text")	*

Following table indicates the mapping from the original property field definition type to the corresponding Product 360 Data Type inclusive additional notes

Property field definition type in Media Manager	Data Type in Product 360	additional notes
Boolean	BOOLEAN	
Date	DATE	

Property field definiton type in Media Manager	DataType in Product 360	additional notes
Decimal number	DECIMAL	<p>rangeMin: it is calculated from the "minimum value", "places" and "decimal places" which are defined in the Media Manager</p> <p>rangeMax: it is calculated from the "maximum value", "places" and "decimal places" which are defined in the Media Manager</p>
Integer	INTEGER	<p>rangeMin: the "minimum value" defined in the Media Manager</p> <p>rangeMax: the "maximum value" defined in the Media Manager</p>
Multiple selection list	STRING	<p>upperBound: -1</p> <p>allowedValues is defined</p>
Selection list	STRING	allowedValues is defined
Text	STRING	number: [1, 100]
Time	TIME	
Long Text	STRING	nummer: [101, 110]

Get single property field definition for all supported languages

URL Pattern	/meta/media/fields/{field-identifier}
Method	GET

Parameters	field-identifier: field identifier for the property field definition which seems like 'F_IMGITEM.IML_ITEM3'
Media types	text/html, application/xml, application/json
Result	All property field definitions as a List of fields grouped by one category. The returned structure contains one category with all fields

Get all property field definitions for all supported languages

URL Pattern	/meta/media/fields
Method	GET
Parameters	-
Media types	text/html, application/xml, application/json
Result	All property field definitions as a List of fields grouped by one category. The returned structure contains one category with all fields

Examples

Retrieve single derivative definition with identifier

REST call for JSON

Rest Call for JSON

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/V1.0/meta/media/derivativeDefinitions/1
```

A MediaAssetDerivativeDefinition JSON object including id and other attributes of the desired derivative definition is returned:

Returned JSON object

```
{
  "id": 1,
  "shortDescription": "JPEG 36dpi",
  "longDescription": "Derivative with JPEG extension and 36 dpi resolution",
  "volumeId": 0
}
```

REST call for XML**Rest Call for XML**

```
curl -u rest:heiler -H "Accept: application/xml" -X GET http://localhost:1512/rest/
V1.0/meta/media/derivativeDefinitions/1
```

A MediaAssetDerivativeDefinition XML object including id and other attributes of the desired derivative definition is returned:

Returned XML object

```
<mediaAssetDerivativeDefinition>
  <id>1</id>
  <shortDescription>JPEG 36dpi</shortDescription>
  <longDescription>Derivative with JPEG extension and 36 dpi resolution</
longDescription>
  <volumeId>0</volumeId>
</mediaAssetDerivativeDefinition>
```

REST Client Java**Rest Client Java Code**

```
MetaMediaRequest metaMediaRequest =
restClient.createMetaRequest().createMediaRequest();
MediaAssetDerivativeDefinition derivativeDefinition =
metaMediaRequest.getDerivativeDefinition( 1 );
```

Retrieve all derivative definitions

REST call for JSON

Rest Call for JSON

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/V1.0/meta/media/derivativeDefinitions
```

The following JSON objec is returned:

Returned JSON object

```
[
  {
    "id": 1,
    "shortDescription": "JPEG 36dpi",
    "longDescription": "Derivative with JPEG extension and 36 dpi resolution",
    "volumeId": 0
  },
  {
    "id": 2,
    "shortDescription": "BMP 90dpi",
    "longDescription": "Derivative with BMP extension and 90 dpi resolution",
    "volumeId": 0
  },
  {
    "id": 3,
    "shortDescription": "PNG 48dpi",
    "longDescription": "Derivative with PNG extension and 48 dpi resolution",
    "volumeId": 0
  },
  ...
]
```

REST call for XML

Rest Call for XML

```
curl -u rest:heiler -H "Accept: application/xml" -X GET http://localhost:1512/rest/V1.0/meta/media/derivativeDefinitions
```

The following XML objec is returned:

Returned XML object

```

<mediaAssetDerivativeDefinitions>
  <mediaAssetDerivativeDefinition>
    <id>1</id>
    <shortDescription>JPEG 36dpi</shortDescription>
    <longDescription>Derivative with JPEG extension and 36 dpi resolution</
longDescription>
    <volumeId>0</volumeId>
  </mediaAssetDerivativeDefinition>
  <mediaAssetDerivativeDefinition>
    <id>2</id>
    <shortDescription>BMP 90dpi</shortDescription>
    <longDescription>Derivative with BMP extension and 90 dpi resolution</
longDescription>
    <volumeId>0</volumeId>
  </mediaAssetDerivativeDefinition>
  <mediaAssetDerivativeDefinition>
    <id>3</id>
    <shortDescription>PNG 48dpi</shortDescription>
    <longDescription>Derivative with PNG extension and 48 dpi resolution</
longDescription>
    <volumeId>0</volumeId>
  </mediaAssetDerivativeDefinition>
  ...
</mediaAssetDerivativeDefinitions>

```

REST Client Java**Rest Client Java Code**

```

MetaMediaRequest metaMediaRequest =
restClient.createMetaRequest().createMediaRequest();
List< MediaAssetDerivativeDefinition > derivativeDefinitions =
metaMediaRequest.getDerivativeDefinitions();

```

Retrieve single property field definition for one supported language

REST call for JSON

Rest Call for JSON

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/meta/media/fields/F_IMGITEM.IMI_ITEM3/en_US
```

A MediaAssetFilePropertyFieldDefinition JSON object including language id and a list of PropertyFieldSingleDefinition is returned:

Returned JSON object

```
{
  "identifier": "F_IMGITEM.IMI_ITEM3",
  "locale": "en_US",
  "name": "Multiple selelction list english",
  "category": "General information(Media asset file attributes)",
  "dataType": "STRING",
  "lowerBound": "0",
  "upperBound": "-1",
  "maxLength": "0",
  "scale": "0",
  "value": "",
  "searchable": "false",
  "readLevel": "0",
  "readWriteLevel": "0",
  "allowedValues": "[First, Second, Third]"
}
```

REST call for XML**Rest Call for XML**

```
curl -u rest:heiler -H "Accept: application/xml" -X GET http://localhost:1512/rest/
V1.0/meta/media/fields/F_IMGITEM.IMI_ITEM3/en_US
```

A MediaAssetFilePropertyFieldDefinition JSON object including language id and a list of PropertyFieldSingleDefinition is returned:

Returned XML object

```
<?xml version="1.0" encoding="UTF-8"?>
<field>
  <identifier>F_IMGITEM.IMI_ITEM3</identifier>
  <locale>en_US</locale>
```

```

<name>Multiple selection list english</name>
<category>General information(Media asset file attributes)</category>
<dataType>STRING</dataType>
<lowerBound>0</lowerBound>
<upperBound>-1</upperBound>
<maxLength>0</maxLength>
<rangeMin></rangeMin>
<rangeMax></rangeMax>
<scale>0</scale>
<value></value>
<searchable>>false</searchable>
<readLevel>0</readLevel>
<readWriteLevel>0</readWriteLevel>
<allowedValues>[First, Second, Third]</allowedValues>
<regularExpression></regularExpression>
</field>

```

REST Client Java**Rest Client Java Code**

```

MetaMediaRequest metaMediaRequest =
restClient.createMetaRequest().createMediaRequest();
PropertyFieldDefinitionField propertyFieldDefinition =
metaMediaRequest.getPropertyFieldDefinition( "F_IMGITEM.IMI_ITEM3", "en_US" );

```

Retrieve single property field definition for all supported languages

REST call for JSON**Rest Call for JSON**

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/meta/media/fields/F_IMGITEM.IMI_ITEM3

```

A List of MediaAssetFilePropertyFieldDefinition JSON object is returned:

Returned JSON object

```

{
  "categories": [
    {
      "identifier": "media-asset-file-attributes-general",

```

```

    "qualifiedName": "General information(Media asset file attributes)",
    "fields": [
      {
        "identifier": "F_IMGITEM.IMI_ITEM3",
        "qualifiedName": "Multiple selelction list english",
        "description": "Property field definition 'F_IMGITEM.IMI_ITEM3' for the
locale 'en_US'"
      },
      {
        "identifier": "F_IMGITEM.IMI_ITEM3",
        "qualifiedName": "MultiSelectList Deutsch",
        "description": "Property field definition 'F_IMGITEM.IMI_ITEM3' for the
locale 'de_DE'"
      }
    ]
  }
]
}

```

REST call for XML**Rest Call for XML**

```
curl -u rest:heiler -H "Accept: application/xml" -X GET http://localhost:1512/rest/
V1.0/meta/media/fields/F_IMGITEM.IMI_ITEM3
```

The following XML objec is returned:

Returned XML object

```

<?xml version="1.0" encoding="UTF-8"?>
<fieldsByCategory>
  <categories>
    <category>
      <identifier>media-asset-file-attributes-general</identifier>
      <qualifiedName>General information(Media asset file attributes)</
qualifiedName>
      <fields>
        <field>
          <identifier>F_IMGITEM.IMI_ITEM3</identifier>
          <qualifiedName>Multiple selelction list test</qualifiedName>
          <description>Property field definition 'F_IMGITEM.IMI_ITEM3' for
the locale 'en_US'</description>
        </field>
        <field>
          <identifier>F_IMGITEM.IMI_ITEM3</identifier>
          <qualifiedName>MultiSelectList DE</qualifiedName>

```

```

        <description>Property field definition 'F_IMGITEM.IMI_ITEM3' for
the locale 'de_DE'</description>
    </field>
</fields>
</category>
</categories>

```

REST Client Java

Rest Client Java Code

```

MetaMediaRequest metaMediaRequest =
restClient.createMetaRequest().createMediaRequest();
RepositoryCategories categories = metaMediaRequest.getPropertyFieldDefinitions(
"F_IMGITEM.IMI_ITEM3" );
RepositoryCategory generalAttributeCategory = categories.getCategories()
                                                    .get( 0 );

List< RepositoryField > propertyFieldDefinitions =
generalAttributeCategory.getFields();

```

Retrieve property field definitions for all supported languages

REST call for JSON

Rest Call for JSON

```

curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/
V1.0/meta/media/fields

```

A List of MediaAssetFilePropertyFieldDefinition JSON object is returned:

Returned JSON object

```

{
  "categories": [
    {
      "identifier": "media-asset-file-attributes-general",
      "qualifiedName": "General information(Media asset file attributes)",
      "fields": [
        {
          "identifier": "F_IMGITEM.IMI_ITEM1",
          "qualifiedName": "Licensefree",

```

```

        "description": "Property field definition 'F_IMGITEM.IMI_ITEM1' for the
locale 'en_US'"
    },
    ...
    {
        "identifier": "F_IMGITEM.IMI_ITEM1",
        "qualifiedName": "Lizenzfrei",
        "description": "Property field definition 'F_IMGITEM.IMI_ITEM1' for the
locale 'de_DE'"
    },
    ...
]
}
]
}

```

REST call for XML**Rest Call for XML**

```
curl -u rest:heiler -H "Accept: application/xml" -X GET http://localhost:1512/rest/
V1.0/meta/media/fields
```

The following XML object is returned:

Returned XML object

```

<?xml version="1.0" encoding="UTF-8"?>
<fieldsByCategory>
  <categories>
    <category>
      <identifier>media-asset-file-attributes-general</identifier>
      <qualifiedName>General information(Media asset file attributes)</
qualifiedName>
      <fields>
        <field>
          <identifier>F_IMGITEM.IMI_ITEM1</identifier>
          <qualifiedName>Licensefree</qualifiedName>
          <description>Property field definition 'F_IMGITEM.IMI_ITEM1' for
the locale 'en_US'</description>
        </field>
        ...
        <field>
          <identifier>F_IMGITEM.IMI_ITEM1</identifier>
          <qualifiedName>Lizenzfrei</qualifiedName>
          <description>Property field definition 'F_IMGITEM.IMI_ITEM1' for
the locale 'de_DE'</description>

```



```

        </field>
        ...
    </fields>
</category>
</categories>

```

REST Client Java

Rest Client Java Code

```

MetaMediaRequest metaMediaRequest =
restClient.createMetaRequest().createMediaRequest();
RepositoryCategories categories = metaMediaRequest.getAllPropertyFieldDefinitions();
RepositoryCategory generalAttributeCategory = categories.getCategories().get( 0 );
List< RepositoryField > allPropertyFieldDefinitions =
generalAttributeCategory.getFields();

```

13.6.11 REST Security API

The REST Security API provides access to securities of the Product Manager.

- [List all Security ACL APIs](#) (see page 901)
 - [Result](#) (see page 902)
- [Read ACL object](#) (see page 902)
 - [Content](#) (see page 903)
- [Create ACL object](#) (see page 904)
- [Examples](#) (see page 907)
 - [Read ACL object for an id](#) (see page 907)
 - [REST call for JSON](#) (see page 907)
 - [REST call for XML](#) (see page 908)
 - [REST Client Java](#) (see page 909)
 - [Create ACL object](#) (see page 910)
 - [REST call for JSON](#) (see page 910)
 - [REST call for XML](#) (see page 911)
 - [REST Client Java](#) (see page 912)

13.6.11.1 List all Security ACL APIs

Returns the list of all available REST APIs for acl security.

URL Pattern	/security/acl/info
Method	GET

Parameters	-
Media type	text/html, application/json, application/xml
Result	A list of all available REST APIs for acl security

Result

A list of all available REST APIs for acl security.

Description	URL Pattern	Method	Parameters	Result	Content
Read ACL Object	/security/acl/{aclId}	GET	-	The ACL object in JSON or XML format	-
Create ACL Object	/security/acl	POST	-	The ACL object, that was created or already existed, in JSON or XML format.	ACL object in JSON or XML format

13.6.11.2 Read ACL object

Returns the ACL object of the specified acl id.

URL Pattern	/security/acl/{acl-id}
Method	GET
Parameters	-
Media type	text/html, application/json, application/xml
Result	The ACL object

Content

The content as returned ACL object has to be a JSON/XML object which includes the properties listed below.

ACL properties			
	Field	Data type	Description
	id	Long	id of the ACL
	entries	List of AclEntry	all AclEntries of the ACL

Each AclEntry has following properties:

Properties of a member of AclEntry			
	Field	Data type	Description
	principal	PrincipalItemReference	An EntityItemReference to principal (user, user group or party)
	fullPermission	boolean	A boolean value indicates whether the principal has the Full permission object right, default value is false
	deletePermission	boolean	A boolean value indicates whether the principal has the Delete permission object right, default value is false
	writePermission	boolean	A boolean value indicates whether the principal has the Write permission object right, default value is false
	readPermission	boolean	A boolean value indicates whether the principal has the Read permission object right, default value is false
Principal properties			

	Field	Data type	Description
	id	String	External identifier of the principal (user, user group or party)
	entityId	Short	Entity id of the principal (user, user group or party)
	deleted	boolean	A flag indicating whether the principal is deleted on the system, default value is false

13.6.11.3 Create ACL object

Creates a new ACL object if it does not exist for the desired AclEntries, otherwise returns the existing ACL object.

URL Pattern	/security/acl
Method	POST
Parameters	-
Media type	text/html, application/json, application/xml
Result	The created or existing ACL object



- Handling for deleted principal: following table indicates the different responses according to the provided content (which contains the deleted flag for principal) and the fact whether the corresponding principal is actually deleted or not.

Flag "deleted" of principal in content	Principal is deleted on the system	Result
false/true	false (principal existing)	AclEntry with that principal is considered for the created Acl on the system. The returned ACL contains the principal with "false" value for the "deleted" flag.
false	true (principal deleted)	An Exception is returned that the specified principal EntityItemReference does not exist.
true	true (principal deleted)	AclEntry with that principal is NOT considered for the created Acl on the system. The created and returned ACL does not contain the principal. If no existing principal is available an exception (HTTP 400) is thrown.

- The given permission combination of each AclEntry might be adjusted according to the ACL object right rule: all implicit permissions of the given ones will be also granted.

fullPermission	deletePermission	writePermission	readPermission	adjusted permissions of created object
true	true/false	true/false	true/false	"fullPermission": true, "deletePermission": true, "writePermission": true, "readPermission": true

fullPermission	deletePermission	writePermission	readPermission	adjusted permissions of created object
false	true	true/false	true/false	"fullPermission": false, "deletePermission": true, "writePermission": true, "readPermission": true
false	false	true	true/false	"fullPermission": false, "deletePermission": false, "writePermission": true, "readPermission": true
false	false	false	true	"fullPermission": false, "deletePermission": false, "writePermission": false, "readPermission": true
false	false	false	false	"fullPermission": false, "deletePermission": false, "writePermission": false, "readPermission": false

- When a principal of the type Party is used in the AclEntry, only a PrincipalItemReference with the id 'Heiler Product Manager' is allowed.
If other id values are used, this will result in a not found principal in the system.

13.6.11.4 Examples

Read ACL object for an id

REST call for JSON

Rest Call for JSON

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/security/acl/3
```

The following JSON object is returned:

Returned JSON object

```
{
  "id": 3,
  "entries": [
    {
      "principal": {
        "id": "'Heiler Product Manager'",
        "entityId": 2800
        "deleted": false
      },
      "fullPermission": false,
      "deletePermission": false,
      "writePermission": false,
      "readPermission": true
    },
    {
      "principal": {
        "id": "'userGroup1'",
        "entityId": 2700,
        "deleted": false
      },
      "fullPermission": false,
      "deletePermission": true,
      "writePermission": true,
      "readPermission": true
    },
    {
      "principal": {
        "id": "'user1'",
        "entityId": 2600,
        "deleted": false
      },
    },
  ]
}
```

```

    "fullPermission": true,
    "deletePermission": true,
    "writePermission": true,
    "readPermission": true
  },
  {
    "principal": {
      "id": "'deletedUser'",
      "entityId": 2600,
      "deleted": true
    },
    "fullPermission": false,
    "deletePermission": true,
    "writePermission": true,
    "readPermission": true
  }
]
}

```

REST call for XML

Rest Call for XML

```
curl -u rest:heiler -H "Accept: application/xml" -X GET http://localhost:1501/rest/V1.0/security/acl/3
```

The following XML object is returned:

Returned XML object

```

<acl>
  <id>3</id>
  <entries>
    <entry>
      <principal>
        <id>'Heiler Product Manager'</id>
        <entityId>2800</entityId>
        <deleted>false</deleted>
      </principal>
      <fullPermission>false</fullPermission>
      <deletePermission>false</deletePermission>
      <writePermission>false</writePermission>
      <readPermission>true</readPermission>
    </entry>
    <entry>
      <principal>
        <id>'userGroup1'</id>

```



```

        <entityId>2700</entityId>
        <deleted>false</deleted>
    </principal>
    <fullPermission>false</fullPermission>
    <deletePermission>true</deletePermission>
    <writePermission>true</writePermission>
    <readPermission>true</readPermission>
</entry>
<entry>
    <principal>
        <id>'user1'</id>
        <entityId>2600</entityId>
        <deleted>false</deleted>
    </principal>
    <fullPermission>true</fullPermission>
    <deletePermission>true</deletePermission>
    <writePermission>true</writePermission>
    <readPermission>true</readPermission>
</entry>
<entry>
    <principal>
        <id>'deletedUser'</id>
        <entityId>2600</entityId>
        <deleted>true</deleted>
    </principal>
    <fullPermission>false</fullPermission>
    <deletePermission>true</deletePermission>
    <writePermission>true</writePermission>
    <readPermission>true</readPermission>
</entry>
</entries>
</acl>

```

REST Client Java

Rest Client Java Code

```

SecurityRequest securityRequest = getRestClient().createSecurityRequest();
Long actualAclId = 3L;
ACL actualAcl = securityRequest.getAcl( actualAclId );

```

Create ACL object

REST call for JSON

Rest Call for JSON

```
curl -u rest:heiler -H "Accept: application/json" -X POST http://localhost:1501/rest/V1.0/security/acl
```

The following JSON object is provided as content:

JSON object as content

```
{
  "entries": [
    {
      "principal": {
        "id": "'user1'",
        "entityId": 2600
      },
      "fullPermission": true,
      "deletePermission": false,
      "writePermission": true,
      "readPermission": false
    }
  ]
}
```

The returned protocol looks like:

Returned JSON object

```
{
  "id": 4,
  "entries": [
    {
      "principal": {
        "id": "'user1'",
        "entityId": 2600,
        "deleted": false
      },
      "fullPermission": true,
      "deletePermission": true,
      "writePermission": true,
      "readPermission": true
    }
  ]
}
```

```

    }
  ]
}

```

REST call for XML

Rest Call for XML

```
curl -u rest:heiler -H "Accept: application/xml" -X POST http://localhost:1501/rest/V1.0/security/acl
```

The following JSON object is provided as content:

XML object as content

```

<acl>
  <entries>
    <entry>
      <principal>
        <id>'userGroup1'</id>
        <entityId>2700</entityId>
      </principal>
      <fullPermission>false</fullPermission>
      <deletePermission>true</deletePermission>
      <writePermission>false</writePermission>
      <readPermission>false</readPermission>
    </entry>
  </entries>
</acl>

```

The returned protocol looks like:

Returned XML object

```

<acl>
  <id>5</id>
  <entries>
    <entry>
      <principal>
        <id>'userGroup1'</id>
        <entityId>2700</entityId>
        <deleted>false</deleted>
      </principal>
      <fullPermission>false</fullPermission>
      <deletePermission>true</deletePermission>
    </entry>
  </entries>
</acl>

```

```

        <writePermission>true</writePermission>
        <readPermission>true</readPermission>
    </entry>
</entries>
</acl>

```

REST Client Java

Rest Client Java Code

```

SecurityRequest securityRequest = getRestClient().createSecurityRequest();
//prepares ACL parameter for addAcl method
ACL aclToPersist = new ACL();
List< AclEntry > aclEntriesToPersist = new ArrayList< AclEntry >();
aclToPersist.setEntries( aclEntriesToPersist );

AclEntry aclEntryUser = new AclEntry();
PrincipalItemReference user1 = new
    PrincipalItemReference( EntityItemReferenceFactory.createByIdentifier( "User1" ));
user1.setEntityId( ( short ) 2600 );
user1.setDeleted( false );
aclEntryUser.setPrincipal( user1 );
aclEntryUser.setReadPermission( true );
aclEntriesToPersist.add( aclEntryUser );

//calls the method
ACL createdAcl = securityRequest.addAcl( aclToPersist );

//gets result
Long actualAclId = createdAcl.getId();
List< AclEntry > createdAclEntries = createdAcl.getEntries()

```

13.6.12 PUBLIC Server Health Status

13.6.12.1 Overview

In a multi-server environment, an external load balancer needs to determine the health of each server of a cluster. A server's health is one important aspect for selecting best server for incoming requests. For basic server monitoring, the application login /pim/webaccess page can be used, however, this page doesn't take the server startup and shutdown period into account. Especially in the scenario of graceful shutdown, where the servers first finishes all running jobs before terminating, it is important that the health status is reflecting the unavailability of the server for new incoming requests correctly.

Health Status Page

The health status page indicates if clients can connect to the server. You do not need any authentication to access this status page. The health status can be accessed with the URI `/public/V1.0/health`.

Example

Request:

```
curl -i -X GET http://localhost:1512/public/V1.0/health
```

Response (when clients can connect):

```
HTTP/1.1 200 OK
Content-Type: text/plain

Server state: RUNNING
```

Response Code	Description
200	Clients can connect to the sever.
503	The server is starting or shutting down and no client can connect to this server.

Server state will be displayed in the response body.

Info REST API

For a more sophisticated health check, load balancers can also query the REST endpoint `/rest/V1.0/manage/system/info`. In this case, consumers of this API need to be able to provide authentication and parse the HTTP response.

Example

```
curl -X GET http://localhost:1512/rest/V1.0/manage/system/info -H 'Authorization: Basic cmVzdDpoZWlsZXI='
```

```

HTTP/1.1 200 OK
{
  "state": "RUNNING",
  "runLevel": "STD_READY",
  "identifier": "pim-server1",
  "runtimeInfo": {
    "freeMemory": 747502440,
    "allocatedMemory": 1790967808,
    "maximumMemory": 1908932608,
    "availableProcessors": 8
  },
  "connectedNodes": [],
  "userInfo": {
    "loginName": "portal"
  }
}

```

13.7 Message Queue API

With version 10.0 we introduced a new way to integrate Product 360 with external applications. The Message Queue API.

The Message Queue API provides drastic advantages in terms of stability and resiliency and offers new ways of optimizing performance sensitive tasks. Currently this API only supports the Apache Active MQ implementation. Please refer to our installation section and the official installation manuals from Apache Active MQ for details.

13.7.1 Overview

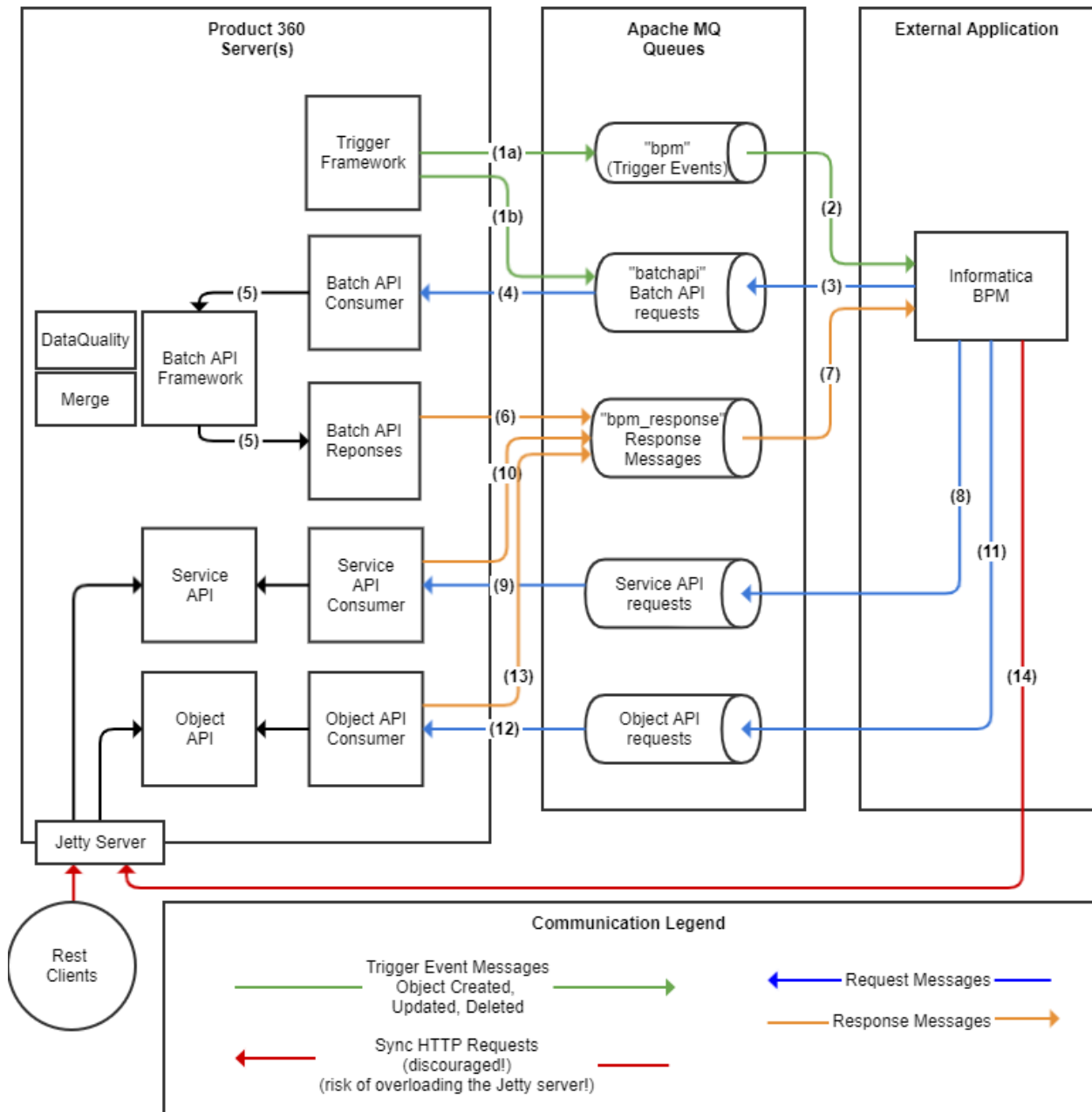
The Message Queue API has been designed with the intention of 3rd party application integration and high volume processing. One of these 3rd parties is our own Business Process Engine. The Message Queue API has been tested with the Informatica BPM system, but is open to any 3rd party application which can read and write to/from an Apache MQ Server.

In order to be most flexible we tried to minimize any efforts in setting up queues and linking applications together.

13.7.2 Queue Communication Overview

Product 360 produces messages as well as receives messages from Apache MQ. This picture shows the general communication streams between an external application and the Product 360 Application Servers by using Message Queues.

Each queue in this picture is an own queue within the Apache MQ Server instance, so a single server will host all those queues.



The queues allow us to decouple the external applications from the product 360 servers and therefore they form also a buffer for peak load scenarios. Each side of the equation can process the messages in their own speed, typically as fast as possible without crashing. If there are more messages within the queue, the queue will safely have them persisted as long as it's needed. In case the Apache Message queue would reach its limit it will automatically make sure that the producer will slow down. All this making sure that no module is overloaded at any time.

Of course, this holds only true in case there are no synchronous REST api calls. None, or just a very minimum. Each rest call is a synchronous call which uses a http thread within the application server. Client users also need those http threads and they are a limited resource. So by executing huge amounts of synchronous requests which might execute some performance intensive logic, it can happen that a server

becomes unresponsive for users. This is something which should be avoided. Using the Message Queue API instead of the synchronous REST api bypasses this.

Let's follow the basic flow:

1. A user modifies something, either manually or with a job.
 - a. A BPM trigger is configured for this modification. A new trigger event message is created with the configured payload. This event is sent to the queue which is configured in the trigger configuration. There can be multiple queues configured for the trigger framework - if more than one, the user can actually chose in which queue the event message should be sent! The setting is within the `server.properties` file, it's called: `infa.bpm.trigger.queue.ids`. It defaults to `" bpm "`.
Also multiple triggers which send events to multiple queues is possible. One for the BPM server, another one for a 3rd party system - even with different payloads.
 - b. A DQ trigger is configured for object created or changed. This trigger will write directly a corresponding DQ message to the batch api queue. As the server which changed the item is not necessarily the same server which processes the Message Queue API. Also this way it's guaranteed that all our batching capabilities are used
2. Informatica BPM (or a 3rd party system) consumes the Trigger Event messages and starts the corresponding workflow process.
3. Based on the workflows logic, a batch api request is sent to the batch api queue. Currently only Data Quality and Merge requests are supported for the [batch api \(see page 926\)](#).
4. The batch api consumer consumes the message from the queue.
5. The batch api framework groups and accumulates multiple messages for a specific amount of time or number of messages in order to optimize the data quality and merge executions. Please refer to [Batch API Queue \(see page 926\)](#) for details on the algorithm and the possible configurations.
6. The results of the data quality and merge executions are split into multiple responses so that every requests receives his own response. This renders the batch processing fully transparent to the external application. The responses are sent to the queue which is defined in the requests as "responseQueue". In this picture the `" bpm_response "` - but it could be a different one!
7. Informatica BPM or a 3rd party application reads the response and processes it by continuing or starting a new workflow process instance.
8. The workflow might also execute other "service api" calls by means of request message. For example enter/leave a workflow status, assigning an item into a task etc. Please have a look at [Service API Queue \(see page 934\)](#) for details on how to compose those messages.
9. The Service API Consumer reads the service api request message and executes it against our standard service api. As this consumer is implemented in a fully generic way he basically is able to execute most of the available service api calls. Not all of them are fully tested and supported. The ones which are are listed on the [Service API Queue \(see page 934\)](#) page
10. The response from the Service api is sent to the response queue, in our example also "bpm_response". This can also be a custom queue!
11. The workflow might also call the object api to obtain a full picture of an object, or to modify the object. In terms of single item processing the object api should be used instead of the list api as the object api is backed by an in memory cache.
12. The object api consumer directly calls the internal object api, bypassing the Service API completely. As the object api is optimized for high throughput the extra marshalling/unmarshalling can be avoided this way. When the object api queue is used, also no further thread pools are involved in the object api - the consumer threads do all the work. In case the object api is executed via rest, an internal thread pool is processing the request. This is to prevent overload scenarios of the application server.
13. The response from the object api is being sent to the response queue, in our example also "bpm_response". This can also be a custom queue!

14. Although discouraged and potentially dangerous, the workflow might also still call the service or object api directly. Please be aware that this will block an http thread during the time of the execution. Depending on the request and the number of concurrent workflows this might lead to an overload in the Jetty Server which will render the server unresponsive for web-clients! As there is a limited number of consumer threads for the Service API Consumer the way via message queue can not overload the system.

13.7.3 General Queue Configuration

All queues which are summarized in "Message Queue API" are configured in the server.properties configuration file.

Each queue has an unique id, the queue id. The ID is name after the queue. prefix. So in this example it's "batchapi".

```
queue.batchapi.type = ${queue.default.type}
queue.batchapi.writer.count = ${queue.default.writer.count}
queue.batchapi.consumer.count = ${queue.default.consumer.count}
queue.batchapi.url = ${queue.default.url}
queue.batchapi.username = ${queue.default.username}
queue.batchapi.password = ${queue.default.password}
queue.batchapi.message.format = XML
queue.batchapi.name = P360_BATCH_API
queue.batchapi.label = BatchAPI
```

- `type` : The type can not be changed as we currently only support ActiveMQ.
- `writer.count` : The number of threads which are used to write messages into the queue
- `consumer.count` : The number of threads which are used to consume messages from the queue
- `url` : the url to the Apache MQ Server
- `username` : the username which should be used for the apache active mq server
- `password` : the password which should be used for the apache active mq server
- `message.format` : Either XML or JSON. Communication with the Informatica BPM Server requires here XML. If other scenarios allow JSON, use it.
- `name` : the name of the Queue within the Apache Active MQ server. If the queue is not available in Apache Active MQ, it will automatically be created
- `label` : the human readable label of the queue. It's used in the trigger configuration UI to chose the the target queue of the trigger event.

13.7.3.1 Queue Purposes

The application allows to configure multiple queues to listen to for service api requests and trigger events. By default the "bpm" queue is used for the trigger events, and the "serviceapi" queue is used for service api requests. Users can configure multiple queues by using a comma seperated list of queue IDs

```
infa.bpm.trigger.queue.ids = bpm
```

```
#infa.bpm.consumer.serviceapi.queue.ids = serviceapi
```

In case there are multiple trigger queues configured, the user can choose into which one the trigger event should be sent when he configures the trigger.

13.7.4 Custom Queues

Customers can easily create own queues by just providing the configuration like described above in the server.properties file. Let's assume we have a downstream syndication system which needs to receive all modifications a user does.

1. Add the "syndication" queue to the server.properties file:

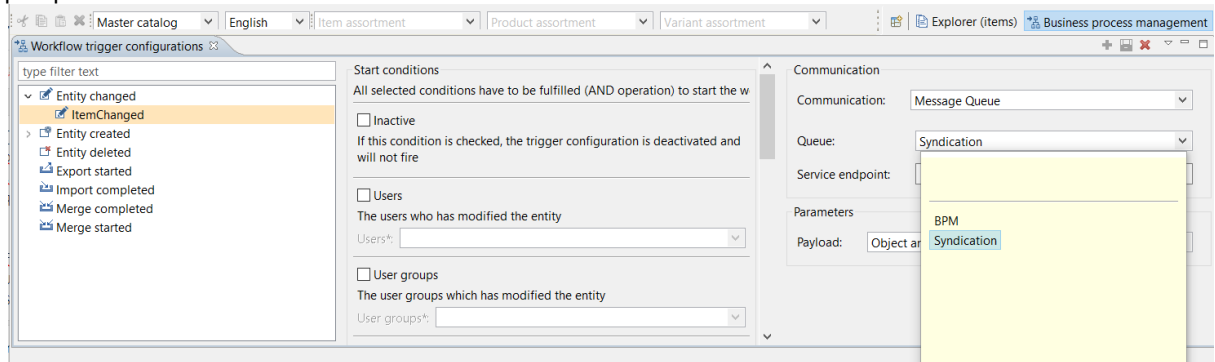
```
queue.syndication.type = ${queue.default.type}
queue.syndication.writer.count = ${queue.default.writer.count}
queue.syndication.consumer.count = ${queue.default.consumer.count}
queue.syndication.url = ${queue.default.url}
queue.syndication.username = ${queue.default.username}
queue.syndication.password = ${queue.default.password}
queue.syndication.message.format = JSON
queue.syndication.name = CUSTOM_SYNDICATION
queue.syndication.label = Syndication
```

We chose JSON as format since it's more efficient than xml and the syndication system can handle it just well.

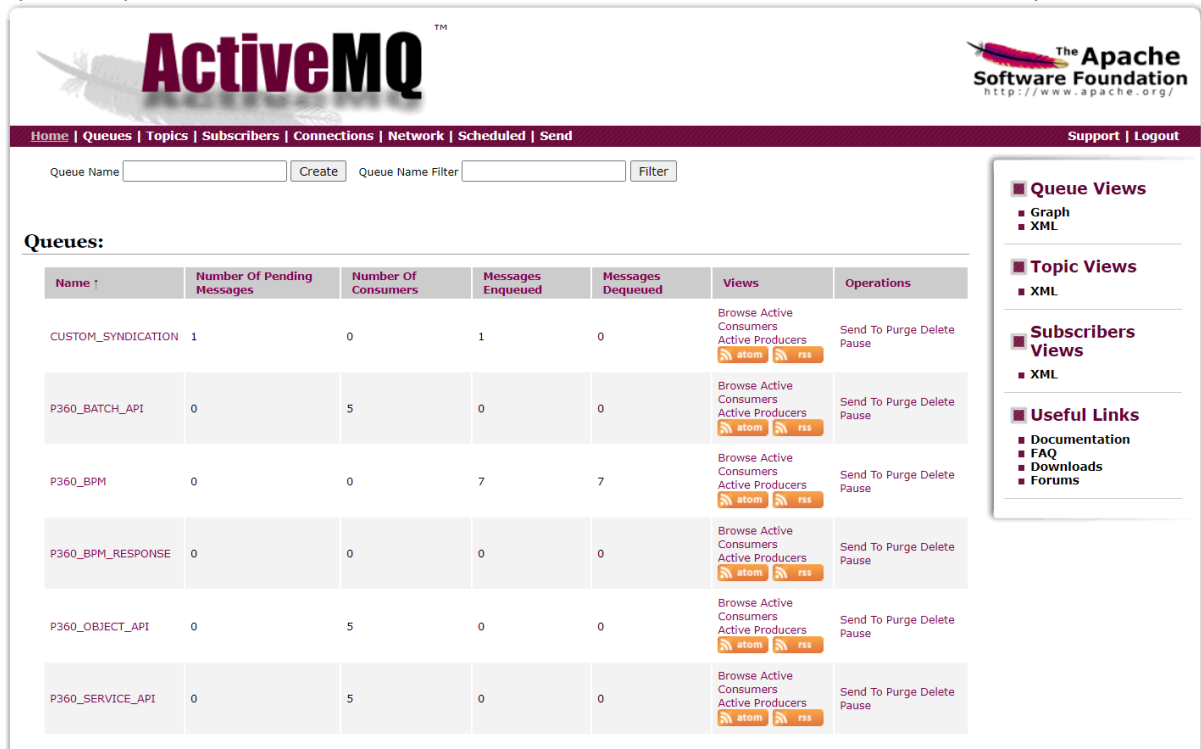
2. Add the "syndication" queue to the trigger queues:

```
infa.bpm.trigger.queue.ids = bpm,syndication
```

3. Choose the "Syndication" queue in the trigger configuration of the "Business Process management" perspective



- Open the Apache Active MQ Console and see the content of the new CUSTOM_SYNDICATION queue



ActiveMQ TM

The Apache Software Foundation
http://www.apache.org/

Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Support | Logout

Queue Name Create Queue Name Filter Filter

Queues:

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
CUSTOM_SYNDICATION	1	0	1	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
P360_BATCH_API	0	5	0	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
P360_BPM	0	0	7	7	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
P360_BPM_RESPONSE	0	0	0	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
P360_OBJECT_API	0	5	0	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
P360_SERVICE_API	0	5	0	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause

Queue Views

- Graph
- XML

Topic Views

- XML

Subscribers Views

- XML

Useful Links

- Documentation
- FAQ
- Downloads
- Forums

- Have a look at the content of the message. It looks something like this

Payload in new Syndication Queue when configured with "Object and change summary" as payload

```

1  {
2      "entityId": "1185@1",
3      "entity": "Article",
4      "data": {
5          "catalog.label": "MASTER",
6          "triggerConfiguration": "ItemChanged",
7          "triggerIdentifier": "hlr.persistance.trigger.entityChanged",
8          "user.label": "Administrator",
9          "userGroup.label": ""
10     },
11     "entityItemChange": {
12         "_module": "UI",
13         "_entity": "Article",
14         "_identifier": "Item_A",
15         "_user": {
16             "_internalId": "1",
17             "_entityId": 2600,
18             "_externalId": "'Administrator'"
19         },
20         "_revision": {
21             "_internalId": "1",

```

```

22         "_entityId": 5600,
23         "_externalId": "'root'"
24     },
25     "_container": {
26         "_internalId": "1",
27         "_entityId": 2900,
28         "_externalId": "'MASTER'"
29     },
30     "_entityItem": {
31         "_internalId": "1185@1",
32         "_entityId": 1000,
33         "_externalId": "'Item_A'@'MASTER'"
34     },
35     "_eventTimestamp": "2022-06-09T10:42:25.420Z",
36     "_changeType": "CHANGED",
37     "_changedFields": [
38         "Article.ManufacturerName"
39     ],
40     "_changedEntities": [
41         "Article"
42     ],
43     "_changeSummary": {
44         "article": {
45             "_changeType": "CHANGED",
46             "_mainSupplierProxy": {
47                 "_entityId": 2800,
48                 "_internalId": "3",
49                 "_externalId": "'Heiler Product Manager'"
50             },
51             "manufacturerName": {
52                 "_old": "Mercedes",
53                 "_current": "BMW"
54             }
55         }
56     }
57 }
58 }

```

13.7.5 General Headers

These headers apply to all queues, unless stated otherwise on the queue page.

13.7.5.1 Request Headers


Headers for request to Product 360

Header name	Type	Mandatory	Stored in	Values	Value example	Purpose
User	general	true	JMS Property	<String>		User to be used for authenticating within Product 360 The user needs to have the needed permissions for the request and of course permission for the Message Queue communication.
Password	general	true	JMS Property	<String>		Password The password of the user. We recommend to use a secure communication to and from Apache Active MQ. The password is in plain text.
MessageFormat	general	true	JMS Property	JSON, XML	XML	Defines the format of the request body. Can either be XML or JSON.
ResponseQueueID	general	false	JMS Property	<String>	bpm_response	Queue id from in server.properties (see above)
JMSCorrelationID	general	false	JMS message header			Identifies the communication series that the message belongs to. The provided correlation id will be sent back in the response so the client link responses to requests.

Header name	Type	Mandatory	Stored in	Values	Value example	Purpose
Origin	general	false	JMS Property	<String>	Test system	Identifies the P360 server environment (DEV/QA/PROD) the message was originates from. This should be taken from a previous trigger message and reflects normally the system.name property in the server.properties.
SuccessTargetService	specific	false	JMS Property	<String>		The name of the target consumer service within the target workflow for success responses.
ErrorTargetService	specific	false	JMS Property	<String>		The target consumer service within the target workflow for error responses

13.7.5.2 Response Headers

These generic response headers are always added to all responses Product 360 consumers send out.

Header name	Type	Stored in	Values	Value example	Purpose
MessageFormat	general	JMS Property	JSON, XML	XML	<p>Defines the content type of the response payload. Can either be XML or JSON.</p> <div>  The message format of the response is defined by the response queue's preferred format (as configured in the server.properties file)! It might be that a client provides a request in JSON, but when the response queue prefers xml, the response will be XML! </div> <p>Informatica BPM (ActiveVOS) only supports XML.</p>
JMSCorrelationID	general	JMS Message header			Identifies the communication series that the message belongs to.
Status	general	JMS Property	Integer (Http like status code)	200	<p>Response status code</p> <p>200 = OK</p> <p>201 = CREATED</p> <p>400 = BAD REQUEST</p> <p>401 = UNAUTHORIZED</p> <p>500 = INTERNAL SERVER ERROR</p> <p>(Additional ones might have been defined by the actual queue consumer implementation)</p>
Origin	general	JMS Property	<String>		Identifies the P360 server environment (DEV/QA/PROD) the message was sent by. Per default this is the value of system.name in the server.properties

Header name	Type	Stored in	Values	Value example	Purpose
P360TargetService	specific	JMS Property			Specifies target endpoint (this is either the <code>SuccessTargetService</code> or the <code>ErrorTargetService</code> from the request header. Depending on the Status of the response)

13.7.6 Default Queues

13.7.6.1 BPM Queue

The main queue to which all trigger events are passed and which should trigger workflow instances initially or by correlation id

Queue ID	Format	Active MQ Name
bpm	XML	P360_BPM

13.7.6.2 Batch API Queue

Some of our Message Queue APIs profit tremendously from our new batching algorithm. The most important ones have been implemented with the Version 10.1 release. **Data Quality and Merge.**

Queue ID	Format	Active MQ Name
batchapi	XML	P360_BATCH_API

13.7.6.3 Service API Queue

General queue providing functionality of the [REST Service API](#) (see page 427).

Queue ID	Format	Active MQ Name
serviceapi	XML	P360_SERVICE_API

13.7.6.4 ObjectAPI Queue

A dedicated message queue for interacting with the Object APIs CRUD operations. The Object API supports both, JSON and XML payloads. We recommend using JSON where possible.

Queue ID	Format	Active MQ Name
objectapi	JSON/XML	P360_OBJECT_API

13.7.7 BPM Queue

The main queue to which all trigger events are passed and which should trigger workflow instances initially or by correlation id

Queue ID	Format	Active MQ Name
bpm	XML	P360_BPM

13.7.7.1 Trigger Message Header

Please note that the General Headers from the [Message Queue API \(see page 914\)](#) page do not apply for the trigger message as this is not a request or a response!

Header name	Stored in	Values	Purpose
JMSCorrelationID	JMS Message header	<String>	Identifies the communication series that the message belongs to. (Not available in trigger events, only in responses where a correlation id was given in the request)

Header name	Stored in	Values	Purpose
MessageFormat	JMS Property	XML/JSON	Similar to HTTP Content-type header. Can either be XML or JSON. Note ActiveVOS can only consume XML.
P360TargetService	JMS Property	<String>	The target consumer service within the target workflow.
Origin	JMS Property	<String>	Identifies the P360 server environment (DEV/QA/PROD) the message was sent by. Per default this is the value of system.name in the server.properties

13.7.7.2 Message Body

The message body consists of the corresponding payload of the trigger which has fired. Details on the payload and how the triggers can be configured can be found in the Trigger Configuration page within the Knowledge Base.

13.7.8 Batch API Queue

Some of our Message Queue APIs profit tremendously from our new batching algorithm. The most important ones have been implemented with the Version 10.1 release. **Data Quality and Merge.**

Queue ID	Format	Active MQ Name
batchapi	XML	P360_BATCH_API

13.7.8.1 Architecture Design

To achieve generic batching, we have done some changes in our Message Queue framework.

- We have introduced a new batch requests queue (P360_BATCH_API) for all batching requests.
- DQ requests queue(P360_DATA_QUALITY) is removed as it is no longer needed. All DQ requests from Trigger Framework or from BPM will be written to P360_BATCH_API queue.
- Batch framework will have its own consumer now for consuming all messages from batch queue(irrespective if it is DQ, Merge or something else in future).

- DQ MQ Consumer is removed as it is not needed.
- Merge requests which need batching should be written on batch requests queue.
- Merge requests on batch queue will be processed synchronously(immediate merge).

Batching Concepts



There is no way around the fact that, taken a single message into consideration, the batching will introduce a small delay in processing of the message. But if many requests are taken into consideration, the overall throughput is way faster!

Incoming requests on batch queue are clubbed together(based on internal key) to form a batch. To decide if a batch is ready for execution, we use these two parameters.

Each service using batch framework can have it's own customized value for those parameters. These can be found in the plugin_customization.ini file.

- **Threshold**
Specifies the size in terms of number of items in a batch after which it is ready for execution. A single request with number of items greater than this threshold value will become a batch with single request. Similarly, a lot of small requests with collective total of items less than threshold will be clubbed into a single batch.
Default: `dataquality.message.batch.threshold=500`
- **Timeout**
Specifies the delay(in millisecond) or time from it's creation after which a batch qualifies for processing(even if it has less number of items than threshold value).
Default: `dataquality.message.batch.timeout=3000`

To change these parameters at runtime, MXBeans are used which allow these values to be updated from JConsole and does not require server restart.

Name	Value
Data Quality batch threshold	500
Data Quality batch timeout	3000



Threshold and Timeout values should not be too small or too big. If values are too small, batch will have very less number of items or will be executed without waiting for several requests to batch. Similarly, if values are too big, it will delay execution of messages as it will try to club too many items in a batch or wait for long time to reach timeout.

Batching can be disabled by setting these properties to 1.

Notes for 10.0 upgrade

New batching framework is backward compatible and old functionality will continue to work in the same way as it was working before.

- Data Quality requests from trigger framework or BPM will be batched as before. Only change will be that these requests will be written on batch queue instead of Data Quality queue (which does not exist after this upgrade).
- Merge requests via Service Api queue or direct REST Api calls from BPM will not be batched and will be processed asynchronously (through Job Framework) as before.
- BPM workflows for merge does not require any change apart from writing the request to new batch queue.
- Workflows for Data Quality call can have extra P360Url parameter (with Data Quality rest endpoint as value) but it is optional. All requests without this parameter will be considered as Data Quality by default.



Merge requests written to Service Api queue or direct REST calls will still be processed as single request(as earlier). Only merge requests via batching queue will be batched.

Request Headers


Please see [Message Queue API \(see page 914\)](#) for the list of generic request headers. This table will only add additional specific ones for this queue!

Header name	Type	Stored in	Values	Value example	Purpose
P360Url	optional	JMS Property	rest/V1.0/ manage/merge or rest/V1.0/ manage/ dataQuality	For merge: rest/V1.0/ manage/merge For data quality (default): rest/V1.0/ manage/ dataQuality	The rest endpoint for service to be invoked (DQ or Merge as of now). It is optional and any message which does not have this parameter will be considered DQ by default.

Response Headers

Please see [Message Queue API \(see page 914\)](#) for the list of generic response headers. No additional headers are provided for this queue.

Protocol in Responses

-  Protocol entries present in the result are only relevant for all the items in the batch, in which items present in the message were executed together. So protocol entries are inconsistent when batching is used. Hence, a single message containing 2 entries might contain protocol relevant for more than 2 entries in it's result, but it will contain the result for those items which have been part of the request!

13.7.8.2 Data Quality

Request Body

DQ request on queue is initiated by either Trigger framework or by a BPM workflow.

DQ Request Message Body

```
<dataQualityProfile>
  <entityIdentifier>Article</entityIdentifier>
  <reportQuery>
    <identifier>byItems</identifier>
    <parameters>
      <entry>
        <key>items</key>
        <value>1@1</value>
      </entry>
    </parameters>
  </reportQuery>
  <rules>IsEmptyManufacturerRule</rules>
</dataQualityProfile>
```

Response Body

DQ Response Message Body

```
{
  "ruleIds": {
    "CheckGtinMED": 12
  },
  "numberOfSuccessfulItems": 5,
  "numberOfFailedItems": 0,
  "items": [
    {
      "entityItem": {
```

```

        "id": "1@1"
    },
    "status": "SUCCESSFUL",
    "failedRuleIds": [],
    "successfulRuleIds": [
        12
    ]
}
],
"protocol": {
    "infoCounter": 3,
    "warningCounter": 0,
    "errorCounter": 0,
    "entries": [
        {
            "severity": "INFO",
            "category": "SUMMARY",
            "message": "1 Regel wird auf 5 Objekte des Typs 'Artikel' angewendet",
            "logDate": "2016-01-04",
            "logTime": "14:52:00"
        },
        {
            "severity": "INFO",
            "category": "SUMMARY",
            "message": "Ausgeführte Regeln: CheckGtinMED",
            "logDate": "2016-01-04",
            "logTime": "14:52:00"
        },
        {
            "severity": "INFO",
            "category": "SUMMARY",
            "message": "Verarbeitung der Regeln beendet.",
            "logDate": "2016-01-04",
            "logTime": "14:52:00"
        }
    ]
}
}
}

```

Preferences

Default value for Data Quality preferences:

Preference	Default Value
dataquality.message.batch.threshold	500

Preference	Default Value
dataquality.message.batch.timeout	3000

13.7.8.3 Merge

Request Body

Field	Required	Default	Description
defaultAction	no	SUPPLEMENT	Default merge mode for all fields in case the corresponding supplier catalog has no persisted merge profile. "SUPPLEMENT", "OVERWRITE", "CLEAN_COPY", "NO_MERGE" This parameter doubles as the default action for the entityQualification parameter object.
entityQualification	no		Optional parameter to fine grain the merge actions down to the field level. Provides also the ability to filter based on qualifications (like: merge only the short description in English). See also special chapter below. More details can be found here : REST Merge API (see page 0)
entityReportQuery	yes		An entity report query which allows to use the available reports 'byItems' only.

Merge Request Body

```
<mergeProfile>
  <defaultAction>SUPPLEMENT</defaultAction>
  <entityQualification>null</entityQualification>
  <entityReportQuery>
    <entityIdentifier>Article</entityIdentifier>
    <identifier>byItems</identifier>
    <parameters>
      <entry>
        <key>items</key>
```

```

        <value>1@10</value>
    </entry>
</parameters>
</entityReportQuery>
</mergeProfile>

```

Response Body

If the batch contains less messages then threshold size, the supplier to master items value pair is returned in the response.

Merge Response Body With Supplier Master Map

```

<mergeResultWithMappings>
  <entity>Article</entity>
  <protocol>
  </protocol>
  <total>
    <supplierMasterMapping supplierId="147@1024" masterId="47764@1"/>
  </total>
  <new>
    <supplierMasterMapping supplierId="147@1024" masterId="47764@1"/>
  </new>
  <newAndUpdated>
    <supplierMasterMapping supplierId="147@1024" masterId="47764@1"/>
  </newAndUpdated>
</mergeResultWithMappings>

```

If the batch contains more messages then threshold size, the report result is returned in the response instead of supplier to master items map.

Merge Response Body With Report Result

```

<mergeResultWithReport>
  <entity>Article</entity>
  <protocol>
  </protocol>
  <total>
    <reportResult>
      <id>10204</id>
      <dataSource>PCM_MASTER</dataSource>
      <type>1</type>
      <purpose>1</purpose>
      <resultTableName>ReportStoreTempA4</resultTableName>
      <count>1</count>
    </reportResult>
  </total>
</mergeResultWithReport>

```



```

</total>
<new>
  <reportResult>
    <id>10204</id>
    <dataSource>PCM_MASTER</dataSource>
    <type>1</type>
    <purpose>1</purpose>
    <resultTableName>ReportStoreTempA4</resultTableName>
    <count>1</count>
  </reportResult>
</new>
<newAndUpdated>
  <reportResult>
    <id>10204</id>
    <dataSource>PCM_MASTER</dataSource>
    <type>1</type>
    <purpose>1</purpose>
    <resultTableName>ReportStoreTempA4</resultTableName>
    <count>1</count>
  </reportResult>
</newAndUpdated>
</mergeResultWithReport>

```

Preferences

Apart from threshold and timeout preferences defined for message queue requests batching, two more preferences are defined for merge.

- **merge.callMergeFinishTrigger** : Merge requests through message queue do not call merge finish trigger by default. This preference parameter can be set to true explicitly if merge finish trigger needs to be called for merge requests through message queue.
- **merge.message.response.threshold** : This preference parameter decides if merge response will have a supplier master mapping or report result. If number of items in a batch are more then this threshold, then merge result will have report result. Otherwise it will have supplier master map.

Default values for merge preferences :

Preference	Default Value
merge.message.batch.threshold	500
merge.message.batch.timeout	3000
merge.message.response.threshold	10000

Preference	Default Value
merge.callMergeFinishTrigger	false

13.7.9 Service API Queue

General queue providing functionality of the [REST Service API](#) (see page 427).

Queue ID	Format	Active MQ Name
serviceapi	XML	P360_SERVICE_API

Fully supported are following endpoints:

- [REST List API Read](#) (see page 470)
- [REST List API Write](#) (see page 514)
- [REST Workflow API](#) (see page 796)
- [REST Task API](#) (see page 783)

13.7.9.1 Message Header

Please see the [Message Queue API](#) (see page 914) for details on generic headers. This table will only list additional ones specific for this queue.

Header name	Stored in	Values	Default	Purpose
P360Url	JMS Property	<String>		The url path addressing a REST endpoint of the Service API. Example: <code>rest/V1.0/list/Article/byCatalog?catalog=MASTER</code>
Method	JMS Property	POST/PUT/GET/DELETE	POST	The REST method for a given endpoint.

Header name	Stored in	Values	Default	Purpose
CallbackTargetService	JMS Property	<String>	same as SuccessTargetService	The target consumer service within the target workflow for scheduled callback responses
EnforceIntermediateOkResponse	JMS Property	true/false	false	Enforces an intermediate ok response message for scheduled endpoints like merge, export, import jobs

13.7.9.2 Error Response

The error responses will be enveloped in an `errorObject` tag with an `errorMessage` text child tag. The response can be plain text or html encoded into xml depending on the error and endpoint.

The response will include a `Status` header property which will represent a status code.

Header name	Value	Description
Status	200/400/401/500	Response status code 200 = OK 400 = BAD REQUEST 401 = UNAUTHORIZED 500 = INTERNAL SERVER ERROR

Example

```
<errorObject>
  <errorMessage><html>
    <head>
      <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"/>
      <title>Error 415 Unsupported Media Type</title>
    </head>
    <body>
      <h2>HTTP ERROR: 415</h2>
      <p>Problem accessing /rest/V1.0/list/Article/byCatalog. Reason:
```

```

<pre>    Unsupported Media Type</pre></p>
<hr /><i><small>Powered by Jetty://</small></i>
</body>
</html>
</errorMessage>
</errorObject>

```

13.7.9.3 Fully Supported APIs

Start Workflow Request

Message Header

Header name	Stored in	Value
P360Url	JMS Property	rest/V2.0/manage/workflow
Method	JMS Property	POST

Message Body

Example 1

```

<workflow>
  <identifier>Workflow1</identifier>
  <label>Workflow1</label>
  <version>1.0</version>
  <status>
    <status>workflowStatus01</status>
    <workflowTask>
      <container>
        <id>1</id>
        <entityId>MASTER</entityId>
      </container>
      <entity>Article</entity>
      <workflowCommunicationMode>QUEUE</workflowCommunicationMode>
      <workflowServiceEndpoint>Workflow01FinishStatus01Service</
workflowServiceEndpoint>
      <workflowQueueId>P360_BPM</workflowQueueId>
      <userGroup>Allmighty</userGroup>
    </workflowTask>
  </status>

```

```

<status>
  <status>workflowStatus02</status>
  <workflowTask>
    <container>
      <id>1</id>
      <entityId>MASTER</entityId>
    </container>
    <entity>Article</entity>
    <workflowCommunicationMode>QUEUE</workflowCommunicationMode>
    <workflowServiceEndpoint>Workflow01FinishStatus02Service</
workflowServiceEndpoint>
    <workflowQueueId>P360_BPM</workflowQueueId>
    <userGroup>Allmighty</userGroup>
  </workflowTask>
</status>
</workflow>

```

Example 2

```

<workflow>
  <identifier>{ $workflowName }</identifier>
  <label>Items with wrong manufacturer</label>
  <version>1.0</version>
  <status>
    <status>DQ_FAILED</status>
    <workflowTask>
      <container>
        <id>' { $catalogName } '</id>
      </container>
      <entity>{ $entityType }</entity>
      <workflowCommunicationMode>QUEUE</workflowCommunicationMode>
      <workflowServiceEndpoint>Scenario1-FinishManufacturerItem</
workflowServiceEndpoint>
      <workflowQueueId>bpm</workflowQueueId>
      <userGroup>InfraUser</userGroup>
    </workflowTask>
  </status>
</workflow>

```

Response Body

Example

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<resultSummary>
  <createdCounter>0</createdCounter>
  <updatedCounter>1</updatedCounter>

```

```
<failedCounter>0</failedCounter>
</resultSummary>
```

End Workflow Request

Message Header

Header name	Stored in	Value
P360Url	JMS Property	rest/V2.0/manage/workflow/ process?processId={processId}"
Method	JMS Property	DELETE

Message Body

- No message body

Response Body

- No response

Process Status Enter

Message Header

Header name	Stored in	Value
P360Url	JMS Property	rest/V2.0/manage/workflow/ status/enter
Method	JMS Property	POST

Message Body

Example

```
<enterWorkflowStatus>
  <processId>1234</processId>
  <workflowId>Workflow1</workflowId>
  <status>workflowStatus01</status>
  <entity>Article</entity>
  <itemId>1@1</itemId>
</enterWorkflowStatus>
```

Response Body

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<statusModificationInfoEnter>
  <successful>true</successful>
  <cannotFindItem>false</cannotFindItem>
  <cannotFindWorkflowId>false</cannotFindWorkflowId>
  <workflowTaskNotFound>false</workflowTaskNotFound>
</statusModificationInfoEnter>
```

Process Status Leave

Message Header

Header name	Stored in	Value
P360Url	JMS Property	rest/V2.0/manage/workflow/status/leave
Method	JMS Property	POST

Message Body

Example

```
<enterWorkflowStatus>
  <processId>1234</processId>
  <workflowId>Workflow1</workflowId>
  <status>workflowStatus01</status>
  <entity>Article</entity>
  <itemId>1@1</itemId>
</enterWorkflowStatus>
```

Response Body

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<statusModificationInfoLeave>
  <successful>true</successful>
  <cannotFindItem>false</cannotFindItem>
  <cannotFindWorkflowId>false</cannotFindWorkflowId>
  <isMissingEnterState>false</isMissingEnterState>
</statusModificationInfoLeave>
```

List Read API

For details on URL parameters and response schema, please see [REST List API Read](#) (see page 470).

Message Header

Header name	Stored in	Value
P360Url	JMS Property	rest/V2.0/list/Article/ byCatalog?catalog=MASTER
Method	JMS Property	GET

Message Body

- No message body

Response Body

Example

```
<entityItemTable>
  <cacheId>no-cache</cacheId>
  <entityIdentifier>Article</entityIdentifier>
  <totalSize>2</totalSize>
  <startIndex>0</startIndex>
  <pageSize>100</pageSize>
  <rowCount>2</rowCount>
  <columnCount>0</columnCount>
  <columns/>
  <rows>
    <row>
      <object>
        <id>1@1</id>
        <entityId>1000</entityId>
      </object>
      <values/>
    </row>
    <row>
      <object>
        <id>2@1</id>
        <entityId>1000</entityId>
      </object>
      <values/>
    </row>
  </rows>
</entityItemTable>
```

List Write API

For details about URL parameters and message body schema please see [REST List API Write](#) (see page 514)

Message Header

Header name	Stored in	Value
P360Url	JMS Property	rest/V2.0/list/Article

Header name	Stored in	Value
Method	JMS Property	POST

Message Body

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<entityItemTable>
  <entityIdentifier>Article</entityIdentifier>
  <columns>
    <column>
      <identifier>Article.CurrentStatus</identifier>
    </column>
  </columns>
  <rows>
    <row>
      <object>
        <id>1@1</id>
        <label>Article_1572531185727002</label>
        <entityId>1000</entityId>
      </object>
      <values>
        <value>01 New</value>
      </values>
    </row>
  </rows>
</entityItemTable>
```

Response Body

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<writeProtocol>
  <counters>
    <errors>0</errors>
    <warnings>0</warnings>
    <createdObjects>0</createdObjects>
    <updatedObjects>1</updatedObjects>
    <objectsWithErrors>0</objectsWithErrors>
    <objectsWithWarnings>0</objectsWithWarnings>
```

```

</counters>
<entries/>
<objects>
  <entry>
    <row>0</row>
    <object>
      <id>1@1</id>
      <label>Article_1572531185727002</label>
      <entityId>1000</entityId>
    </object>
    <status>
      <entry>UPDATED</entry>
    </status>
  </entry>
</objects>
</writeProtocol>

```

Create Task

Message Header

Header name	Stored in	Value
P360Url	JMS Property	rest/V2.0/manage/task
Method	JMS Property	POST

Message Body

Example

```

<taskCreationProfile>
  <task>
    <name>Verify all tools</name>
    <taskType>SingleTask</taskType>
    <entity>Product2G</entity>
    <user>Administrator</user>
    <workflowCommunicationMode>QUEUE</workflowCommunicationMode>
    <workflowQueueId>P360_BPM</workflowQueueId>
    <workflowCorrelationId>123</workflowCorrelationId>
  </task>
  <content>
    <identifier>byItems</identifier>
  </content>
</taskCreationProfile>

```

```

<parameters>
  <entry>
    <key>items</key>
    <value>1@1</value>
  </entry>
</parameters>
</content>
</taskCreationProfile>

```

Response Body

Example

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<entityItemReference>
  <id>10024</id>
</entityItemReference>

```

Scheduling APIs like Merge, Export, Import, DataQuality, Revision

Several Service API endpoints do support asynchronous workflow callbacks.

Endpoints:

- [Merge](#) (see page 755)
(discouraged for single items! Please use the [Batch API Queue](#) (see page 926) for merge operations on single items!)
- [Export](#) (see page 729)
- [Revision](#) (see page 769)
- [Dataquality](#) (see page 704)
(discouraged for single items! Please use the [Batch API Queue](#) (see page 926) for data quality operations on single items!)
- [Import](#) (see page 741)

For those endpoints there is a special response behavior in place. Those endpoints do not respond immediately with a "job planned" response by default, instead the response message is fired when the job is finished and the callback is applied.

If an immediate response message is needed, its possible to specify two headers:

Header name	Stored in	Value	Default
EnforceIntermediate OkResponse	JMS Property	true	false

Header name	Stored in	Value	Default
CallbackTargetService	JMS Property	[workflow endpoint]	same as SuccessTargetService

If those headers are specified, it is possible to control the response endpoints for success, failure and callback responses independently.

Also the query parameters like `workflowServiceEndpoint`, `workflowCorrelationId`, `workflowQueueId` are resolved automatically by the respective JMS message properties and must not be provided in the url.

13.7.10 ObjectAPI Queue

A dedicated message queue for interacting with the Object APIs CRUD operations. The Object API supports both, JSON and XML payloads. We recommend using JSON where possible.

Queue ID	Format	Active MQ Name
objectapi	JSON/XML	P360_OBJECT_API

 The object api consumer only supports Version 2 of the [REST Object API](#) (see page 591)

13.7.10.1 Request & Response Headers

Please see [Message Queue API](#) (see page 914) for details about generic message headers. This table will list only additional headers specific for this queue.

Header name	Type	Stored in	Values	Value example	Purpose
Operation	specific	JMS Property	CREATE READ UPDATE DELETE	CREATE	Defines which method, or operation should or has been performed with the object. This header is also included in the response message

13.7.10.2 Response Headers

Please see [Message Queue API \(see page 914\)](#) for details about generic response headers. There are no additional response headers for the object api responses

13.7.10.3 Parameters

The parameters for the CRUD operations are documented on the [REST Object API Read V2 \(see page 631\)](#) and [REST Object API Write \(see page 673\)](#) pages. Meaning and syntax are identical to the message queue based object api.

The following sections will provide examples of the payloads for JSON as well as XML

13.7.10.4 Create Operation

Request Body

Parameters

Parameter Name	Datatype	Mandatory	Default	Description
<code>_entity</code>	String	true		The identifier of the root entity
<code>_updateIfExists</code>	boolean	false	false	Based on this parameter an already existing entity item will be updated. In order for this to work, the identifier and container (if the entity requires one) must be part of the <code>_data</code> element
<code>_mimeValueArchive</code>	String	false		The unique ID of the uploaded mime archive file as returned from the REST File API (see page 738)

Parameter Name	Datatype	Mandatory	Default	Description
<code>_container</code>	String	false		<p>Optional parameter which will set or override the container as it is defined in the JSON payload. In case of an Item, this would be a ENTITY_ITEM Syntax of a catalog, as the Catalog is the container entity of Item.</p> <p>This parameter is especially useful in testing scenarios in which the same payload needs to be processed repeatedly for different containers.</p>
<code>_data</code>		true		<p>The <code>_data</code> element contains the entity items data. Please see REST Object API Write (see page 673) and Payload Examples (see page 692) for details on the syntax.</p>

Examples

Create Item in TestCatalog, do not update if exists (XML)

```

1  <create>
2    <_entity>Article</_entity>
3    <_container>'TestCatalog'</_container>
4    <_updateIfExists>>false</_updateIfExists>
5    <_data>
6      <identifier>identifier-1</identifier>
7      <gtin>40532130000011</gtin>
8      <manufacturerName>her father's de</manufacturerName>
9      <orderUnit>
10       <_code>RK</_code>
11     </orderUnit>
12     <lang>
13       <_qualification>
14         <language>
15           <_code>deu</_code>
16         </language>
17       </_qualification>
18       <descriptionShort>receive no other answer, and, after a s</
descriptionShort>
19     </lang>

```

```

20     </_data>
21 </create>

```

Create item in TestCatalog, do not update if exists (JSON)

```

1  {
2    "_entity": "Article",
3    "_container": "'TestCatalog'",
4    "_updateIfExists": false,
5    "_data" : {
6      "identifier" : "identifier-1",
7      "gtin" : "40532130000011",
8      "manufacturerName" : "her father's de",
9      "orderUnit" : {
10       "_code" : "RK"
11     },
12     "lang" : [ {
13       "_qualification" : {
14         "language" : {
15           "_code" : "deu"
16         }
17       },
18       "descriptionShort" : "receive no other answer, and, after a s"
19     } ]
20   }
21 }

```

Response Body

The response contains always the status code and if possible all needed information to identify the object which has been created/modified/deleted. This includes it's repository entity identifier, entityItem object, the external identifier, the container. Additionally to that, the response also contains a protocol element which contains all errors or warnings which came up during validation and persistence.

Success Response

Successful Response (XML)

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <_writeResult>
3    <_status>201</_status>
4    <_entity>Article</_entity>
5    <_entityItem>
6      <_internalId>54991@3599</_internalId>
7      <_entityId>1000</_entityId>
8      <_externalId>'identifier-1'@'TestCatalog'</_externalId>
9    </_entityItem>

```



```

10    <_identifier>identifier-1</_identifier>
11    <_container>
12        <_internalId>3599</_internalId>
13        <_entityId>7000</_entityId>
14        <_externalId>'TestCatalog'</_externalId>
15    </_container>
16    <_protocol>
17        <infoCounter>0</infoCounter>
18        <warningCounter>0</warningCounter>
19        <errorCounter>0</errorCounter>
20        <entries/>
21    </_protocol>
22 </_writeResult>

```

Error Response

Error Response in case the item already exists and updateIfExists is set to false

```

1    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2    <_writeResult>
3        <_status>400</_status>
4        <_entity>Article</_entity>
5        <_protocol>
6            <infoCounter>0</infoCounter>
7            <warningCounter>0</warningCounter>
8            <errorCounter>1</errorCounter>
9            <entries>
10                <entry>
11                    <severity>ERROR</severity>
12                    <category>UNIQUENESS</category>
13                    <property></property>
14                    <message>The entity item already exists and can't be
created twice. Provide 'updateIfExists=true' query parameter in case the
object should be updated when already there:
''identifier-1'[Article]@'TestCatalog'[SupplierCatalog]' </message>
15                </entry>
16            </entries>
17        </_protocol>
18    </_writeResult>

```

13.7.10.5 Read Operation

Request Body

Parameters

Parameter	Required	Default	Data type	Parameter description	Example
<code>_entity</code>	yes	none	String	The identifier of the root entity	<code><_entity>Article</_entity></code>
<code>_entityItem</code>	yes, multiple entity items allowed	none	String	A list of entity items in the P360 Service API's ENTITY_ITEM syntax. If more than one item is requested, all items will be returned in a single response object.	<pre> <_entityItem>'identifier -1'@'MASTER'</ entityItem> <_entityItem>'identifier -2'@'MASTER'</ entityItem> </pre>

Parameter	Required	Default	Data type	Parameter description	Example
<code>_entityFilter</code>	no, multiple filters are supported!	none	String	<p>A list of entity identifiers which should be part of the result. If omitted, the full object with all data is returned. We recommend to provide a list of entities which are required in order to gain performance.</p> <p>Note: In case an <code>entityFilter</code> is provided, also the root entity is a filter value. So by only providing the root entity as filter value, only the fields of the root level are returned in the data.</p>	<p>Only return the root fields:</p> <pre><_entityFilter>Article</entityFilter></pre> <p>Json Example:</p> <pre>"_entityFilter": ["Article"]</pre> <p>Only return the root fields, and the sales prices (two <code>_entityFilter</code> elements!)</p> <pre><_entityFilter>Article</_entityFilter> <_entityFilter>ArticleLang</_entityFilter></pre> <p>Json Example:</p> <pre>"_entityFilter": ["Article", "ArticleLang"]</pre> <p>Only return the language specific data like Short and Long Description</p> <pre><_entityFilter>ArticleLang</_entityFilter></pre>

Parameter	Required	Default	Data type	Parameter description	Example
<code>_qualificationFilter</code>	no, multiple filters are supported!	none	String	<p>This parameter allows to restrict the output to certain qualifications. One example would be a filter so that only Euro and US-Dollar prices for the customer Informatica are returned.</p> <p>The filter is a list of qualification settings. A qualification setting is a <i>qualification name</i> followed by a comma-separated list of values which are put into parentheses. The qualification name is defined in the repository and is included in the Meta-API.</p> <p>See also REST List API Read for Sub Entities (see page 490)</p>	<p>Example for prices in Euro and US-Dollar and customer Informatica:</p> <pre><_qualificationFilter>currency(EUR,USD)</_qualificationFilter></pre> <pre><_qualificationFilter>customer(Informatica)</_qualificationFilter></pre> <p>Json Example:</p> <pre>"qualificationFilter": ["currency(EUR,USD)","customer(Informatica)"]</pre>
<code>_revision</code>	no	root	ENTITY_IDENTIFIER	<p>The revision for which the data should be retrieved</p> <p>(Note that "root" is the identifier of the Working Revision, aka Head revision. The internal ID of this revision is 1)</p>	<pre><_revision>'root'</_revision></pre> <pre><_revision>1</_revision></pre> <pre><_revision>'myRevisionIdentifier'</_revision></pre> <p>Json Example:</p> <pre>"_revision": "'root'"</pre>

Parameter	Required	Default	Data type	Parameter description	Example
<code>_includeLabels</code>	no	false	boolean	<p>If set to true, the returned document will contain <code>_label</code> elements for all fields or qualifications which have an enumeration. The label will be returned in the locale of the request like this:</p> <pre> "orderUnit" : { "_current" : { "_key" : { "_entityId" : 3100, "_internalId" : "932", "_externalId" : "'C62'" }, "_code" : "C62", "_label" : "piece" } } </pre>	<pre> <_includeLabels>true</_includeLabels> <_includeLabels>false</_includeLabels> </pre>

Parameter	Required	Default	Data type	Parameter description	Example
<code>_includeIds</code>	no	false	boolean	<p>If set to false, the payload avoids to contain any internal ids which might be system specific. This specifically applies to the ENTITY_ITEM datatype which will then no longer contain the "_internalId" element, only the "_externalId". Also the "_entityId" element will be replaced by an "entity" element which then holds the alphanumeric identifier of the entity and not it's numeric id.</p> <p>While reading and writing to the same Product 360 environment, we always recommend to include the ids since write performance is increased as no internal lookup needs to be done. However, when reading from one system and applying to another P360 system (different database schemas!), includeIds should be set to false, otherwise the write operation might fail since the internal ids do not match to the target system.</p>	<pre><_includeIds>true</_includeIds></pre> <pre><_includeIds>false</_includeIds></pre>

Parameter	Required	Default	Data type	Parameter description	Example
				In case the object is read for a 3rd party system, you might very well disable the internalIds to further reduce the response size!	
<code>_includeMimeTypeArchive</code>	no	false	boolean	<p>If set to true a ZIP Archive will be provided on the server which can be downloaded using the File API (see page 738). This archive will contain all MIME_VALUE binaries of all objects which have been returned by the call to the object API.</p> <p>Is the single item read request is used, only the mime values for this single item are part of the archive. In case the multi-item read request is used the archive will contain the values for all items.</p>	

Examples

Read two article objects with all their data, XML

```

1 <read>
2   <_entity>Article</_entity>
3   <_entityItem>'identifier-1'@'TestCatalog'</_entityItem>
4   <_entityItem>'identifier-2'@'TestCatalog'</_entityItem>
5 </read>

```

Read two article objects, filtered for ArticleLang sub-entity. XML

```

1  <read>
2    <_entity>Article</_entity>
3    <_entityItem>'identifier-1'@'TestCatalog'</_entityItem>
4    <_entityItem>'identifier-2'@'TestCatalog'</_entityItem>
5    <_entityFilter>ArticleLang</_entityFilter>
6  </read>

```

Read two article objects, filtered for ArticleLang sub-entity. JSON

```

1  {
2    "_entity": "Article",
3    "_entityItem": [
4      "'identifier-1'@'TestCatalog'",
5      "'identifier-2'@'TestCatalog'"
6    ],
7    "_entityFilter": [
8      "ArticleLang"
9    ]
10 }

```

Response Body

Examples

Response of 2 items, XML

```

1  <root>
2    <_document>
3      <_entity>Article</_entity>
4      <_entityItem>
5        <_entity>Article</_entity>
6        <_externalId>'identifier-1'@'TestCatalog'</_externalId>
7      </_entityItem>
8      <_revision>
9        <_entity>Revision</_entity>
10       <_externalId>'root'</_externalId>
11     </_revision>
12     <_container>
13       <_entity>SupplierCatalog</_entity>
14       <_externalId>'TestCatalog'</_externalId>
15     </_container>

```



```

16      <_data>
17          <identifier>identifier-1</identifier>
18          <noCUp0rOU>1.0000</noCUp0rOU>
19          <gtin>40532130000011</gtin>
20          <currentStatus>
21              <_code>NEW</_code>
22          </currentStatus>
23          <manufacturerName>BMW</manufacturerName>
24          <catalog>
25              <_code>TestCatalog</_code>
26          </catalog>
27          <quantityMin>1.0000</quantityMin>
28          <mainSupplier>
29              <_code>Heiler Product Manager</_code>
30          </mainSupplier>
31          <priceQuantity>1.0000</priceQuantity>
32          <kitComponent>>false</kitComponent>
33          <quantityInterval>1.0000</quantityInterval>
34          <kitParent>>false</kitParent>
35          <orderUnit>
36              <_code>RK</_code>
37          </orderUnit>
38          <soldOnlyInKits>>false</soldOnlyInKits>
39          <lang>
40              <_qualification>
41                  <language>
42                      <_code>deu</_code>
43                  </language>
44              </_qualification>
45              <descriptionShort>receive no other answer, and, after a
46          s</descriptionShort>
47          </lang>
48          <log>
49              <_qualification>
50                  <channel>
51                      <_code>HPM</_code>
52                  </channel>
53              </_qualification>
54          <modificationDate>2022-06-09T17:43:23.370Z</
55          modificationDate>
56          <creationUser>
57              <_code>rest</_code>
58          </creationUser>
59          <modificationUser>
60              <_code>rest</_code>
61          </modificationUser>
62          <creationDate>2022-06-09T17:15:20.560Z</creationDate>
63          </log>
64          <ownLog>
65          <modificationDate>2022-06-09T17:43:23.383Z</
66          modificationDate>
67          </ownLog>
68      </_data>

```

```

66 </_document>
67 <_document>
68   <_entity>Article</_entity>
69   <_entityItem>
70     <_entity>Article</_entity>
71     <_externalId>'identifier-2'@'TestCatalog'</_externalId>
72   </_entityItem>
73   <_revision>
74     <_entity>Revision</_entity>
75     <_externalId>'root'</_externalId>
76   </_revision>
77   <_container>
78     <_entity>SupplierCatalog</_entity>
79     <_externalId>'TestCatalog'</_externalId>
80   </_container>
81   <_data>
82     <identifier>identifier-2</identifier>
83     <noCUPerOU>1.0000</noCUPerOU>
84     <gtin>40532130000011</gtin>
85     <currentStatus>
86       <_code>NEW</_code>
87     </currentStatus>
88     <manufacturerName>her fatherâ€™s de</manufacturerName>
89     <catalog>
90       <_code>TestCatalog</_code>
91     </catalog>
92     <quantityMin>1.0000</quantityMin>
93     <mainSupplier>
94       <_code>Heiler Product Manager</_code>
95     </mainSupplier>
96     <priceQuantity>1.0000</priceQuantity>
97     <kitComponent>false</kitComponent>
98     <quantityInterval>1.0000</quantityInterval>
99     <kitParent>false</kitParent>
100    <orderUnit>
101      <_code>RK</_code>
102    </orderUnit>
103    <soldOnlyInKits>false</soldOnlyInKits>
104    <lang>
105      <_qualification>
106        <language>
107          <_code>deu</_code>
108        </language>
109      </_qualification>
110      <descriptionShort>receive no other answer, and, after a
111    s</descriptionShort>
112    </lang>
112    <log>
113      <_qualification>
114        <channel>
115          <_code>HPM</_code>
116        </channel>
117      </_qualification>

```

```

118         <creationUser>
119             <_code>rest</_code>
120         </creationUser>
121         <creationDate>2022-06-09T18:03:44.880Z</creationDate>
122     </log>
123     <ownLog>
124         <modificationDate>2022-06-09T18:03:44.910Z</
modificationDate>
125     </ownLog>
126 </_data>
127 </_document>
128 </root>

```

Response of two articles with only their ArticleLang sub-entity data, XML

```

1  <root>
2      <_document>
3          <_entity>Article</_entity>
4          <_entityItem>
5              <_entity>Article</_entity>
6              <_externalId>'identifier-1'@'TestCatalog'</_externalId>
7          </_entityItem>
8          <_revision>
9              <_entity>Revision</_entity>
10             <_externalId>'root'</_externalId>
11          </_revision>
12          <_container>
13              <_entity>SupplierCatalog</_entity>
14              <_externalId>'TestCatalog'</_externalId>
15          </_container>
16          <_data>
17              <lang>
18                  <_qualification>
19                      <language>
20                          <_code>deu</_code>
21                      </language>
22                  </_qualification>
23                  <descriptionShort>receive no other answer, and, after a
s</descriptionShort>
24              </lang>
25          </_data>
26      </_document>
27      <_document>
28          <_entity>Article</_entity>
29          <_entityItem>
30              <_entity>Article</_entity>
31              <_externalId>'identifier-2'@'TestCatalog'</_externalId>
32          </_entityItem>
33          <_revision>
34              <_entity>Revision</_entity>
35              <_externalId>'root'</_externalId>

```

```

36         </_revision>
37     <_container>
38         <_entity>SupplierCatalog</_entity>
39         <_externalId>'TestCatalog'</_externalId>
40     </_container>
41     <_data>
42         <lang>
43             <_qualification>
44                 <language>
45                     <_code>deu</_code>
46                 </language>
47             </_qualification>
48             <descriptionShort>receive no other answer, and, after a
49 s</descriptionShort>
49         </lang>
50     </_data>
51 </_document>
52 </root>

```

Read two articles, filtered for only the ArticleLang sub-entity, JSON

```

{
  "_document": [
    {
      "_entity": "Article",
      "_entityItem": {
        "_entity": "Article",
        "_externalId": "'identifier-1'@'TestCatalog'"
      },
      "_revision": {
        "_entity": "Revision",
        "_externalId": "'root'"
      },
      "_container": {
        "_entity": "SupplierCatalog",
        "_externalId": "'TestCatalog'"
      },
      "_data": {
        "lang": [
          {
            "_qualification": {
              "language": {
                "_code": "deu"
              }
            },
            "descriptionShort": "receive no other answer, and, after a s"
          }
        ]
      }
    }
  ],
}

```

```

{
  "_entity": "Article",
  "_entityItem": {
    "_entity": "Article",
    "_externalId": "'identifier-2'@'TestCatalog'"
  },
  "_revision": {
    "_entity": "Revision",
    "_externalId": "'root'"
  },
  "_container": {
    "_entity": "SupplierCatalog",
    "_externalId": "'TestCatalog'"
  },
  "_data": {
    "lang": [
      {
        "_qualification": {
          "language": {
            "_code": "deu"
          }
        },
        "descriptionShort": "receive no other answer, and, after a s"
      }
    ]
  }
}
]
}

```

13.7.10.6 Update Operation

Request Body

Parameters

Name	Datatype	Default	Description
<code>_entity</code>	String		The entity which should be updated
<code>_entityItem</code>	ENTITY_ITEM		Entity Item to be updated (see above)

Name	Datatype	Default	Description
<code>_createIfMissing</code>	boolean	true	Based on this parameter a missing entity item will be created with the given identifier. In order for this to work, the <code>externalId</code> must be used for the <code>entityItem</code> parameter. In case the <code>internalId</code> is used and the object is not in the system a 404 code will be returned.
<code>_mimeValueArchive</code>	String		The unique ID of the uploaded mime archive file as returned from the REST File API (see page 738)

Examples

Simple request body to update the manufacturer name of an item

Update Request, JSON

```

1  {
2    "_entity": "Article",
3    "_entityItem": "'identifier-1'@'TestCatalog'",
4    "_createIfMissing": false,
5    "_data" : {
6      "manufacturerName" : "BMW"
7    }
8  }
```

Update request, XML

```

1  <update>
2    <_entity>Article</_entity>
3    <_entityItem>'identifier-1'@'TestCatalog'</_entityItem>
4    <_createIfMissing>>false</_createIfMissing>
5    <_data>
6      <manufacturerName>BMW</manufacturerName>
7    </_data>
8  </update>
```

Response Body

Examples

Success Response

Success Response, JSON

```

1  {
2      "_status": 200,
3      "_entity": "Article",
4      "_entityItem": {
5          "_internalId": "54992@3599",
6          "_entityId": 1000,
7          "_entity": null,
8          "_externalId": "'identifier-1'@'TestCatalog'"
9      },
10     "_identifier": "identifier-1",
11     "_container": {
12         "_internalId": "3599",
13         "_entityId": 7000,
14         "_entity": null,
15         "_externalId": "'TestCatalog'"
16     },
17     "_protocol": {
18         "infoCounter": 0,
19         "warningCounter": 0,
20         "errorCounter": 0,
21         "entry": [
22             ]
23     }
24 }

```

Success Response, XML

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <_writeResult>
3      <_status>200</_status>
4      <_entity>Article</_entity>
5      <_entityItem>
6          <_internalId>54992@3599</_internalId>
7          <_entityId>1000</_entityId>
8          <_externalId>'identifier-1'@'TestCatalog'</_externalId>
9      </_entityItem>
10     <_identifier>identifier-1</_identifier>
11     <_container>

```

```

12     <_internalId>3599</_internalId>
13     <_entityId>7000</_entityId>
14     <_externalId>'TestCatalog'</_externalId>
15 </_container>
16 <_protocol>
17     <infoCounter>0</infoCounter>
18     <warningCounter>0</warningCounter>
19     <errorCounter>0</errorCounter>
20     <entries/>
21 </_protocol>
22 </_writeResult>

```

Error Response

Error Response, XML

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <_writeResult>
3      <_status>400</_status>
4      <_entity>Article</_entity>
5      <_entityItem>
6          <_internalId>54992@3599</_internalId>
7          <_entityId>1000</_entityId>
8          <_externalId>'identifier-1'@'TestCatalog'</_externalId>
9      </_entityItem>
10     <_identifier>identifier-1</_identifier>
11     <_container>
12         <_internalId>3599</_internalId>
13         <_entityId>7000</_entityId>
14         <_externalId>'TestCatalog'</_externalId>
15     </_container>
16     <_protocol>
17         <infoCounter>0</infoCounter>
18         <warningCounter>0</warningCounter>
19         <errorCounter>1</errorCounter>
20         <entries>
21             <entry>
22                 <severity>ERROR</severity>
23                 <category>RANGE</category>
24                 <property>Article_ArticleType.ManufacturerName</property>
25                 <message>Manufacturer: The text 'This is a way too long
26                 manufacturer Name as onl...' exceeds the maximum length; it can be a
27                 maximum of 50 characters long. (Currently: 90 characters)</message>
28             </entry>
29         </entries>
30     </_protocol>
31 </_writeResult>

```


13.7.10.7 Delete Request

A request which can delete a single entity item. Equivalent to [REST List API Delete](#) (see page 539)

Request Body

Parameters

Name	Datatype	Default	Description
<code>_entity</code>	String		The entity of the object which should be deleted
<code>_entityItem</code>	ENTITY_ITEM		Entity Item to be deleted

Delete an object, JSON

```

1  {
2      "_entity": "Article",
3      "_entityItem": "'identifier-2'@'TestCatalog'"
4  }
```

Response Body

Example

Successful deletion response, JSON

```

1  {
2      "_status": 200,
3      "_entity": "Article",
4      "_entityItem": {
5          "_internalId": "54995@3599",
6          "_entityId": 1000,
7          "_externalId": "'identifier-2'@'TestCatalog'"
8      },
9      "_identifier": "identifier-2",
10     "_container": {
```

```

11         "_internalId": "3599",
12         "_entityId": 7000,
13         "_externalId": "'TestCatalog'"
14     }
15 }

```

Successful deletion, XML

```

1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <root>
3      <_status>200</_status>
4      <_entity>Article</_entity>
5      <_entityItem>
6          <_internalId>55557@3599</_internalId>
7          <_entityId>1000</_entityId>
8          <_externalId>'identifier-2'@'TestCatalog'</_externalId>
9      </_entityItem>
10     <_identifier>identifier-2</_identifier>
11     <_container>
12         <_internalId>3599</_internalId>
13         <_entityId>7000</_entityId>
14         <_externalId>'TestCatalog'</_externalId>
15     </_container>
16 </root>

```

13.8 How to transfer entity including ACL objects to another PIM system

Hint

Setting the ACL flag via REST to an item does NOT execute the underlying logic like inheritance of ACL objects to sub objects.

An other limitation is at the moment: The set ACL id has to exist at the system.

- [Use case](#) (see page 967)
- [Flow](#) (see page 967)
- [Detail REST calls](#) (see page 967)
 - [Find items with ACL reference](#) (see page 968)
 - [Find ACL objects by id](#) (see page 969)
 - [Create ACL object in target system](#) (see page 971)
 - [Create items in target system with references to created ACL objects](#) (see page 974)
- [More example for assignment of entity with new ACL reference ids in target system](#) (see page 976)
 - [Assortment](#) (see page 976)
 - [Export template](#) (see page 978)
 - [Get id-name map for all export templates](#) (see page 978)
 - [Assign ACL to desired export template](#) (see page 980)

13.8.1 Use case

This page will describe how to transfer objects with assigned object rights from one P360 system (source) to an other P360 system (target) using the Service API.

Item objects are used as example.

13.8.2 Flow

1. Source system: Read the affected items via REST, if available the item brings its own ACL reference id
2. Source system: Read the necessary ACL objects with the previous found reference ids
3. Target system: Write the ACL objects
4. Target system: Write items with new ACL reference ids

13.8.3 Detail REST calls

Used objects
Item no (source system)
<u>Item no (target system)</u>
ACL id (source system)
<u>ACL id (target system)</u>
Principal id (source and target system)
ACL flag (source and target system)
Catalog name (source system)
<u>Catalog name (target system)</u>

13.8.3.1 Find items with ACL reference

First we have to find the ACL ids used by each item:

Request - GET

GET - [http://host:1512/rest/V1.0/list/Article/byCatalog?
catalog=CatalogToTransfer&fields=Article.SupplierAID,Article.AclFlag,Article.AclProxy&metaData=true](http://host:1512/rest/V1.0/list/Article/byCatalog?catalog=CatalogToTransfer&fields=Article.SupplierAID,Article.AclFlag,Article.AclProxy&metaData=true)

Response

```
{
  "cacheId": "20160802_104907_0",
  "entityIdentifier": "Article",
  "totalSize": 3,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 3,
  "columnCount": 3,
  "columns": [
    {
      "identifier": "Article.SupplierAID",
      "dataType": "STRING",
      "name": "Item no."
    },
    {
      "identifier": "Article.AclFlag",
      "dataType": "STRING",
      "name": "Object right type"
    },
    {
      "identifier": "Article.AclProxy",
      "dataType": "ENTITY_ITEM",
      "name": "Object rights"
    }
  ],
  "rows": [
    {
      "object": {
        "id": "2120@1002",
        "label": "itemToTransfer001",
        "entityId": 1000
      },
      "values": [
        "itemToTransfer001",
```

```

    "Individual",
    {
      "id": "9",
      "label": "Object rights 9",
      "entityId": 4400
    }
  ],
},
{
  "object": {
    "id": "2121@1002",
    "label": "itemToTransfer002",
    "entityId": 1000
  },
  "values": [
    "itemToTransfer002",
    "Individual",
    {
      "id": "10",
      "label": "Object rights 10",
      "entityId": 4400
    }
  ]
},
{
  "object": {
    "id": "2122@1002",
    "label": "itemToTransfer003",
    "entityId": 1000
  },
  "values": [
    "itemToTransfer003",
    "Individual",
    {
      "id": "10",
      "label": "Object rights 10",
      "entityId": 4400
    }
  ]
}
]
}

```

13.8.3.2 Find ACL objects by id

Now we load the ACL objects by the ACL id. Each ACL object has to be loaded with a separate request.

Request - GET

GET - http://host:1512/rest/V1.0/security/acl/9

Response

```
{
  "id": 9,
  "entries": [
    {
      "principal": {
        "id": "AllRights",
        "entityId": 2700
      },
      "fullPermission": true,
      "deletePermission": true,
      "writePermission": true,
      "readPermission": true
    },
    {
      "principal": {
        "id": "Heiler Product Manager",
        "entityId": 2800
      },
      "fullPermission": false,
      "deletePermission": false,
      "writePermission": true,
      "readPermission": true
    }
  ]
}
```

Load ACL object for id 10.

Request - GET

GET - http://host:1512/rest/V1.0/security/acl/10

Response

```

{
  "id": 10,
  "entries": [
    {
      "principal": {
        "id": "specialUser",
        "entityId": 2600
      },
      "fullPermission": true,
      "deletePermission": true,
      "writePermission": true,
      "readPermission": true
    },
    {
      "principal": {
        "id": "Heiler Product Manager",
        "entityId": 2800
      },
      "fullPermission": false,
      "deletePermission": false,
      "writePermission": false,
      "readPermission": true
    }
  ]
}

```

13.8.3.3 Create ACL object in target system

Be sure that the principal references of the ACL objects are set up correct. Each ACL object gets created by a separate request.

Request - POST

POST - <http://host:1512/rest/V1.0/security/acl>

POST Body

```

{
  "entries": [
    {
      "principal": {
        "id": "AllRights",
        "entityId": 2700
      }
    }
  ]
}

```

```

    },
    "fullPermission": true,
    "deletePermission": true,
    "writePermission": true,
    "readPermission": true
  },
  {
    "principal": {
      "id": "Heiler Product Manager",
      "entityId": 2800
    },
    "fullPermission": false,
    "deletePermission": false,
    "writePermission": true,
    "readPermission": true
  }
]
}

```

Response

```

{
  "id": 25,
  "entries": [
    {
      "principal": {
        "id": "AllRights",
        "entityId": 2700
      },
      "fullPermission": true,
      "deletePermission": true,
      "writePermission": true,
      "readPermission": true
    },
    {
      "principal": {
        "id": "Heiler Product Manager",
        "entityId": 2800
      },
      "fullPermission": false,
      "deletePermission": false,
      "writePermission": true,
      "readPermission": true
    }
  ]
}

```


Create 2nd ACL object (source system id **10**).

Request - POST

POST - http://host:1512/rest/V1.0/security/acl

POST Body

```
{
  "entries": [
    {
      "principal": {
        "id": "specialUser",
        "entityId": 2600
      },
      "fullPermission": true,
      "deletePermission": true,
      "writePermission": true,
      "readPermission": true
    },
    {
      "principal": {
        "id": "Heiler Product Manager",
        "entityId": 2800
      },
      "fullPermission": false,
      "deletePermission": false,
      "writePermission": false,
      "readPermission": true
    }
  ]
}
```

Response

```
{
  "id": 26,
  "entries": [
    {
      "principal": {
        "id": "specialUser",
        "entityId": 2600
      },

```

```

    "fullPermission": true,
    "deletePermission": true,
    "writePermission": true,
    "readPermission": true
  },
  {
    "principal": {
      "id": "Heiler Product Manager",
      "entityId": 2800
    },
    "fullPermission": false,
    "deletePermission": false,
    "writePermission": false,
    "readPermission": true
  }
]
}

```

13.8.3.4 Create items in target system with references to created ACL objects

In that sample call only the items with the corresponding ACL references are created. Of course it is also possible to create the items first and update the items with a 2nd REST call to set the ACL references.

Please note: The reference has to be done with the ACL object id generated by the target system. In our sample the ACL objects are related to each other like follows:

ACL id source system	<u>ACL id target system</u>
9	<u>25</u>
10	<u>26</u>

Request - POST

POST - http://host:1512/rest/ V1.0/list/Article

POST Body

```

{
  "columns": [
    {
      "identifier": "Article.AclFlag"
    },
    {
      "identifier": "Article.AclProxy"
    }
  ],
  "rows": [
    {
      "object": {
        "id": "'itemToTransfer001'@'CatalogToTransfer'"
      },
      "values": [
        "Individual",
        "25"
      ]
    },
    {
      "object": {
        "id": "'itemToTransfer002'@'CatalogToTransfer'"
      },
      "values": [
        "Individual",
        "26"
      ]
    },
    {
      "object": {
        "id": "'itemToTransfer003'@'CatalogToTransfer'"
      },
      "values": [
        "Individual",
        "26"
      ]
    }
  ]
}

```

Response

```

{
  "counters": {
    "errors": 0,
    "warnings": 0,
    "createdObjects": 3,
    "updatedObjects": 0,
  }
}

```

```

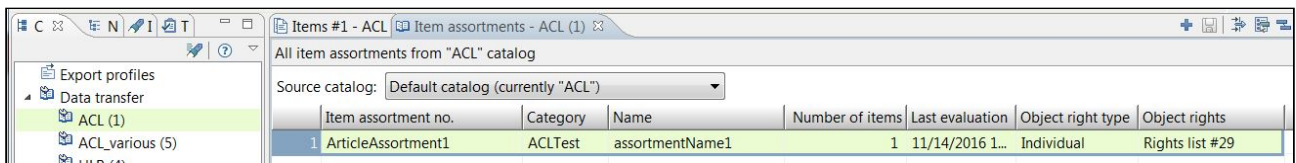
"objectsWithErrors": 0,
"objectsWithWarnings": 0
},
"entries": [],
"objects": [
  {
    "row": 0,
    "object": {
      "id": "2145@1009",
      "label": "itemToTransfer001",
      "entityId": 1000
    },
    "status": [
      "CREATED"
    ]
  },
  {
    "row": 1,
    "object": {
      "id": "2146@1009",
      "label": "itemToTransfer002",
      "entityId": 1000
    },
    "status": [
      "CREATED"
    ]
  },
  {
    "row": 2,
    "object": {
      "id": "2147@1009",
      "label": "itemToTransfer003",
      "entityId": 1000
    },
    "status": [
      "CREATED"
    ]
  }
]
}

```

13.8.4 More example for assignment of entity with new ACL reference ids in target system

13.8.4.1 Assortment

ArticleAssortment is used as example:



The screenshot shows the Informatica MDM interface. On the left, a tree view displays 'Export profiles', 'Data transfer', 'ACL (1)', 'ACL_various (5)', and 'LUP (4)'. The main pane is titled 'Items #1 - ACL' and 'Item assortments - ACL (1)'. It shows 'All item assortments from "ACL" catalog' with a 'Source catalog' dropdown set to 'Default catalog (currently "ACL")'. Below this is a table with the following data:

	Item assortment no.	Category	Name	Number of items	Last evaluation	Object right type	Object rights
1	ArticleAssortment1	ACLTest	assortmentName1	1	11/14/2016 1...	Individual	Rights list #29

Request - POST

POST - <http://host:1512/rest/V1.0/list/ArticleAssortment>

POST Body

```
{
  "columns": [
    {
      "identifier": "ArticleAssortment.AclFlag"
    },
    {
      "identifier": "ArticleAssortment.AclProxy"
    }
  ],
  "rows": [
    {
      "object": {
        "id": "'ArticleAssortment1'@'ACL'"
      },
      "values": [
        "Individual",
        "29"
      ]
    }
  ]
}
```

Response

```
{
  "counters": {
    "errors": 0,
    "warnings": 0,
```

```

    "createdObjects": 0,
    "updatedObjects": 1,
    "objectsWithErrors": 0,
    "objectsWithWarnings": 0
  },
  "entries": [],
  "objects": [
    {
      "row": 0,
      "object": {
        "id": "4",
        "label": "assortmentName1",
        "entityId": 2100
      },
      "status": [
        "UPDATED"
      ]
    }
  ]
}

```

13.8.4.2 Export template

Get id-name map for all export templates

We need a trick to get all id-name mapping for export template, because no identifier field is defined for the Entity "ExportTemplate" in standard Product 360, which is but meaningful for REST call.

Set the entity ExportTemplate to support the service API in Repository

The screenshot shows the Informatica MDM configuration interface. On the left, a tree view under 'Document Root' shows the hierarchy: <repository> Repository > Types > Custom. The 'ExportTemplate [Export format template]' entity is selected. On the right, a configuration table for 'ExportTemplate' is displayed.

Active	<input checked="" type="checkbox"/> true
Entity Id	3800
Entity Type	GenericDataE
Identifier	ExportTempla
▲ Misc	
Child Entities	
Container Entity	
Container Field	
Identifier Field	
Parent Entities	
Searchable	<input checked="" type="checkbox"/> true
Supports Data Quality	<input checked="" type="checkbox"/> false
Supports import	<input checked="" type="checkbox"/> true
Supports service API	<input checked="" type="checkbox"/> true
▲ Persistence	
Deletion Mode	BOTH
▲ Presentation	
Default View ID	
Description	
Documentation	

Call following REST to get id-name map:

Request - POST

GET - [http://host:1512/rest/V1.0/list/ExportTemplate/bySearch?query=ExportTemplate.Name != ""](http://host:1512/rest/V1.0/list/ExportTemplate/bySearch?query=ExportTemplate.Name != \)

Response

```
{
  "cacheld": "20161114_173525_0",
  "entityIdentifier": "ExportTemplate",
  "totalSize": 3,
  "startIndex": 0,
  "pageSize": 100,
  "rowCount": 3,
  "columnCount": 0,
  "columns": [],
  "rows": [
```

```

{
  "object": {
    "id": "2",
    "label": "Supplier export template",
    "entityId": 3800
  },
  "values": []
},
{
  "object": {
    "id": "9",
    "label": "exportWithMediaPortal_EN",
    "entityId": 3800
  },
  "values": []
},
{
  "object": {
    "id": "10",
    "label": "ExportTemp1",
    "entityId": 3800
  },
  "values": []
}
]
}

```

Assign ACL to desired export template

Then you can assign ACL to ExportTemplate like other root entities:

Request - POST

POST - http://host:1512/rest/ V1.0/list/ExportTemplate

POST Body

```
{
  "columns": [
    {
      "identifier": "ExportTemplate.AclFlag"
    },
    {
      "identifier": "ExportTemplate.AclProxy"
    }
  ],
  "rows": [
    {
      "object": {
        "id": "10"
      },
      "values": [
        "Individual",
        "25"
      ]
    }
  ]
}
```

Response

```
{
  "counters": {
    "errors": 0,
    "warnings": 0,
    "createdObjects": 0,
    "updatedObjects": 1,
    "objectsWithErrors": 0,
```

```
"objectsWithWarnings": 0
},
"entries": [],
"objects": [
  {
    "row": 0,
    "object": {
      "id": "10",
      "label": "ExportTemp1",
      "entityId": 3800
    },
    "status": [
      "UPDATED"
    ]
  }
]
```

Copyright

© Copyright Informatica LLC 1993, 2024

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica and the Informatica logo are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at <https://www.informatica.com/trademarks.html>. Other company and product names may be trade names or trademarks of their respective owners.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.