



# Informatica<sup>TM</sup>

## REST Service API

Informatica MDM - Product 360

Version: 10.1

## Table of Contents

<b>1</b>	<b>Available APIs .....</b>	<b>17</b>
1.1	List API .....	17
1.2	Management API .....	17
1.3	Enum API .....	17
1.4	Meta API .....	18
1.5	Media API .....	18
1.6	Security API .....	18
1.7	Object API .....	18
<b>2</b>	<b>Getting Started.....</b>	<b>19</b>
2.1	Authentication .....	19
<b>3</b>	<b>General Information.....</b>	<b>19</b>
3.1	Data Representation .....	19
3.2	Entity Item Reference .....	19
3.3	Field Qualification .....	20
3.4	Search Query Language.....	20
3.5	Rest Java Client .....	20
3.6	Rest API Versions .....	20
3.6.1	List of changes in V2.0.....	20
<b>4</b>	<b>REST General Information .....</b>	<b>21</b>
4.1	REST Service Authentication .....	21
4.1.1	Basic Authentication.....	21
4.1.1.1	Simple example.....	21
4.1.1.2	Username / Password Basic Authentication .....	22
4.1.1.3	Token Basic Authentication .....	22
4.1.2	OAuth .....	23
4.2	REST Datatypes .....	23
4.2.1	Datatypes .....	23
4.2.1.1	STRING.....	23
4.2.1.2	INTEGER.....	23
4.2.1.3	DECIMAL.....	23

4.2.1.4	DATE.....	24
4.2.1.5	TIME .....	24
4.2.1.6	DATETIME .....	24
4.2.1.7	BOOLEAN .....	25
4.2.1.8	ENTITY_ITEM .....	25
4.2.1.9	MIME_VALUE.....	26
4.2.1.10	ANY.....	27
4.3	REST Field Qualification .....	27
4.3.1	Syntax .....	28
4.3.2	Logical Key Value Definition .....	28
4.3.2.1	With Enumeration .....	29
4.3.2.2	Without Enumeration .....	29
4.3.2.3	Using default values.....	30
4.3.2.4	Using wildcard matching.....	31
4.4	REST Transition Fields .....	31
4.4.1	Syntax .....	31
4.4.2	Examples .....	32
4.5	REST Search Query Language .....	32
4.5.1	Field Qualification.....	33
4.5.2	Characteristic .....	33
4.5.3	Operators.....	33
4.5.4	Value Literal.....	35
4.5.4.1	Dynamic Value.....	35
4.6	REST Java Client .....	35
4.6.1	Example.....	35
4.7	REST Characteristic.....	36
4.7.1	Characteristic Operand.....	37
4.7.1.1	Syntax .....	37
4.7.1.2	Usage examples .....	38
4.7.2	DESCENDANT Operator .....	42
4.7.2.1	Syntax .....	42
4.7.2.2	Limitations .....	43
4.7.2.3	Usage Examples .....	43
4.7.3	Limitation .....	45

<b>5</b>	<b>REST List API.....</b>	<b>46</b>
5.1	Overview .....	46
5.2	List-Based Mass Data Access .....	46
5.3	Root Entity vs. Sub Entity Access .....	46
5.4	Report vs. Search .....	46
5.5	Report Services .....	47
5.6	Search Services .....	47
5.7	Examples .....	47
5.8	REST List API Info .....	47
5.8.1	List all entities with reports .....	48
5.8.1.1	Result .....	48
5.8.2	List all available reports for an entity .....	48
5.8.2.1	Result .....	49
5.8.3	List report specific parameters .....	49
5.8.3.1	Result .....	50
5.8.4	Examples .....	50
5.8.4.1	List all entities with reports .....	50
5.8.4.2	Retrieving all reports for the entity Article .....	51
5.8.4.3	Retrieving all report specific parameters for the report byCatalog .....	51
5.9	REST List API Read .....	52
5.9.1	Caching .....	53
5.9.1.1	Use Caching with Paging .....	53
5.9.1.2	Disable Caching.....	53
5.9.2	REST List API Read for Root Entities.....	53
5.9.2.1	Executing a specific report .....	54
5.9.2.2	Execute a specific report for root entities for MIME values .....	66
5.9.3	REST List API Read for Sub Entities .....	68
5.9.3.1	Execute a specific report for Read.....	68
5.9.3.2	Execute a specific report of sub entities for MIME values .....	80
5.9.4	REST List API Read MIME files of Root and Sub Entities .....	83
5.9.4.1	Root Entity .....	83
5.9.4.2	Sub Entity .....	86
5.10	REST List API Write .....	89
5.10.1	REST List API Write for Root Entities .....	89

5.10.1.1	Write values into qualified fields .....	89
5.10.1.2	Examples .....	92
5.10.2	REST List API Write for Sub Entities.....	101
5.10.2.1	Write values into fields of a certain sub entity.....	101
5.10.2.2	Examples .....	103
5.11	REST List API Delete .....	112
5.11.1	Delete all objects returned by a report .....	112
5.11.1.1	Result .....	113
5.11.2	Delete sub entity records of objects returned by a report.....	113
5.11.2.1	Result .....	114
5.11.3	Examples .....	115
5.11.3.1	Delete all item in the supplier catalog TOOLS.....	115
5.11.3.2	Delete all sales prices with currency USD in the supplier catalog TOOLS .....	116
5.12	REST List API Errors.....	116
5.12.1	General List API Errors .....	116
5.12.2	Report Specific Errors .....	117
5.12.3	Search Specific Errors .....	117
<b>6</b>	<b>REST Object API.....</b>	<b>118</b>
6.1	Read Requests.....	118
6.1.1	Single Item.....	118
6.1.2	Multiple Items (since 10.1.0.02).....	119
6.1.3	Query Parameters .....	121
6.1.4	Permissions .....	122
6.1.4.1	Object Permissions .....	122
6.1.4.2	Field Permissions .....	122
6.1.4.3	Qualification Permissions (aka Qualified Field Permissions) .....	122
6.1.5	Result .....	123
6.1.5.1	Repository Entities and Fields.....	123
6.1.5.2	Meta attributes.....	123
6.1.5.3	Data Element.....	123
6.1.5.4	Example Result.....	124
6.1.5.5	Characteristic Values .....	147
6.1.6	Examples .....	153
<b>7</b>	<b>REST Management API.....</b>	<b>154</b>

7.1	REST Assortment API .....	155
7.1.1	Get assortment content.....	155
7.2	REST Data Quality API .....	155
7.2.1	Rule Execution (since 8.0.03) .....	156
7.2.1.1	Content .....	156
7.2.1.2	Result .....	157
7.2.1.3	Examples .....	158
7.2.2	Schedule Rule Execution .....	161
7.2.2.1	Query Parameters .....	162
7.2.2.2	Content .....	162
7.2.2.3	Result .....	163
7.2.2.4	Workflow Callback (since 8.0.03).....	163
7.2.2.5	Workflow Callback (since 10.0).....	164
7.2.3	Examples .....	165
7.2.3.1	Executing a DQ checks for the catalog TOOLS .....	165
7.2.4	Deprecated .....	167
7.2.4.1	Execute a data quality rule configuration group immediately .....	167
7.2.4.2	Execute a set of data quality rule configurations .....	168
7.2.4.3	Execute channel's set of data quality rule configurations.....	169
7.2.4.4	Examples .....	170
7.3	REST Environment API .....	171
7.3.1	REST Environment APIs used for Extraction, Download and Integration of Environment transfer .....	171
7.3.2	Extraction .....	171
7.3.2.1	Query Parameters .....	171
7.3.2.2	Result .....	172
7.3.3	Download .....	172
7.3.3.1	URL Parameters .....	173
7.3.3.2	Result .....	173
7.3.4	Integration.....	173
7.3.4.1	Query Parameters .....	173
7.3.4.2	Form Data Parameters.....	174
7.3.4.3	Result .....	174
7.3.5	Checking the Progress of the Extraction/Integration Job .....	175
7.3.5.1	URL Parameters .....	175
7.3.5.2	Result .....	175

7.3.5.3	Possible Values of "currentState" .....	176
7.4	REST Export API.....	178
7.4.1	Schedule Export Job .....	178
7.4.1.1	Query Parameters .....	179
7.4.1.2	Content .....	179
7.4.1.3	Result .....	182
7.4.1.4	Complete example .....	183
7.4.2	Information about an export profile .....	183
7.4.2.1	Result .....	184
7.4.2.2	Example .....	184
7.4.3	Cancel Export Job .....	184
7.4.3.1	URL Parameters .....	185
7.4.3.2	Result .....	185
7.5	REST File API.....	185
7.5.1	Uploading a File .....	185
7.5.1.1	Result .....	186
7.5.2	Examples .....	186
7.5.2.1	Uploading a file with file name Data1.xls .....	186
7.6	REST Import API .....	187
7.6.1	Schedule Import Job .....	187
7.6.1.1	Query Parameters .....	187
7.6.1.2	Content .....	188
7.6.1.3	Result .....	190
7.6.2	Cancel Import Job.....	191
7.6.2.1	Result .....	191
7.6.3	Examples .....	191
7.6.3.1	Scheduling an import for a certain date and time .....	191
7.6.3.2	Cancel a running import job.....	192
7.7	REST KPI API .....	192
7.7.1	Schedule calculation of key performance indicator (KPI) values.....	193
7.7.1.1	Content .....	193
7.7.1.2	Result .....	194
7.7.1.3	Examples .....	195
7.7.2	Remove persisted results of key performance indicator (KPI) calculations .....	196
7.7.2.1	Content .....	196

7.7.2.2	Result .....	197
7.7.2.3	Examples .....	198
7.8	REST Merge API .....	200
7.8.1	Schedule Merge Job.....	200
7.8.1.1	Query Parameters .....	200
7.8.1.2	Content .....	201
7.8.1.3	Result .....	208
7.8.2	Workflow Callback .....	208
7.8.2.1	Workflow Callback (since 10.0).....	208
7.8.2.2	Workflow Callback Example .....	208
7.8.3	Examples .....	209
7.8.3.1	Schedule merge of the full "Apparel" catalog.....	209
7.9	REST Revision API .....	210
7.9.1	Schedule Revision Release Job .....	211
7.9.1.1	Query Parameters .....	211
7.9.1.2	Content .....	213
7.9.1.3	Result .....	216
7.9.2	Get Revision Job Information.....	216
7.9.2.1	Result .....	217
7.10	REST System API .....	218
7.10.1	Create a thread dump.....	218
7.10.2	Create thread dumps for all servers.....	219
7.10.3	Echo Text .....	219
7.10.3.1	Example: .....	219
7.10.4	Create token for Token Basic Authentication.....	220
7.10.4.1	Example: .....	220
7.10.5	Create token for Token Basic Authentication, based on a SAML Response.....	221
7.10.5.1	Example: .....	221
7.10.6	REST System API for License .....	222
7.10.6.1	Retrieve License Information .....	222
7.11	REST Task API .....	223
7.11.1	Create Task.....	223
7.11.1.1	Content .....	223
7.11.1.2	Result .....	228
7.11.2	Update Task .....	228



7.11.2.1 Content .....	229
7.11.3 Get Task .....	229
7.11.4 Delete Task .....	229
7.11.5 Update Task Content .....	229
7.11.5.1 Content .....	230
7.11.6 Get Task Content .....	230
7.11.7 Workflow callback .....	230
7.11.8 Information Page .....	231
7.11.9 Examples .....	231
7.11.9.1 Create a task for all items in the catalog TOOLS .....	231
7.11.9.2 Create a task for supplier .....	232
7.11.9.3 Update a task: Change name and description .....	232
7.11.9.4 Update a task's content: Add another item .....	233
7.12 REST Workflow API .....	233
7.12.1 Create / Update Workflow Details .....	234
7.12.1.1 General Info .....	234
7.12.1.2 Content .....	234
7.12.1.3 Example .....	237
7.12.2 Get Workflow Details .....	238
7.12.2.1 General Info .....	239
7.12.2.2 Parameters .....	239
7.12.3 Enter Workflow Process Status .....	240
7.12.3.1 General Info .....	240
7.12.3.2 Content .....	240
7.12.3.3 Example .....	241
7.12.4 Leave Workflow Process Status .....	242
7.12.4.1 General Info .....	242
7.12.4.2 Content .....	242
7.12.4.3 Example .....	243
7.12.5 End Workflow Process .....	243
7.12.5.1 General Info .....	243
7.12.5.2 Parameters .....	244
7.12.5.3 Example .....	244
7.12.6 Get Workflow Process Status of an entity item per process .....	244
7.12.6.1 General Info .....	244

7.12.6.2	Parameters .....	245
7.12.6.3	Example .....	245
7.12.7	Add error message to the BPM perspective protocol view .....	245
7.12.7.1	General Info .....	246
7.12.7.2	Parameters .....	246
7.12.8	REST Workflow Task .....	246
7.12.8.1	Example .....	246
7.12.8.2	WorkflowTask.....	248
7.12.8.3	Task Callback .....	249
<b>8</b>	<b>REST Enumeration API.....</b>	<b>254</b>
8.1	All Enumerations .....	254
8.1.1	Result .....	254
8.2	Meta Data and Enumeration Entries .....	255
8.2.1	Result .....	255
8.3	Examples .....	256
8.3.1	Retrieving all enumerations .....	256
8.3.2	Retrieving the entries of enumeration Enum.Languages .....	257
8.3.3	Retrieving the entries of enumeration Enum.SupplierWithMainSupplier .....	258
<b>9</b>	<b>REST Media API.....</b>	<b>259</b>
9.1	Single Media Asset File.....	259
9.2	Preview of Media Asset File .....	259
9.3	Media Asset File Location .....	260
9.4	Upload single Media Asset File .....	260
9.5	Examples .....	261
9.5.1	Retrieve single media asset file in 36dpi JPEG quality with identifier D123456 .....	261
9.5.2	Retrieve media asset file preview.....	262
9.5.3	Retrieve media asset file location .....	262
9.5.4	Upload media asset file to media asset server .....	262
9.6	REST List API for MediaAssetFile entity.....	264
9.6.1	Adjustments for MediaAssetFile entity .....	264
9.6.1.1	Internal id .....	264
9.6.1.2	Special reports .....	265
9.6.2	Examples .....	266

9.6.2.1	Retrieve media asset file with "byIdentifiers" report for external id .....	266
9.6.2.2	Retrieve media asset file with "byItems" report for internal id .....	267
9.6.2.3	Retrieve media asset files which belong (be assigned) to a category and are used in PIM core .....	268
9.6.2.4	Retrieve all media asset files which belong to a category or its child categories.....	269
9.6.2.5	Update media asset file(Root Entity) .....	270
9.6.2.6	Update media asset file language special attribute(Sub Entity) .....	271
9.6.2.7	Delete media asset file with identifier D11000100112532 .....	273
9.6.2.8	Create media asset file.....	273
9.6.3	Media search query for MediaAssetFile entity .....	273
9.6.3.1	Syntax for mediaQuery .....	274
9.6.3.2	Examples .....	277
9.6.4	Search query for MediaAssetFile entity .....	278
9.6.4.1	Syntax for query .....	279
9.6.4.2	Examples .....	283
9.7	REST Media Asset Category API .....	291
9.7.1	Get Media Asset Category .....	291
9.7.2	Get top Media Asset Categories.....	292
9.7.3	Get direct child Media Asset Categories.....	292
9.7.4	Add Media Asset Category .....	292
9.7.5	Update Media Asset Category .....	293
9.7.6	Delete Media Asset Category .....	293
9.7.7	Examples .....	293
9.7.7.1	Retrieve single media asset category with identifier .....	293
9.7.7.2	Retrieve all top media asset categories .....	294
9.7.7.3	Retrieve all direct child media asset categories of the category with identifier .....	295
9.7.7.4	Add a top media asset category .....	295
9.7.7.5	Add a child media asset category under the category with identifier .....	296
9.7.7.6	Update the name of the media asset category with identifier .....	297
9.7.7.7	Delete a media asset category with identifier .....	298
<b>10</b>	<b>REST Meta API .....</b>	<b>298</b>
10.1	All Root Entities .....	299
10.1.1	Result .....	299
10.2	Metadata for a Root Entity.....	300
10.2.1	Parameters .....	300

10.2.2	Result .....	300
10.3	All Fields of a Root Entity .....	301
10.3.1	Result .....	301
10.4	Metadata for a Sub Entity .....	302
10.4.1	Parameters .....	302
10.4.2	Result .....	302
10.5	Metadata for a Field .....	303
10.5.1	Parameters .....	303
10.5.2	Result .....	303
10.6	Metadata for a Qualifier .....	305
10.6.1	Result .....	305
10.7	Examples .....	306
10.7.1	Retrieving all root entities .....	306
10.7.2	Retrieving meta data of the root entity Article .....	306
10.7.3	Retrieving all fields of the root entity Article .....	308
10.7.4	Retrieving meta data of the sub entity ArticlePriceValuePurchase.....	308
10.7.5	Retrieving metadata for the field ArticleLang.DescriptionLong.....	309
10.7.6	Retrieving meta data of the field ArticlePriceVa.DescriptionLong with qualification .....	310
10.7.7	Retrieving meta data of the qualifier Language for entity ArticleLang .....	311
10.8	REST Meta Media API .....	311
10.8.1	Derivative definition .....	312
10.8.1.1	Get single derivative definition .....	312
10.8.1.2	Get all derivative definitions.....	313
10.8.2	Media Asset File Property field definition .....	313
10.8.2.1	Get single property field definition for one supported language .....	313
10.8.2.2	Get single property field definition for all supported languages.....	316
10.8.2.3	Get all property field definitions for all supported languages.....	317
10.8.3	Examples .....	317
10.8.3.1	Retrieve single derivative definition with identifier .....	317
10.8.3.2	Retrieve all derivative definitions .....	318
10.8.3.3	Retrieve single property field definition for one supported language .....	320
10.8.3.4	Retrieve single property field definition for all supported languages .....	322
10.8.3.5	Retrieve property field definitions for all supported languages.....	324
11	REST Security API .....	326

11.1	List all Security ACL APIs .....	326
11.1.1	Result .....	326
11.2	Read ACL object .....	327
11.2.1	Content .....	327
11.3	Create ACL object .....	328
11.4	Examples .....	330
11.4.1	Read ACL object for an id .....	330
11.4.1.1	REST call for JSON .....	330
11.4.1.2	REST call for XML .....	332
11.4.1.3	REST Client Java .....	333
11.4.2	Create ACL object .....	333
11.4.2.1	REST call for JSON .....	333
11.4.2.2	REST call for XML .....	334
11.4.2.3	REST Client Java .....	335
<b>12</b>	<b>REST API Tutorial .....</b>	<b>336</b>
12.1	Loading Data For Examples .....	336
12.2	Tools .....	336
12.3	Recording .....	336
12.4	REST API Tutorial - Data Model .....	336
12.4.1	Data Model .....	336
12.5	REST API Tutorial - List Read for Root Entities .....	338
12.5.1	1. Which objects should be included (rows) -> Report .....	338
12.5.2	2. Which fields are required (columns) .....	338
12.5.3	3. Options .....	338
12.5.4	Examples .....	338
12.5.4.1	Root entity fields only .....	338
12.5.4.2	Root fields and qualified sub entity fields .....	339
12.5.5	Transition field access .....	339
12.5.6	Paged access .....	340
12.6	REST API Tutorial - List Write for Root Entities .....	340
12.6.1	Update .....	340
12.6.1.1	Example .....	340
12.6.1.2	Example with error .....	341
12.6.2	Create .....	342

12.6.2.1 Example .....	342
<b>12.7 REST API Tutorial - List Read for Sub Entities.....</b>	<b>343</b>
12.7.1 Motivation .....	343
12.7.2 Query .....	343
12.7.2.1 Examples .....	343
12.7.3 Qualification filter .....	344
12.7.3.1 Example .....	344
<b>12.8 REST API Tutorial - List Write for Sub Entities .....</b>	<b>344</b>
12.8.1 Example .....	344
<b>13 REST API How To's .....</b>	<b>346</b>
13.1 Updating Product's Structure Group with Rest API .....	346
13.2 Assigning User Groups to a User .....	347
13.3 Monitor all REST requests in the log file .....	347
13.4 Query the list of items assigned to multiple structure groups of a structure .....	348
13.5 How to read and write values for characteristics.....	349
13.5.1 Definitions .....	349
13.5.2 Qualifications .....	350
13.5.3 Datatypes .....	350
13.5.4 Fields.....	351
13.5.4.1 Special Case: LookupValues .....	351
13.5.5 Read Access .....	351
13.5.5.1 Examples based on the root entity .....	351
13.5.5.2 Examples based on the sub-entity .....	354
13.5.6 Write Access.....	371
13.5.6.1 Examples based on the root entity .....	371
13.5.6.2 Examples based on the sub-entity .....	372
<b>14 REST API Troubleshooting.....</b>	<b>378</b>
14.1 Characteristics .....	378
14.2 Structure Groups.....	378
14.2.1 Access to structure group fails .....	378
14.2.2 StructureGroup levels.....	379
14.2.3 Filtering on structure group attribute.....	379
14.2.4 Not possible to search by Structure Group Attribute Value.....	380

14.3	Media Assets .....	380
14.3.1	Get media asset document with attributes fails .....	380
<b>15</b>	<b>PUBLIC Server Health Status .....</b>	<b>381</b>
15.1	Overview .....	381
15.1.1	Health Status Page.....	381
15.1.1.1	Example .....	381
15.1.2	Info REST API .....	382
15.1.2.1	Example .....	382

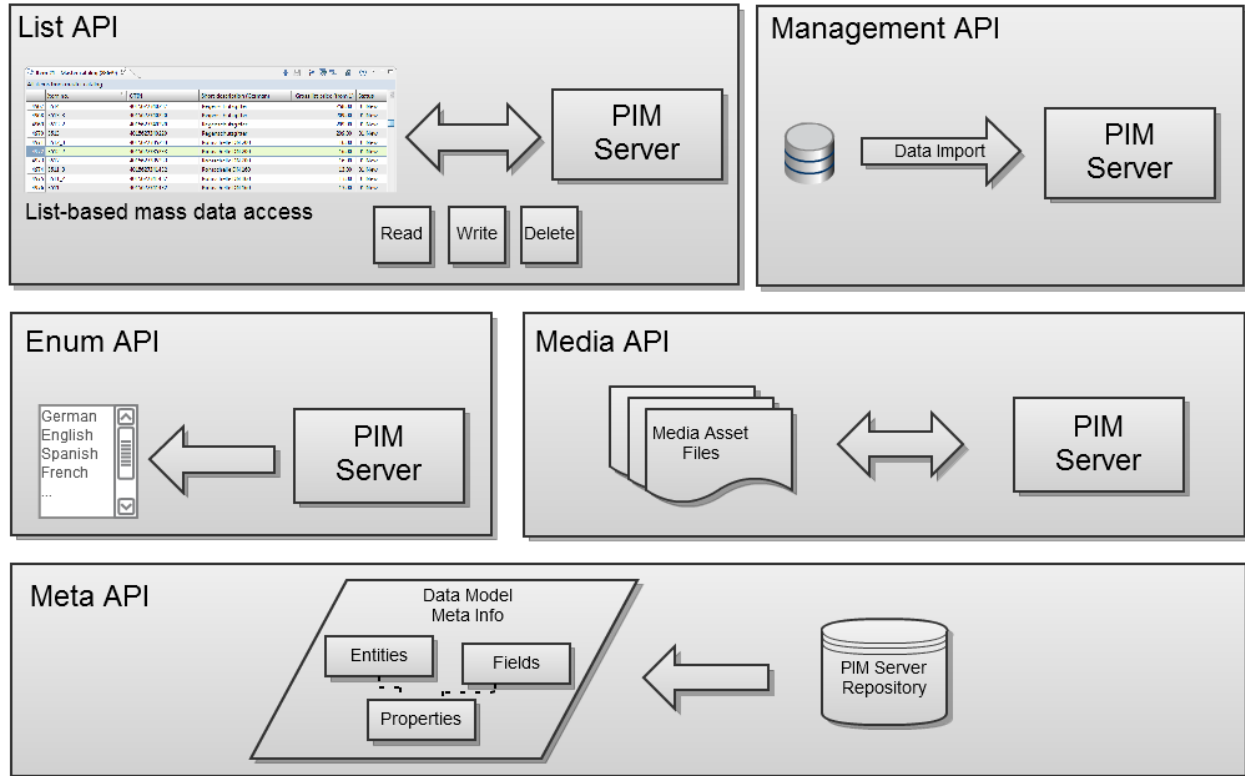
Based on the REST Service Infrastructure introduced in Version 6 we created a set of services which allows you to read and write almost any data which is described by the repository. In general we provide services to create, read, update, and delete lists of entity items with a configurable set of field 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 17)
  - [List API](#)(see page 17)
  - [Management API](#)(see page 17)
  - [Enum API](#)(see page 17)
  - [Meta API](#)(see page 18)
  - [Media API](#)(see page 18)
  - [Security API](#)(see page 18)
  - [Object API](#)(see page 18)
- [Getting Started](#)(see page 19)
  - [Authentication](#)(see page 19)
- [General Information](#)(see page 19)
  - [Data Representation](#)(see page 19)
  - [Entity Item Reference](#)(see page 19)
  - [Field Qualification](#)(see page 20)
  - [Search Query Language](#)(see page 20)
  - [Rest Java Client](#)(see page 20)
  - [Rest API Versions](#)(see page 20)



# 1 Available APIs



## 1.1 List API

The [REST List API](#) (see page 46) 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.

## 1.2 Management API

The [.REST Management API v8.0](#) (see page 154) 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.

## 1.3 Enum API

The [.REST Enumeration API v8.1](#) (see page 254) provides read-only access to all enumerations of the Product Manager. The enum entry keys are used in field values as well as for the qualification of fields.

## 1.4 Meta API

The [R\(see page 298\)EST Meta API\(see page 298\)](#) 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.

## 1.5 Media API

The REST Media API provides access to all kinds of media assets in the Product Manager.

## 1.6 Security API

The REST Security API provides access to securities of the Product Manager.

## 1.7 Object API

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\(see page 0\)](#)
  - [Single Item\(see page 0\)](#)
  - [Multiple Items \(since 10.1.0.02\)\(see page 0\)](#)
  - [Query Parameters\(see page 0\)](#)
  - [Permissions\(see page 0\)](#)
    - [Object Permissions\(see page 0\)](#)
    - [Field Permissions\(see page 0\)](#)
    - [Qualification Permissions \(aka Qualified Field Permissions\)\(see page 0\)](#)
- [Result\(see page 0\)](#)
  - [Repository Entities and Fields\(see page 0\)](#)
    - [Names\(see page 0\)](#)
    - [Datatypes\(see page 0\)](#)
  - [Meta attributes\(see page 0\)](#)
  - [Data Element\(see page 0\)](#)
  - [Example Result\(see page 0\)](#)
  - [Characteristic Values\(see page 0\)](#)
    - [Additional Meta Attributes\(see page 0\)](#)
    - [Characteristic Records Example\(see page 0\)](#)
- [Examples\(see page 0\)](#)

## 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 16) 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`

### 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 21).

## 3 General Information

### 3.1 Data Representation

The [REST Service API v10.1](#)(see page 16) 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 23).

### 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.

### 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 27).

### 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 32).

### 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 [Java client](#)(see page 35) 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.

### 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.

#### 3.6.1 List of 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 <b>false</b>,</p> <ul style="list-style-type: none"> <li>▪ In V1.0 the enumeration label is returned</li> <li>▪ In V2.0 the enumeration key is returned.</li> </ul>
includeLabels	<ul style="list-style-type: none"> <li>▪ New default value is false</li> </ul>

Change	Description
	<ul style="list-style-type: none"> <li>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.</li> </ul>
cached	New default value in responses is <b>no-cache</b> for the case, the client doesn't request using caching.

## 4 REST General Information

### 4.1 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 16) which comes out of the box.

- [Basic Authentication](#) (see page 21)
  - [Simple example](#) (see page 21)
  - [Username / Password Basic Authentication](#) (see page 22)
  - [Token Basic Authentication](#) (see page 22)
- [OAuth](#) (see page 23)

#### 4.1.1 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!

##### 4.1.1.1 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
```

#### 4.1.1.2 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 bXlvc2VybmFtZTppeVBhc3N3b3Jk" -H "Content-Type: application/json" http://hpm.heiler.com:1501/rest/V1.0/list/suppliercatalog/all
```

#### 4.1.1.3 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 218). 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 218). 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
jA1MzFGMzE5Q0E0QkM5OTczRjAyM0QyQzc2RUeWQTNODQ2ODQ3MTQyMjM5NjdBOTIyM0UyNTlCQzNDOTI1Q0
MwQkM2MEIyRjk5RTg5MTBDRDFNERERUMzNENGM0MzNUY2OEY3NTM0MTdGMUM5MzU1NzQwRjJFMUVDMkJDNTJ
CMUE5RTg4OUFFMjc0RUM0MDc0NURDNDJCQTM0Y0ERGRjVGRDNGRjY1ODk3RDk2NEVDMkE2MEI5RTcwOTI4ODlG
MERCQTM2MzdFNEIzOTAxMDYzOTQ4MjM4RjkxODcwQjk3RTgwOEZDQzZGMTIwMkIxNkIzQThCRkZERjIwMDkxO
UI5MDZBNzAwRUM3MzM1QjQ1RTY5QkM4NUM3MEEzMDY2N0U4OTkxRTQxQ0YwMUREN0FFOUY5QjJEMjk0NUE2Mj
Q2RjgyN0IyNUUzRUQ2MEQxNjVDN0U1OUQyOERENDcwNUQzRkU1QTM0YjJGRUZZCQUQzNjRDNTgNjg2NTQ2QjA
1OEQzQTdEMzE1QkQ4M0QzRjJFQUFCQUVENTM0RDE0RTFGMzdcNTkxRDQ5REVENTM1NjM2RkQyNTFGQ0NGNDRG
NUY=
```

### 4.1.2 OAuth

Not yet supported.

## 4.2 REST Datatypes

### 4.2.1 Datatypes

The [REST Datatypes](#)(see page 23) 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.

#### 4.2.1.1 STRING

<b>Definition</b>	Unicode Characters
<b>Pattern</b>	
<b>Example</b>	"Hello World!", "Heute ist ein schöner Tag", "世界您好!"
<b>Corresponding Java Datatype</b>	java.lang.String

#### 4.2.1.2 INTEGER

<b>Definition</b>	Integer Numbers (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
<b>Pattern</b>	[+/-] 0-9*
<b>Example</b>	4711
<b>Corresponding Java Datatype</b>	java.lang.Long

#### 4.2.1.3 DECIMAL

<b>Definition</b>	<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>
-------------------	--

	The fraction consists of a decimal point followed by zero or more decimal digits. The string must contain at least one digit in either the integer or the fraction.
<b>Pattern</b>	[+/-] 0-9*[.0-9*]
<b>Example</b>	100.90
<b>Corresponding Java Datatype</b>	java.math.BigDecimal

#### 4.2.1.4 DATE

<b>Definition</b>	Calendar Date (12/30/1899 to 12/31/9999)
<b>Pattern</b>	yyyy-MM-dd
<b>Example</b>	2012-05-01 (first of May in 2012)
<b>Corresponding Java Datatype</b>	java.sql.Date

#### 4.2.1.5 TIME

<b>Definition</b>	Time (00:00:00 to 24:00:00)
<b>Pattern</b>	HH:mmZ
<b>Example</b>	12:05 (five minutes after 12 o'clock noon)
<b>Corresponding Java Datatype</b>	java.sql.Time

#### 4.2.1.6 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	



	Input	Output
	yyyy-MM-dd'T'HH:mm:ss:SSSZ	
<b>Example</b>	2012-05-01T12:00:00 2012-05-01T12:00:00-0700 2012-05-01T12:00:00:000 2012-05-01T12:00:00:000-0700	
<b>Corresponding Java Datatype</b>	java.sql.Timestamp	

## 4.2.1.7 BOOLEAN

<b>Definition</b>	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".
<b>Pattern</b>	true / false
<b>Example</b>	true, false
<b>Corresponding Java Datatype</b>	java.lang.Boolean

## 4.2.1.8 ENTITY\_ITEM

<b>Definition</b>	<p>A Product Manager object which is described in the <a href="#">Meta API</a>(see page 298). 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 <a href="#">REST List API Read for Root Entities</a>(see page 53) page.</p>
-------------------	---

<b>Pattern</b>	<p>Entity Item ID Syntax:</p> <p>Entity Item without a container item:</p> <p>'&lt;external_identifier&gt;' &lt;internal_id&gt;</p> <p>Entity Item with a container item:</p> <p>'&lt;external_identifier&gt;' &lt;internal_id&gt;@'&lt;container_external_identifier&gt;' &lt;container_internal_id&gt;</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>
<b>Example</b>	<p>'SomeItem'@'SupplierCat1'</p> <p>12345@5</p> <p>'SomeSupplier'</p> <p>42</p> <p>'SomeItem\'WithASingleQuote'@'MASTER'</p>
<b>Corresponding Java Datatype</b>	com.heiler.pim.webservice.client.EntityItemReference

#### 4.2.1.9 MIME\_VALUE

<b>Definition</b>	<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 will 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>
<b>Pattern</b>	<i>No Simple string representation, it's always rendered as JSON/XML structure</i>

<b>Example</b>	<pre>{   "label": "examplePicture.jpg",   "mimeType": "image/jpg",   "relativeFilePath": "dir1\\dir2\\dir3\\ examplePicture. [sequenceToMakeFileNameUnique].jpg" }</pre>
<b>Corresponding Java Datatype</b>	com.heiler.pim.webservice.client.MIMEValueReference

## 4.2.1.10 ANY

<b>Definition</b>	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.
<b>Pattern</b>	see at the specific datatype
<b>Example</b>	see at the specific datatype
<b>Corresponding Java Datatype</b>	java.lang.Object

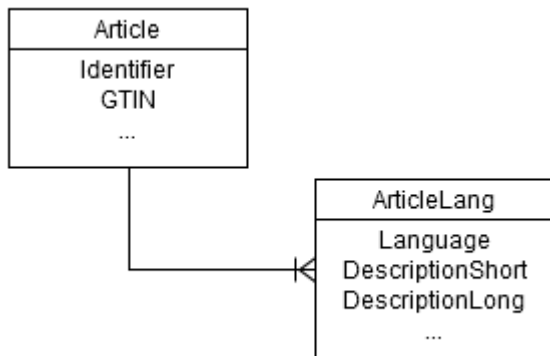
## 4.3 REST Field Qualification

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 28)
- [Logical Key Value Definition](#)(see page 28)
  - [With Enumeration](#)(see page 29)
  - [Without Enumeration](#)(see page 29)
  - [Using default values](#)(see page 30)
  - [Using wildcard matching](#)(see page 31)

Product Manager defines many business objects which are described in the Product Manager repository. The repository 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 it's child objects is described in the repository. You can find a detailed description about the Repository here.

An example would be the object "Article" and it's 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.

### 4.3.1 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"> <li>• Article.SupplierAID</li> <li>• Article.EAN</li> <li>• Article.ManufacturerName</li> </ul>
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>&lt;Parent LK 1, ParentLK 2, Own LK 1, Own LK 2&gt;</p>

### 4.3.2 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.

#### 4.3.2.1 With Enumeration

You can define everything which can be interpreted by the enumeration as a synonym to one of its 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 its 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 its 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.

#### 4.3.2.2 Without Enumeration

Logical keys without enumeration must be defined depending on their datatype. Please see the [REST Datatypes](#)(see page 23) 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)

Datatype	Format	Example
Constant	any string a-Z followed by zero or more 0-9	eng, de, public
ENTITY_ITEM	<p>Entity Item without a container item: '&lt;external_identifier&gt; &lt;internal_id&gt;</p> <p>Entity Item with a container item: '&lt;external_identifier&gt; &lt;internal_id&gt;@&lt;container_external_id&gt; &lt;container_internal_id&gt;</p> <p><b>Please don't forget use single quotes when you want to specify the external identifier of the entity item!</b></p>	<p>'Someltem'@'SupplierCat1'</p> <p>12345@5</p> <p>'SomeSupplier'</p> <p>42</p>
Dynamic Value	<p>A dynamic value is a placeholder which will be resolved while parsing the field qualification 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"> <li>• <code>\${CurrentDate}</code> = the current date (without time)</li> <li>• <code>\${CurrentTime}</code> = the current time (without date)</li> <li>• <code>\${CurrentTimestamp}</code> = the current date and time</li> </ul>

#### 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)`

#### 4.3.2.3 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.

#### 4.3.2.4 Using wildcard matching

If no value can be defined as qualification, it is possible to use the

`${Empty}`

qualifier.

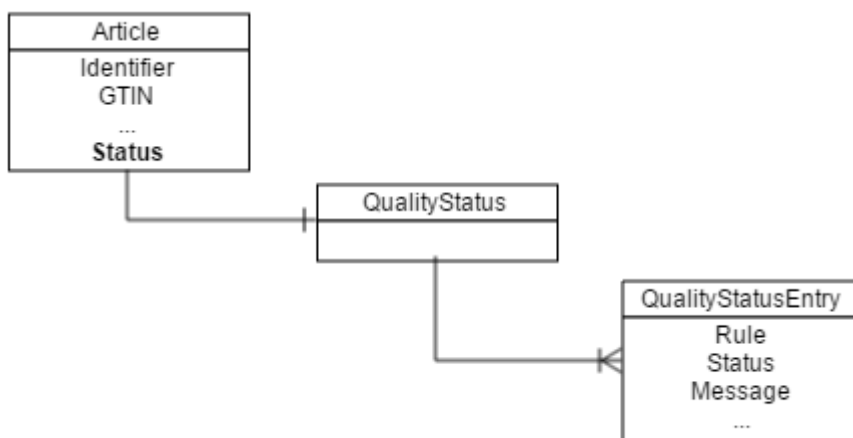
## 4.4 REST Transition Fields

The [REST List Read API](#)(see page 52) supports so-called transition fields. The syntax for such transition fields is documented on this page

- [Syntax](#)(see page 31)
- [Examples](#)(see page 32)


The PIM repository 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 27).

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.



#### 4.4.1 Syntax

Transition field: <Fully qualified field> -> ... -> <n-Fully qualified field>

Element	Definition
Fully qualified field	<p>A fully qualified field as described in <a href="#">REST Field Qualification</a>(see <a href="#">page 27</a>). The field on the left of the arrow (-&gt;) must be a proxy transition field pointing to another repository object, as e.g.</p> <ul style="list-style-type: none"> <li>Article.Status</li> <li>ArticleStructureMap.StructureGroup</li> <li>ArticleLogistic.PackageWeightUnitID</li> </ul> <div style="border: 1px solid orange; 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>

#### 4.4.2 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)*

### 4.5 REST Search Query Language

The HSQ syntax directly reflects the search expression API explained in Entity Search - 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
<p>HSQ Query: [&lt;CONDITION&gt;]+</p> <p>HSQ Condition: &lt;TERM&gt; [AND OR &lt;CONDITION&gt;]*   [NOT]? (&lt;CONDITION&gt;)</p> <p>HSQ Term: [NOT]? &lt;FIELD QUALIFICATION   CHARACTERISTIC&gt; &lt;OPERATOR&gt; &lt;VALUE LITERAL&gt;   [NOT]? &lt;FIELD QUALIFICATION   CHARACTERISTIC&gt; IN (&lt;VALUE LITERAL&gt; [,&lt;VALUE LITERAL&gt;]*)</p>

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
```

### 4.5.1 Field Qualification

The HSQ expression is built with a fully qualified field. Please see the [REST Field Qualification](#)(see page 27) documentation on the used syntax here. Note, only a single field is allowed in the context of the search query.

### 4.5.2 Characteristic

Please see the [REST Characteristic](#)(see page 36) documentation chapter on the used syntax here. Note, only a single characteristic is allowed in the context of the search query.

### 4.5.3 Operators

Operators are interpreted case insensitive, thus "equals" and "EQUALS" and "eQuAlS" are, equal 😊

Operator	Allowed Datatypes	Description
equals	String	Value must be <b>exactly</b> 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
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)

Operator	Allowed Datatypes	Description
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
<=	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 "\\\_"

#### 4.5.4

### Value Literal

The textual representation of the value is described in the [REST Datatypes](#)(see page 23). 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.

#### 4.5.4.1 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

## 4.6 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 [Java client](#)(see page 35) 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.

### 4.6.1 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
MyPassword </code>
9      */
10     public static void main( String[] args )
11     {
12         String basicUrl = extractBasicUrl( args );
13         String userName = extractUserName( args );
14         String password = extractPassword( args );
15         try
16         {
17             //Create and login to the client...
18             RestClient restClient = new RestClient();
19             restClient.loginWithBasicAuth( basicUrl, userName, password,
Locale.ENGLISH );
20             //Create and parameterize the report query you want to execute...
21             EntityItemReference masterCatalog =
EntityItemReferenceFactory.createByIdentifier( "MASTER" ); //$NON-NLS-1$
22             ReportQuery reportQuery = new ReportQuery( "byCatalog" ).addParamete
rValue( "catalog", masterCatalog ); //$NON-NLS-1$ //$NON-NLS-2$
23             //Create and parameterize the list request
24             ListReadRequest listReadRequest =
restClient.createListReadRequest();
25             EntityItemTable resultTable = listReadRequest.setFields(
"Article.SupplierAID" ) //$NON-NLS-1$
26                                                         .setPageSize( 100 )
27                                                         .setStartIndex( 0 )
28                                                         .getRootItems(
"Article", reportQuery ); //$NON-NLS-1$
29             for ( EntityItemTableRow row : resultTable )
30             {
31                 List< Object > values = row.getValues();
32                 String currentSupplierAid = ( String ) values.get( 0 );
33                 System.out.println( currentSupplierAid );
34             }
35         }
36         catch ( RestClientException e )
37         {
38             e.printStackTrace();
39         }
40     }

```

## 4.7 REST Characteristic

- [Characteristic Operand](#)(see page 37)
  - [Syntax](#)(see page 37)
  - [Usage examples](#)(see page 38)

- [DESCENDANT Operator](#)(see page 42)
  - [Syntax](#)(see page 42)
  - [Limitations](#)(see page 43)
  - [Usage Examples](#)(see page 43)
- [Limitation](#)(see page 45)

In this chapter describes how to use characteristics in the Service API. It introduces the characteristic operand and the DESCENDANT operator.

## 4.7.1 Characteristic Operand

### 4.7.1.1 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>'&lt;external_identifier&gt; &lt;internal_id&gt;</p> <p><b>Please don't forget to use single quotes when you want to specify the external identifier of the characteristic!</b></p>
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")
```

#### 4.7.1.2 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 32)

```
http://localhost:1512/rest/V1.0/list/Article/bySearch?query=characteristic('Ingredient') s
tartsWith "Bean"
```

We have these items with characteristic records:

ItemIdentifier	Characteristic	Characteristic record value
Item1	Ingredient	"Salt"
Item2	Ingredient	"Beans"
Item3	Ingredient	"Egg"

ItemIdentifier	Characteristic	Characteristic record value
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](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"](http://localhost:1512/rest/V1.0/list/Article/bySearch?query=characteristic('Numeric') = 15678 and characteristic('TerminationDate') = \)

We have these items with characteristic records:

ItemIdentifier	Characteristic	Characteristic record value		Characteristic	Characteristic record value
Item1	Numeric	12345		TerminationDate	2017-12-31
Item2	Numeric	15678		TerminationDate	2017-12-31
Item3	Numeric	15678		TerminationDate	2017-11-20

ItemIdentifier	Characteristic	Characteristic record value		Characteristic	Characteristic record value
Item4	Numeric	1335		TerminationDate	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**.

## 4.7.2 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.

### 4.7.2.1 Syntax

Operator	Allowed datatypes	Description
DESCENDANT	all datatypes except MIME	<p>This operator searches for characteristics having a hierarchical relationship.</p> <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

#### 4.7.2.2 Limitations

It is **not** possible to use the same characteristic more than once in one DESCENDANT expression. So the following example ist **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 ist **not** valid:

```
NOT (characteristic( 'ABC' ) equals value DESCENDANT characteristic( 'ABCD' ) equals
otherValue)
```

#### 4.7.2.3 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
Item3	Ingredient	Ingredients	Sugar

Item Identifier	Characteristic	Lookup of Characteristic	Characteristic record value
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
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
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**.

### 4.7.3 Limitation

Currently it is not possible to search for the datatype MIME in search expressions.

## 5 REST List API

The [REST List API](#)(see page 46) 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 46)
- [List-Based Mass Data Access](#)(see page 46)
- [Root Entity vs. Sub Entity Access](#)(see page 46)
- [Report vs. Search](#)(see page 46)
- [Report Services](#)(see page 47)
- [Search Services](#)(see page 47)
- [Examples](#)(see page 47)

### 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.

### 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. 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.

### 5.3 Root Entity vs. Sub Entity Access

As described in Domain Model (Repository), 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.

### 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 it's 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 page to learn more about Reports and how to contribute you 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 page to learn more about the possibilities of the generic search engine and also its limitations.

## 5.5 Report Services

Such a query can represent a **predefined report**, provided to the Product Manager by an entity report 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 )

## 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.

## 5.7 Examples

Additionally to the examples provided in this documentation, you can also use the ListAPI Postman collection: ListAPI.postman\_collection.json

## 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 48)
  - [Result](#)(see page 48)
- [List all available reports for an entity](#)(see page 48)
  - [Result](#)(see page 49)
- [List report specific parameters](#)(see page 49)
  - [Result](#)(see page 50)
- [Examples](#)(see page 50)
  - [List all entities with reports](#)(see page 50)
  - [Retrieving all reports for the entity Article](#)(see page 51)

- [Retrieving all report specific parameters for the report byCatalog](#)(see page 51)

### 5.8.1 List all entities with reports

Returns a list of all entities which have contributed reports

URL Pattern	/list/info
Method	GET
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.

#### 5.8.1.1 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

### 5.8.2 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.

#### 5.8.2.1 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

#### 5.8.3 List report specific parameters

Returns a list of all report specific parameters.

URL Pattern	/list/{entity-identifier}/{report-name}/info
Method	GET
Parameters	No parameters are available
Media types	text/html, application/json, application/xml
Result	Returns a list of report specific parameters. If text/html is request, also the general, report independent parameters are returned.

### 5.8.3.1 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	Objet	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.

## 5.8.4 Examples

### 5.8.4.1 List all entities with reports

#### Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/list/info
```

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();
```

#### 5.8.4.2 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" );
```

#### 5.8.4.3 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" );
```

## 5.9 REST List API Read

The [List API](#)([see page 46](#)) 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 53](#))
  - [Use Caching with Paging](#)([see page 53](#))
  - [Disable Caching](#)([see page 53](#))

## 5.9.1 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.

### 5.9.1.1 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.

### 5.9.1.2 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.

Therefore 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!)

## 5.9.2 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 27)) in case they belong to a sub entity of the requested root entity.

- [Executing a specific report](#)(see page 54)
  - [Parameters](#)(see page 54)
  - [Result](#)(see page 58)
  - [Examples](#)(see page 61)
  - [Executing the report byCatalog for the catalog TOOLS](#)(see page 61)
  - [Executing the report byCatalog for the catalog TOOLS and specifying a list of fields \(parameter formatData is false\)](#)(see page 62)
  - [Executing the report byCatalog for the catalog TOOLS and specifying a list of fields \(parameter formatData is true\)](#)(see page 64)
  - [Searching for root entity objects](#)(see page 65)
- [Execute a specific report for root entities for MIME values](#)(see page 66)
  - [Examples for MIME values](#)(see page 67)
  - [Rest Java Client Example for MIME values](#)(see page 67)

### 5.9.2.1 Executing a specific report

URL Pattern	/list/{entity-identifier}/{report-name}
Method	GET
Parameters	Common list parameters are supported. Additional parameters of an entity report are report specific and can be obtained from the report service itself.
Media types	application/json, application/xml
Result	A <a href="#">list model</a> (see page 53) 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
startIndex	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 -1 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>
<a href="#">fields</a> (see page 27)	no		String	<p>Comma-separated list of fields to be provided as result. Please see the <a href="#">REST Field Qualification</a>(see page 27) as well as <a href="#">REST Transition Fields</a>(see page 31) 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>
formatData	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-</i></p>

General list query parameters				
				<p><i>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 <a href="#">data type representation</a>(see page 0) section for details.</p>
<a href="#">includeLabels</a> (see page 25)	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>
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 <b>second</b> 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><b>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</b></p>



General list query parameters				
				<b>provided.</b>
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 not allowed to query all items of a Catalog "Henri" and specifying an assortment which is bound to the Mastercatalog. You will receive an appropriate error message in case the assortment does not fit.</p>
updateAssortment	no	false	Boolean	<p>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 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.</p>
cached	no	-	String	See <a href="#">Caching</a> (see page 52) for details
softDeleteMode	no	ALIVE_ONLY	String	<p>The mode for loading soft deleted objects. Possible values:  <b>ALIVE_ONLY</b> - Only include non-deleted objects in the result list. This is the default mode.</p> <p><b>DEAD_OR_ALIVE</b> - 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>

**General list query parameters**

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.

List header	
entityIdentifier	The entity of the HPM repository the list model contains entries of.
totalSize	The total number of results of the query.
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 <b>fields</b> 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.

Column	
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.
values	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

Entry														
Field	Description													
	<p>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 <i>id</i> and <i>label</i> 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 <i>id</i> and <i>label</i>.</p> <table><tr><th>Data type</th><th>Upper bound</th><th>JSON type</th></tr><tr><td>ENTITY_ITEM</td><td>&gt;1</td><td>Array of ProxyObjects (properties: <i>id</i> and <i>label</i>)</td></tr><tr><td>ENTITY_ITEM</td><td>&lt;=1</td><td>ProxyObject (properties: <i>id</i> and <i>label</i>)</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: <i>id</i> and <i>label</i> )	ENTITY_ITEM	<=1	ProxyObject (properties: <i>id</i> and <i>label</i> )	not ENTITY_ITEM		String (delimiter for multiple values is semi-colon)
Data type	Upper bound	JSON type												
ENTITY_ITEM	>1	Array of ProxyObjects (properties: <i>id</i> and <i>label</i> )												
ENTITY_ITEM	<=1	ProxyObject (properties: <i>id</i> and <i>label</i> )												
not ENTITY_ITEM		String (delimiter for multiple values is semi-colon)												

#### Examples

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
<pre>curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/list/Article/byCatalog?catalog=TOOLS</pre>

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.MainSupplier,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 32). 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.

#### 5.9.2.2 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 pageSize, startIndex, <a href="#">fields(see page 27)</a> and qualificationFilter 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	application/octet-stream
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 );
```

### 5.9.3 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 68)
  - [Examples](#)(see page 71)
- [Execute a specific report of sub entities for MIME values](#)(see page 80)
  - [Parameters](#)(see page 80)
  - [Examples](#)(see page 80)
  - [Rest Java Client Example](#)(see page 81)

#### 5.9.3.1 Execute a specific report for Read

URL Pattern	/list/{entity-identifier}/{sub-entity-identifier}/{report-name}
Method	GET
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.
Media types	application/json, application/xml
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
	startIndex	no	0	Integer	see <a href="#">REST List API Read for Root Entities</a> (see page 54)
	pageSize	no	100	Integer	see <a href="#">REST List API Read for Root Entities</a> (see page 54)
	<a href="#">fields</a> (see page 27)	no		String	see <a href="#">REST List API Read for Root Entities</a> (see page 54)
	metaData	no	false	Boolean	see <a href="#">REST List API Read for Root Entities</a> (see page 54)
	formatData	no	false	Boolean	see <a href="#">REST List API Read for Root Entities</a> (see page 54)
	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><i>n-ASC/DESC</i> 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 <b>second</b> 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><b>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</b></p>

General list query parameters					
					<p><b>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).</b></p> <p><b>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!</b></p> <p><b>If not necessary for the client to process the rows in those groups in a sorted way, he should not provide this parameter.</b></p>
	qualifi cation Filter	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: qualificationFilter=currency(EUR,USD),customer(Heiler)</p>
	fieldFil ter  (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 <a href="#">REST Datatypes</a>(see page 23) are supported for the parameters, except MIME_VALUE.</p> <p>Syntax: fieldFilter= [not] &lt;FieldIdentifier1&gt;([Parameter1], ..., [ParameterN]), [not] &lt;FieldIdentifierN&gt;([Parameter1], ..., [ParameterN])</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: fieldFilter=ArticleLang.Keyword("MyMarker")</p>

**General list query parameters**

					<p>Example for ArticleLang which has <b>no</b> "MyMarker" as keyword</p> <pre>fieldFilter=not ArticleLang.Keyword("MyMarker")</pre> <p>Example for ArticleLang which has <b>no</b> keyword</p> <pre>fieldFilter=ArticleLang.Keyword()</pre> <p>Example for ArticleLang which has <b>any</b> keyword</p> <pre>fieldFilter=not ArticleLang.Keyword()</pre> <p>Please also find more examples on this parameter in the ListAPI postman collection. See the Examples section of <a href="#">REST List API</a>(see page 46).</p>
	revision	no	-	ENTITY_ITEM	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.Valu
e&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 *byItem* 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 *byItem* 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 *byItem* 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 "MySpplierCatalog" 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.

### 5.9.3.2 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> , <a href="#">fields(see page 27)</a> 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	application/octet-stream
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\(see page 27\)](#) 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
<pre>curl -u rest:heiler -H "Accept: application/octet-stream" -X GET http://localhost:1512/rest/V2.0/list/Article/ArticleCharacteristicValue/mimes/byCatalog</pre>



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 ))
```

## 5.9.4 REST List API Read MIME files of Root and Sub Entities

The [REST List API Read MIME files of Sub Entities](#)(see page 83) 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 83)
  - [Execute a specific report for root entities](#)(see page 83)
  - [Parameters](#)(see page 84)
  - [Examples](#)(see page 85)
  - [Rest Java Client Example](#)(see page 85)
- [Sub Entity](#)(see page 86)
  - [Execute a specific report for sub root entities](#)(see page 86)
  - [Parameters](#)(see page 86)
  - [Additional parameters](#)(see page 86)
  - [Examples](#)(see page 87)
  - [Rest Java Client Example](#)(see page 88)

### 5.9.4.1 **Root Entity**

Execute a specific report for root entities

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 27) 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	application/octet-stream
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
startIndex	no	0	Integer	see <a href="#">REST List API Read for Root Entities</a> (see page 54)
pageSize	no	100	Integer	see <a href="#">REST List API Read for Root Entities</a> (see page 54)
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:  <code>qualificationFilter=currency(EUR,USD),customer(Heiler)</code></p>
<a href="#">fields</a> (see page 27)	no		String	<p>Comma-separated list of fields to be provided as result. Please see the <a href="#">REST Field Qualification</a>(see page 27) as well as <a href="#">REST Transition Fields</a>(see page 31) 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 );
```

### 5.9.4.2 **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> , <a href="#">fields(see page 27)</a> 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	application/octet-stream
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				
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.

 Parameter [fields\(see page 27\)](#) 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 ))
```

## 5.10 REST List API Write

- [REST List API Write for Root Entities](#)(see page 89)
- [REST List API Write for Sub Entities](#)(see page 101)

### 5.10.1 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 89)
  - [Parameters](#)(see page 90)
  - [Content](#)(see page 90)
  - [Result](#)(see page 91)
- [Examples](#)(see page 92)
  - [Setting non-formatted values](#)(see page 92)
  - [Setting formatted values](#)(see page 95)
  - [Setting MIMEValues for entity](#)(see page 97)

#### 5.10.1.1 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.
	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 58)) 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 23).
- 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 counters element			
	Field	Data type	Description
	errors	Integer	Total number of errors
	warnings	Integer	Total number of warnings
	createdObjects	Integer	Number of created objects
	updatedObjects	Integer	Number of updated objects
	objectsWithErrors	Integer	Number of objects with errors
	objectsWithWarnings	Integer	Number of objects with warnings
Properties of an entries element			
	Field	Data type	Description
	row	Integer	The index of the row causing this error or warning. The first row has the index 0.

	objectType	String	The type of the object (language dependent label is used). Might be empty.
	object	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.
<b>Properties of an objects element</b>			
	<b>Field</b>	<b>Data type</b>	<b>Description</b>
	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

### 5.10.1.2 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     "rows": [
14       {
15         "object":
16         {
17           "id": "'A1'@'MASTER'"
18         },
19         "values": [
20           "Heiler Software AG",
21           {
22             "id": "39"
23           },
24           "12.34"
25         ]
26       },
27       {
28         "object":
29         {
30           "id": "'A2'@'MASTER'"
31         },
32         "values": [
33           "Heiler Software AG",
34           {
35             "label": "fifteen kg drum"
36           },
37           "NotANumber"
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:

```

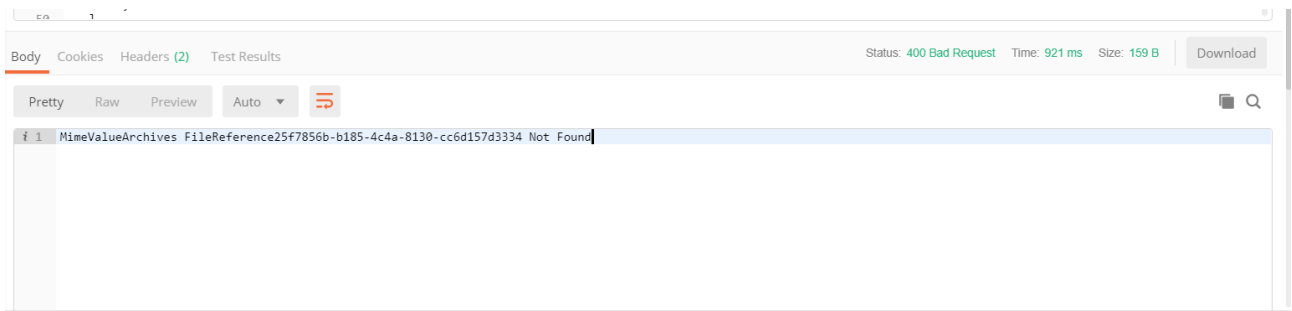
1  {
2      "columns": [
3          {
4              "identifier": "LookupValue.MIMEValue"
5          }
6      ],
7      "rows": [
8          {
```

```

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:



## 5.10.2 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 101)
  - [Parameters](#)(see page 102)
  - [Content](#)(see page 102)
  - [Result](#)(see page 102)
- [Examples](#)(see page 103)
  - [Setting values in sub entity ArticleLang](#)(see page 103)
  - [Setting values to subentity from root entity](#)(see page 105)
  - [Setting lookup value of an item](#)(see page 106)
  - [Setting MIMEValues for SubEntity](#)(see page 108)

### 5.10.2.1 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 58)) 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 23).
- 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 91)

### 5.10.2.2 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": "UnitSystemSpecific.Lang.Name(1,9)"
8      },
9      {
10       "identifier": "UnitSystemSpecific.Lang.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",

```

```

22         "Mein Name für die Einheit in diesem Einheitensystem"
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  }

```

```

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 }

```

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": {

```

```

30         "language": "German"
31     },
32     "values":
33     [
34         {
35             "label": "img1.jpg",
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": "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                     "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 "mimeValueArchives": [
59 {
60

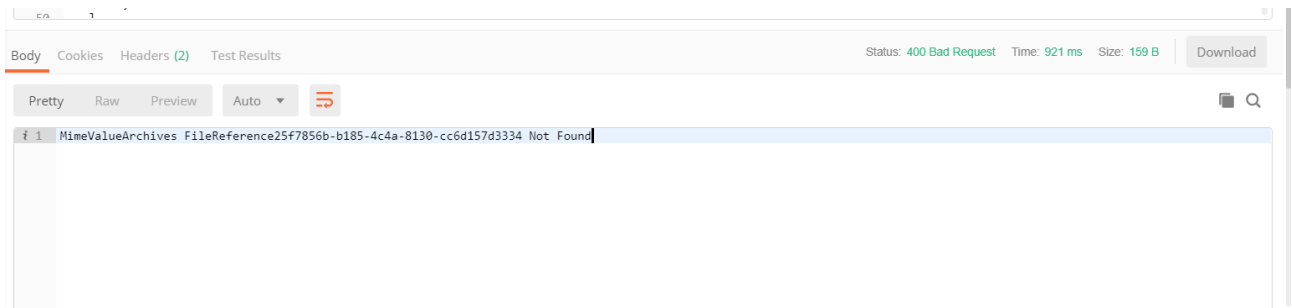
```

```

61      "id": "25f7856b-b185-4c4a-8130-cc6d157d3334",
62      "originalFilename": "uploadMime.zip"
63    }
64  ]
65 }

```

The response will look like:



## 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 112)
  - [Result](#)(see page 113)
- [Delete sub entity records of objects returned by a report](#)(see page 113)
  - [Result](#)(see page 114)
- [Examples](#)(see page 115)
  - [Delete all item in the supplier catalog TOOLS](#)(see page 115)
  - [Delete all sales prices with currency USD in the supplier catalog TOOLS](#)(see page 116)

### 5.11.1 Delete all objects returned by a report

URL Pattern	/list/{entity-identifier}/{report-name}
Method	DELETE
Parameters	Only the parameters of the specified entity report are available.
Content-Type	application/x-www-form-urlencoded
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.
--------	--

#### 5.11.1.1 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 91).

#### 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.

#### 5.11.2 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	/list/{entity-identifier}/{sub-entity-identifier}/{report-name}
Method	DELETE

Content-Type				application/x-www-form-urlencoded
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	Data type	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: qualificationFilter=currency(EUR,USD) ,customer(Heiler)</p>

#### 5.11.2.1 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 91).

#### **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!

## 5.11.3 Examples

### 5.11.3.1 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 );
```

### 5.11.3.2 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)" );
```

## 5.12 REST List API Errors

### 5.12.1 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

Occasion	Return code	Message
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"> <li>The response body will contain the error message of the original exception as plain text without stack trace in this case</li> <li>The stack trace of the exception is logged on the server</li> </ul>

### 5.12.2 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
Invalid values for report parameters	400 (Bad request)	<p>If paramter value could not be converted: "Value of given parameter X has a wrong format"</p> <p>If proxy does not exist: "Given item for parameter X does not exist"</p>
Missing report	404 (Not found)	Report '<ReportName>' does not exist.

### 5.12.3 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)	<p>Search Query syntax is incorrect</p> <p>Example:</p> <p>Unexpected character found at position 11</p>

Occasion	Return code	Message
		a = 1 AND b / 2 ^
No search available	404 (Resource not found)	

## 6 REST Object API

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 118)
  - [Single Item](#)(see page 118)
  - [Multiple Items](#) (since 10.1.0.02)(see page 119)
  - [Query Parameters](#)(see page 121)
  - [Permissions](#)(see page 122)
    - [Object Permissions](#)(see page 122)
    - [Field Permissions](#)(see page 122)
    - [Qualification Permissions \(aka Qualified Field Permissions\)](#)(see page 122)
  - [Result](#)(see page 123)
    - [Repository Entities and Fields](#)(see page 123)
      - [Names](#)(see page 123)
      - [Datatypes](#)(see page 123)
    - [Meta attributes](#)(see page 123)
    - [Data Element](#)(see page 123)
    - [Example Result](#)(see page 124)
    - [Characteristic Values](#)(see page 147)
      - [Additional Meta Attributes](#)(see page 147)
      - [Characteristic Records Example](#)(see page 148)
  - [Examples](#)(see page 153)

### 6.1 Read Requests

#### 6.1.1 Single Item

URL Pattern	/object/{entity-identifier}/{entity_item}
Method	GET

Accept	application/json application/xml										
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>400 (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	400 (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										
400 (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										

### 6.1.2 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			
	Header	Supported Values	Default
	Accept	application/ json application/xml	application/xml

	<table><tr><th>Header</th><th>Supported Values</th><th>Default</th></tr><tr><td>Content-Type</td><td>multipart/form-data</td><td></td></tr></table>	Header	Supported Values	Default	Content-Type	multipart/form-data			
	Header	Supported Values	Default						
Content-Type	multipart/form-data								
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><p>items=4711@1 items='myItem'@'myCatalog'</p><p>items=4711@1, 'myItem'@'myCatalog'</p></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>	<p>items=4711@1 items='myItem'@'myCatalog'</p> <p>items=4711@1, 'myItem'@'myCatalog'</p>
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>	<p>items=4711@1 items='myItem'@'myCatalog'</p> <p>items=4711@1, 'myItem'@'myCatalog'</p>						
Example	<p>/rest/V1.0/object/Article/byItems</p>								
Return Codes	<table><tr><th>Code</th><th>Reason</th></tr><tr><td>400 (OK)</td><td>Everything went fine</td></tr><tr><td>413 (Payload too large)</td><td><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></td></tr><tr><td>500</td><td>Internal server error, please check the server logs</td></tr></table>	Code	Reason	400 (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
Code	Reason								
400 (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								



### 6.1.3 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>
revision	no	root	ENTITY_ITEM	The revision for which the data should be retrieved	<p>revision='root'</p> <p>revision=1</p> <p>revision='myRevisionIdentifier'</p>

Parameter	Required	Default	Datatype	Parameter description	Example
					revision=4711 (where 4711 is the internal id of the revision).
includeLabels  (available with 10.1.0.02)	no	false	boolean	<p>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:</p> <pre> "orderUnit" : {   "_current" : {     "_key" : {       "_entityId" : 3100,       "_internalId" : "932",       "_externalId" : "'C62'"     },     "_code" : "C62",     "_label" : "piece"   } } </pre>	<p>includeLabels=true</p> <p>includeLabels=false</p>

## 6.1.4 Permissions

### 6.1.4.1 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.

### 6.1.4.2 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.

### 6.1.4.3 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.

## 6.1.5 Result

### 6.1.5.1 Repository Entities and Fields

#### Names

A new repository property "shortIdentifier" 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.

### 6.1.5.2 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 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.**

### 6.1.5.3 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.

#### 6.1.5.4 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": "h1r-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": "hlr-system/ajomgjomrxyjews.jpg"
    },
    "imageIdentifier": {
      "_current": "hlr-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"
            }
          ]
        }
      }
    ]
  }
}

```

#### 6.1.5.5 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": "\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": [
{
  "_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": "\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"  
      ]  
    }  
  }  
]  
}  
]  
}  
]  
}  
}
```

### 6.1.6 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='
```

## 7 REST Management API

The [.REST Management API v8.0](#)([see page 154](#)) 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 155](#)) — 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 155](#)) — The Rest Data Quality API allows to schedule data quality jobs.
- [REST Environment API](#)([see page 171](#))
- [REST Export API](#)([see page 178](#)) — The Rest Export API allows to schedule new export jobs and to cancel running export jobs.
- [REST Import API](#)([see page 187](#)) — The Rest Import API allows to schedule new import jobs and to cancel running import jobs.
- [REST KPI API](#)([see page 192](#)) — 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 200](#)) — The Rest Merge API allows to schedule merge jobs.
- [REST Revision API](#)([see page 210](#)) — 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 218](#))
- [REST Task API](#)([see page 223](#)) — The Rest Task API supports the creation, updating and content setting of PIM tasks.
- [REST Workflow API](#)([see page 233](#)) — The Rest Workflow API allows to manage workflows, workflow status and process status. This API is used for the Informatica BPM workflow engine integration.
- [REST File API](#)([see page 185](#)) — The Rest File API can be used for uploading files to the Product Manager Server, e.g. for providing the source data of an import.

## 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 155)

### 7.1.1 Get assortment content

URL Pattern	/manage/assortment/{assortment-id}/content
Method	GET
Parameters	updateAssortment: 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.

## 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 156)
  - [Content](#)(see page 156)
  - [Result](#)(see page 157)
  - [Examples](#)(see page 158)
- [Schedule Rule Execution](#)(see page 161)
  - [Query Parameters](#)(see page 162)
  - [Content](#)(see page 162)
  - [Result](#)(see page 163)
  - [Workflow Callback \(since 8.0.03\)](#)(see page 163)
  - [Workflow Callback \(since 10.0\)](#)(see page 164)
- [Examples](#)(see page 165)
  - [Executing a DQ checks for the catalog TOOLS](#)(see page 165)
- [Deprecated](#)(see page 167)
  - [Execute a data quality rule configuration group immediately](#)(see page 167)
  - [Execute a set of data quality rule configurations](#)(see page 168)

- [Execute channel's set of data quality rule configurations](#)(see page 169)
- [Examples](#)(see page 170)

### 7.2.1 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

#### 7.2.1.1 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!
ruleGroups	no		String Array	Optional parameter which specifies rule group 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
channels	no		ENTITY_ITEM	Optional parameter which specifies channels in form of an entity item array. Either rules, ruleGroups, channels 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.

#### 7.2.1.2 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
<b>items</b>		<b>An array of item objects which provide the results for all executed rules per item</b>
entityItem	ENTITY_ITEM	The reference to the entity item for which the rules have been executed
status	String	The overall status for all rules. Might be SUCCESSFUL, FAILED or UNKNOWN. SUCCESSFUL means that <b>all</b> rules have completed successfully FAILED means that <b>at least</b> a single rule

		<p>failed</p> <p>UNKNOWN means that for <b>at least</b> one rule no results are available. Usually this can only happen in case some unexpected error happens - please check the protocol in this case.</p>
failedRuleIds	Array of Integer	The id's of all rules which have failed for this item (see ruleIds above)
successfulRuleIds	Array of Integer	The id's of all rules which have succeeded for this item (see ruleIds above)
<b>protocol</b>		<b>The protocol (also known as problem log) of the execution.</b>
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
<b>entries</b>	<b>Array of protocol entries</b>	
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

### 7.2.1.3 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
      ]
    }
  ]
}

```



```

        "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"
        }
    ]
}
}
}

```

## 7.2.2 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

### 7.2.2.1 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

### 7.2.2.2 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,

Field	Required	Default	Datatype	Parameter description
				ruleGroups, channels 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 rules, ruleGroups, channels 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 rules, ruleGroups, channels 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.

### 7.2.2.3 Result

An object reference to the data quality job.

Properties of the returned object		
Field	Data type	Description
id	Integer	The job ID

### 7.2.2.4 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.

### 7.2.2.5 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 to 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>
        <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>

```

## 7.2.3 Examples

### 7.2.3.1 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'"
      }
    ]
  }
}
```

## 7.2.4 Deprecated

Please use /manage/dataquality/jobs instead of the following old URL's.

### 7.2.4.1 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.
	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

An object reference to the data quality job.

Properties of the returned object			
	Field	Data type	Description
	id	Integer	The job ID

### 7.2.4.2 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.

#### 7.2.4.3 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.

#### 7.2.4.4 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_descriptions",

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"
}
```

## 7.3 REST Environment API

### 7.3.1 REST Environment APIs used for Extraction, Download and Integration of Environment transfer

#### 7.3.2 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

##### 7.3.2.1 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
```

### 7.3.2.2 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>
```

### 7.3.3 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.

### 7.3.3.1 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
```

### 7.3.3.2 Result

A stream of the object to be downloaded is returned.

## 7.3.4 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

### 7.3.4.1 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

Parameter	Required	Default	Datatype	Parameter description
				for integration of the environment.

#### 7.3.4.2 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
```

#### 7.3.4.3

##### 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>

```

### 7.3.5 Checking the Progress of the Extraction/Integration Job

URL Pattern	/manage/environment/transfer/{jobId}
Method	GET
Content types	application/json, application/xml
Parameters	jobId received by triggering the extraction job
Returns	<p>jobId and currentState 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>

#### 7.3.5.1 URL Parameters

Parameter	Required	Default	Datatype	Parameter description
{jobId}	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
```

#### 7.3.5.2 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>
```

**7.3.5.3****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



State	Value	Discription
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
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
CANCELED_SHUTDOWN	canceled.shutdo wn	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

State	Value	Discription
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.

## 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 178)
  - [Query Parameters](#)(see page 179)
  - [Content](#)(see page 179)
    - [Property variables \(available with 8.0.02\)](#)(see page 180)
    - [Property dataSources \(available with 8.0.02\)](#)(see page 181)
  - [Result](#)(see page 182)
  - [Complete example](#)(see page 183)
- [Information about an export profile](#)(see page 183)
  - [Result](#)(see page 184)
  - [Example](#)(see page 184)
- [Cancel Export Job](#)(see page 184)
  - [URL Parameters](#)(see page 185)
  - [Result](#)(see page 185)

### 7.4.1 Schedule Export Job

Schedules an export.

URL Pattern	/manage/export
Method	POST
Content types	application/json, application/xml
Media types	application/json, application/xml
Result	The job object of the scheduled export job

## 7.4.1.1 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

## 7.4.1.2 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

Properties				
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 ( <i>deprecated</i> 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 23).

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
<code>identifier</code>	yes		String	Identifier of the report
<code>parameters</code>	no		Map<String,String>	Parameters
<code>assortmentFilter</code>	no		ENTITY_ITEM	Reference to an assortment (must be a reference object)
<code>updateAssortment</code>	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   }

```

```

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.

#### 7.4.1.3 Result

Field	Data type	Description
id	Integer	The ID of the export job

Field	Data type	Description
label	String	The label of the export job

#### 7.4.1.4 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     },
21     "Item list 2" :{
22       "reportQuery": {
23         "identifier": "bySearch",
24         "parameters": {
25           "query": "Article.SupplierAID startsWith \"A1\""
26         }
27       }
28     }
29   }

```

#### 7.4.2 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.
--------	---

#### 7.4.2.1 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

#### 7.4.2.2 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"
    }
  }
}
```

#### 7.4.3 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/{ <b>jobId</b> }/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.

#### 7.4.3.1 URL Parameters

Parameter	Required	Default	Datatype	Parameter description
{jobId}	yes		Integer	Unique ID of the job to cancel

#### 7.4.3.2 Result

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.

## 7.5 REST File API

The Rest File API can be used for uploading files to the Product Manager Server, e.g. for providing the source data of an import.

- [Uploading a File](#)(see page 185)
  - [Result](#)(see page 186)
- [Examples](#)(see page 186)
  - [Uploading a file with file name Data1.xls](#)(see page 186)

#### 7.5.1 Uploading a File

URL Pattern	/manage/file
Method	POST

Parameters	originalFilename
Content types	binary/octet-stream
Returns	a file object containing the ID of the upload folder and the file name

### 7.5.1.1 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).

## 7.5.2 Examples

### 7.5.2.1 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 );
```

## 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 187)
  - [Query Parameters](#)(see page 187)
  - [Content](#)(see page 188)
  - [Result](#)(see page 190)
- [Cancel Import Job](#)(see page 191)
  - [Result](#)(see page 191)
- [Examples](#)(see page 191)
  - [Scheduling an import for a certain date and time](#)(see page 191)
  - [Cancel a running import job](#)(see page 192)

### 7.6.1 Schedule Import Job

Schedules an import. The import files have to be uploaded first using the [REST File API](#)(see page 185).

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

#### 7.6.1.1 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

Parameter	Required	Default	Datatype	Parameter description
				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

#### 7.6.1.2 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	<a href="#">DATETIME</a> (see page 24)	Date and time when the import should be started
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> .

Properties				
Field	Required	Default	Datatype	Parameter description
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 <i>Article</i> (standard without customizing)				
Field	Required	Default	Datatype	Parameter description
parent	yes		String	The id of the catalog.
groupAssignmentMode	no	0	Integer	If set to <ul style="list-style-type: none"> <li>0: References to structure groups are added to existing structure groups</li> <li>1: References to structure groups replace existing structure groups</li> </ul>
Properties of the options element				
Field	Required	Default	Datatype	Parameter description
createNewObjects	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.

Properties of the options element				
Field	Required	Default	Datatype	Parameter description
updateExistingObjects	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.
dryRunBeforeImport	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.
dryRunOnly	no	false	Boolean	Only a dry run is executed, no objects are imported.
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 <code>true</code> , the import settings of the import mapping will be used, instead of the options. (since version 8.0.6.05)

### 7.6.1.3 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

## 7.6.2 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.

### 7.6.2.1 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.

## 7.6.3 Examples

### 7.6.3.1 Scheduling an import for a certain date and time

```
1 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
```

```
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 ]
14 }
```

```

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"
}

```

### 7.6.3.2 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"
}

```

## 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.



### 7.7.1 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

#### 7.7.1.1 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 <math>\geq 0</math>.</p>

### 7.7.1.2 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

### 7.7.1.3 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"
}
```

## 7.7.2 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	application/json, application/xml
Result	A Deletion Result object containing a Webservice Protocol of the execution.

### 7.7.2.1 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	<p>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.</p> <p>In case this parameter is not given, then all KPIs will be used that are contributed as KPI extension point.</p>
referenceTime	yes	-	DATETIME	Reference time that specifies the end time of the time interval that will be considered.
daysToBeBacktracked	yes	-	Integer	<p>Specifies the days to be backtracked ending at the Date from the referenceTime parameter.</p> <p>Must be convertible to an integer that is <math>\geq 0</math>.</p>

### 7.7.2.2 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
<b>protocol</b>		<b>The protocol (also known as problem log) of the execution.</b>
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

Properties of the returned object in the POST response		
<i>entries</i>	<i>Array of protocol entries</i>	
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

### 7.7.2.3 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"
    }
]
}

```

## 7.8 REST Merge API

The Rest Merge API allows to schedule merge jobs.

- [Schedule Merge Job](#)(see page 200)
  - [Query Parameters](#)(see page 200)
  - [Content](#)(see page 201)
  - [Result](#)(see page 208)
- [Workflow Callback](#)(see page 208)
  - [Workflow Callback \(since 10.0\)](#)(see page 208)
  - [Workflow Callback Example](#)(see page 208)
- [Examples](#)(see page 209)
  - [Schedule merge of the full "Apparel" catalog](#)(see page 209)

### 7.8.1 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

#### 7.8.1.1 Query Parameters

Parameter	Required	Default	Datatype	Parameter description
updateAssortment	false	false	Boolean	If updateAssortment is set, corresponding assortment will be updated (which can be time consuming) <b>before</b> performing merge.



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

### 7.8.1.2 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 entityReportQue		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

Field	Required	Default	Datatype	Parameter description
	ry is mandatory!			
assortment	no		ENTITY_ITEM	Optional assortment to be merged
entityReport Query	no		EntityReport Query	An entity report query which allows to use the available reports like 'byCatalog', 'byItems'... (Since 8.0.03)
entityReport QueryExecution	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	<a href="#">DATETIME</a> (see <a href="#">page 24</a> )	Date and time when the merge should be started.
defaultAction	no	"SUPPLEMENT"	String	<p>Default merge mode for all fields in case the corresponding supplier catalog has no persisted merge profile. "SUPPLEMENT", "OVERWRITE", "CLEAN_COPY", "NO_MERGE"</p> <p>This parameter doubles as the default action for the entityQualification parameter object.</p> <div>  <p>There is a priority order for the merge actions! Highest priority has the entityQualification parameter. In case this one is given, then this is how the merge is</p> </div>

Field	Required	Default	Datatype	Parameter description
				<p>going to be performed. If this parameter is <b>not</b> 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 defaultAction parameter. In case the entityQualification is omitted <b>and</b> there are no supplier catalog merge settings, then all fields and entities will be merged according to this defaultAction parameter. Finally, in case this one is also not there, a SUPPLEMENT merge is performed.</p>
entityQualification	no		EntityQualification	<p>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.</p>

#### 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 <b>Changed</b> English Value"	"My English Value"	"My English Value"
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 <b>Changed</b> English Value"	"My English Value"	"My English Value"
		Description (en)	"My <b>Changed</b> English Value"		"My <b>Changed</b> 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 <b>Changed</b> English Value"	"My English Value"	"My <b>Changed</b> 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 <b>Changed</b> English Value"	"My English Value"	"My <b>Changed</b> 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 it's 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
entity	String	the unique identifier of the entity / sub-entity. See the <a href="#">Meta API</a> (see page 298) for details on the possible entities.
defaultAction	String	one of the Merge Actions (NO_MERGE, SUPPLEMENT, OVERWRITE, CLEAN_COPY)
qualificationFilters	String[]	<p>With the qualificationFilters array you can limit this EntityQualification so it only applies for a subset (just a specific language for example)</p> <p>The qualificationFilters 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 <a href="#">Meta-API</a>(see page 298).</p> <p>Example for German language:  "qualificationFilters" : [ "language(DE)" ]  The syntax is quite similar to the <a href="#">List API</a>(see page 68) for sub-entities.</p>
fieldActions	FieldAction[]	Array of FieldAction elements. Each FieldAction holds the fieldIdentifier of a field and an fieldAction. In case a

Attribute	Datatype	Description
		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"
  }
]
}

```

### 7.8.1.3 Result

An object reference to the merge job.

Field	Data type	Description
id	Integer	The job ID
label	String	The label of the job

## 7.8.2 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 52)) .

### 7.8.2.1 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".

### 7.8.2.2 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>
<entry key="FILTERED_BY_DQ"/>
<entry key="FILTERED_BY_PRODUCT_FINDER"/>
</reportResults>
</jobFinished>

```

## 7.8.3 Examples

### 7.8.3.1 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"
      } ]
    } ]
  }
}

```

## 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 211)
  - [Query Parameters](#)(see page 211)
  - [Content](#)(see page 213)
  - [Result](#)(see page 216)
- [Get Revision Job Information](#)(see page 216)
  - [Result](#)(see page 217)

### 7.9.1 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 it's 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 it's 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.

#### 7.9.1.1 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

Parameter	Required	Default	Datatype	Parameter description
				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

### 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"`.

### 7.9.1.2 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 47).

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 <a href="#">List Info API</a> (see page 47).
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

Parameter	Required	Default	Datatype	Parameter description
				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 assortment parameter. If set to true 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 );

```

### 7.9.1.3 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"
}
```

## 7.9.2 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



## 7.9.2.1 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>

```

## 7.10 REST System API

### 7.10.1 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

### 7.10.2 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

### 7.10.3 Echo Text

Simple function to test server availability, authentication and correct charset/encoding use.

URL Pattern	/manage/system/echo/{echoText}
Method	GET
Parameters	echoText: the text string which get echoes back.
Content types	application/json
Returns	the given echoText with the server internal charset/encoding.

#### 7.10.3.1 Example:

Request:

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/manage/system/echo/helloWorld
```

Response:

```
helloWorld
```

### 7.10.4 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 21)). 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	grant_type: the type of token. "authorization_code" for an access token
Content types	application/json
Returns	A JSON Object, containing: access-token: the token expires-in: period of validity of the token (in seconds)

#### 7.10.4.1 Example:

Request:

```
POST /rest/V1.0/manage/system/security/token
grant_type: authorization_code
```

Response:

```
{
  "access-token":
  "4ABE78F118D7068322B9A1AEB5B76C385C32777D52FB971FF70593C28CE94C8DB43BD5E7EF4EAB84DA8A
  E435F231B5F274F3643A2B2527350861E38029A31D5CD826A28D68D70625E8D1F47...",
  "expires-in": 3600
}
```

```
}

```

### 7.10.5 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. "authorization_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 <a href="#">token (in seconds)</a></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.

#### 7.10.5.1 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
}
```

## 7.10.6 REST System API for License



Available since 7.0.04

### 7.10.6.1 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.

## 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 246)

The Rest Task API supports the creation, updating and content setting of PIM tasks.

- [Create Task](#)(see page 223)
  - [Content](#)(see page 223)
  - [Result](#)(see page 228)
- [Update Task](#)(see page 228)
  - [Content](#)(see page 229)
- [Get Task](#)(see page 229)
- [Delete Task](#)(see page 229)
- [Update Task Content](#)(see page 229)
  - [Content](#)(see page 230)
- [Get Task Content](#)(see page 230)
- [Workflow callback](#)(see page 230)
- [Information Page](#)(see page 231)
- [Examples](#)(see page 231)
  - [Create a task for all items in the catalog TOOLS](#)(see page 231)
  - [Create a task for supplier](#)(see page 232)
  - [Update a task: Change name and description](#)(see page 232)
  - [Update a task's content: Add another item](#)(see page 233)

### 7.11.1 Create Task

URL Pattern	/manage/task
Method	POST
Content types	application/json
Media types	application/json
Result	ENTITY_ITEM object of the new task.

#### 7.11.1.1 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	Data type	Read Only	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 contentMode.
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.	<b>Only if feature 'Language specific Task data' is disabled.</b> Mandatory.
description	String		Optional long description of the task.	<b>Only if feature 'Language specific Task data' is disabled.</b>
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.



Properties of the Task object				
Field	Data type	Read Only	Parameter description	Remarks
<b>userGroup</b>	ENTITY_ITEM		The user group the task will be assigned to.	<p>One of the users of the given user group has to accept the task and will be the assigned user then.</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. 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> <p>Error if <i>userGroup</i> specified does not exist.</p>
<b>supplier</b>	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.	
creation User	ENTITY_ITEM	yes	The user who created the task.	Set automatically
creation Date	DATE TIME (see	yes	The date/time the task has been created.	Set automatically

Properties of the Task object				
Field	Data type	Read Only	Parameter description	Remarks
	<a href="#">page 23</a> )			
escalationDate	DATE TIME (see <a href="#">page 23</a> )		The date/time the task escalates and the delegate is assigned replacing the originally assigned user.	
deadline	DATETIME (see <a href="#">page 23</a> )		The date/time the task is scheduled to be fulfilled. If deadline is reached, task will be assigned to responsible.	
finishEstimate	DATETIME (see <a href="#">page 23</a> )		The date/time the assigned user plans to fulfill the task (pure information).	
finishDate	DATETIME (see <a href="#">page 23</a> )		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).
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.

Properties of the Task object				
Field	Data type	Read Only	Parameter description	Remarks
accepted	Boolean	yes	Whether task has been accepted or not.	
acceptanceDate	<a href="#">DATE TIME</a> (see <a href="#">page 23</a> )	yes	The date/time when the task has been accepted by an assignee.	Set automatically when accepting a task.
workflow TaskId	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.
template	ENTITY_ITEM		UI template the task is bound to	For flexible task UI
workflow ServiceEndpoint	String		The endpoint which is to be triggered on Informatica BPM side when a workflow task is marked as finished.	
workflow CorrelationId	String		The correlation id of the workflow instance which is to be triggered on InfaBPM side when a workflow task is marked as finished.	
workflow CommunicationMode	QUEUE/REST		The communication mode which can define message queue or REST communication to BPM	
workflow QueueId	String		The queue id from the server.properties which is defined by the syntax "queue.[queueid].name"	

Properties of the Task object				
Field	Data type	ReadOnly	Parameter description	Remarks
			which is used for the response message	
taskLangs	Map<String,String>		The language specific data of a task.	<b>Only if feature 'Language specific Task data' is enabled</b>
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.	

#### 7.11.1.2 Result

An object reference of the new task.

Field	Data type	Description
id	Integer	The task id
label	String	The label of the task

#### 7.11.2 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 229) .

#### 7.11.2.1 Content

The content has to be a Task JSON object (see above).

#### 7.11.3 Get Task

URL Pattern	/manage/task/{task-id}
Method	GET
Media types	application/json
Result	Task object

#### 7.11.4 Delete Task

URL Pattern	/manage/task/{task-id}
Method	DELETE
Media types	application/json
Result	'true'/'false' string whether object could be deleted

#### 7.11.5 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.

#### 7.11.5.1 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.

#### 7.11.6 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 53) 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
```

#### 7.11.7 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.

### 7.11.8 Information Page

A short version of the API description is available at `/manage/task/info`.

### 7.11.9 Examples

#### 7.11.9.1 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'"}]
        }
}
```

**7.11.9.2 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'"}]
  }
}
```

**7.11.9.3 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"
}
```

## 7.11.9.4 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"
    }
  ]
}
```

## 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 234)
- [Get Workflow Details](#)(see page 238)
- [Enter Workflow Process Status](#)(see page 240)
- [Leave Workflow Process Status](#)(see page 242)
- [End Workflow Process](#)(see page 243)
- [Get Workflow Process Status of an entity item per process](#)(see page 244)
- [Add error message to the BPM perspective protocol view](#)(see page 245)

## 7.12.1 Create / Update Workflow Details

Creates / updates a workflow definition.

### 7.12.1.1 General Info

<b>URL Pattern</b>	/manage/workflow
<b>Method</b>	POST
<b>Parameters</b>	-
<b>Content types</b>	application/json
<b>Media types</b>	application/json
<b>Result</b>	Workflow creation / update  optional: Workflow status creation / update

### 7.12.1.2 Content

The content has to be a JSON object which includes the properties listed below.

#### Workflow

Field	Required	Default	Datatype	Parameter description	Example
<b>identifier</b>	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.	
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	1.0

Field	Required	Default	Datatype	Parameter description	Example
				cannot be modified. To add or delete new status entries e.g. you have to send the workflow data with a newer version!	

## Status

Field	Required	Default	Datatype	Parameter description	Example
<b>status</b>	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 <a href="#">REST Workflow Task</a> (see page 246) API for details	<pre>{   "container":     "1",   "containerEntityId":     "MasterCatalog",   "entity":     "Article" }</pre>
rejectDecision	no	n/a	WorkflowDecision	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":     [{</pre>

Field	Required	Default	Datatype	Parameter description	Example
					<pre> "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	
singleChoice	no	false	Boolean	Defines if the children workflow decisions are interpreted in the dialog as radio button group (true), or check boxes (false)	
id	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.	
decisions	no	n/a	ArrayList<WorkflowDecision>	Possibility of defining a hierarchical reject dialog, combining for example radio button groups with sub ordinary check boxes.	

Result

Properties

Field	Data type	Description
List	ResultSummary[]	<p>Counters indicating the number of created, updated and failed entities.</p> <p>NOTE: Successful Workflow status creations are reported as updated entities</p>

### 7.12.1.3 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'" }
        }
      ]
    }
  ]
}

```

### 7.12.2 Get Workflow Details

Get a workflow definition.

## 7.12.2.1 General Info

<b>URL Pattern</b>	/manage/workflow/{workflowIdentifier}
<b>Method</b>	GET
<b>Parameters</b>	-
<b>Content types</b>	application/json
<b>Media types</b>	application/json
<b>Result</b>	Workflow details Workflow status list

## 7.12.2.2 Parameters

**Workflow**

Field	Required	Default	Datatype	Parameter description	Example
<b>workflowIdentifier</b>	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
resolveWorkflowTasks	no	false	boolean	If set to true, the workflow task information is resolved, too	
includeObsoletes	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 }
  ]
}
```

### 7.12.3 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.

#### 7.12.3.1 General Info

<b>URL Pattern</b>	/manage/workflow/status/enter
<b>Method</b>	POST
<b>Parameters</b>	-
<b>Content types</b>	application/json
<b>Media types</b>	application/json
<b>Result</b>	success result

#### 7.12.3.2 Content

The content has to be a JSON object which includes the properties listed below.



Field	Required	Default	Datatype	Parameter description	Example
<b>processId</b>	yes	n/a	String	Unique identifier of the workflow instance. Within Informatica BPM workflow instances are called processes.	'1234567'
<b>workflowId</b>	yes	n/a	String	Unique identifier of the workflow itself	'workflowXYZ'
<b>status</b>	yes	n/a	String	A status name	'stateA', 'stateB'
<b>entity</b>	yes	n/a	String	The entity name	'Article', 'Variant' or 'Product2G'
<b>itemId</b>	yes	n/a	String	The itemids in the typical service API syntax	'1234@1'(see page 233)
resetAccept User	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'
hint	no	n/a	String	A comment field, which can be sent to the server, when this status has been entered	

### 7.12.3.3 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
}
```

## 7.12.4 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.

### 7.12.4.1 General Info

<b>URL Pattern</b>	/manage/workflow/status/leave
<b>Method</b>	POST
<b>Parameters</b>	-
<b>Content types</b>	application/json
<b>Media types</b>	application/json
<b>Result</b>	success result

### 7.12.4.2 Content

The content has to be a JSON object which includes the properties listed below.

Field	Required	Default	Datatype	Parameter description	Example
<b>processId</b>	yes	n/a	String	Unique identifier of the workflow instance	'1234567'
<b>status</b>	yes	n/a	String	A status name	'stateA', 'stateB'
<b>entity</b>	yes	n/a	String	The entity name	'Article', 'Variant' or 'Product2G'
<b>itemId</b>	yes	n/a	String	The itemids in the typical service API syntax	'1234@1'(see page 233)

Field	Required	Default	Datatype	Parameter description	Example
hint	no	n/a	String	A comment field, which can be sent to the server, when this status has been left	

#### 7.12.4.3 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
}
```

#### 7.12.5 End Workflow Process

Ends a workflow process and soft deletes all related status entries

##### 7.12.5.1 General Info

<b>URL Pattern</b>	/manage/workflow/process
<b>Method</b>	DELETE
<b>Parameters</b>	-
<b>Content types</b>	application/json

<b>Media types</b>	application/json
<b>Result</b>	-

#### 7.12.5.2 Parameters

Field	Required	Default	Datatype	Parameter description	Example
<b>processId</b>	yes	n/a	String	Unique identifier of the workflow instance	'1234567'

#### 7.12.5.3 Example

Request

<b>DELETE</b>
<a href="http://localhost:1512/rest/V1.0/manage/workflow/process?processId=9375">http://localhost:1512/rest/V1.0/manage/workflow/process?processId=9375</a>

### 7.12.6 Get Workflow Process Status of an entity item per process

Resolves information about the current status entries related to a given workflow process

#### 7.12.6.1 General Info

<b>URL Pattern</b>	/manage/workflow/status
<b>Method</b>	GET
<b>Parameters</b>	-
<b>Content types</b>	application/json
<b>Media types</b>	application/json
<b>Result</b>	success result and status information

### 7.12.6.2 Parameters

Field	Required	Default	Datatype	Parameter description	Example
<b>processId</b>	yes	n/a	String	Unique identifier of the workflow instance	'1234567'
<b>entity</b>	yes	n/a	String	The entity name	'Article', 'Variant' or 'Product2G'
<b>itemId</b>	yes	n/a	String	The itemids in the typical service API syntax	'1234@1'(see page 233)

### 7.12.6.3 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"
    }
  ]
}
```

### 7.12.7 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

### 7.12.7.1 General Info

<b>URL Pattern</b>	/manage/workflow/message
<b>Method</b>	PUT
<b>Parameters</b>	-
<b>Content types</b>	application/json
<b>Media types</b>	application/json
<b>Result</b>	HTTP 201

### 7.12.7.2 Parameters

Field	Required	Default	Datatype	Parameter description	Example
<b>message</b>	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

## 7.12.8 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 233). 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 246)
- [WorkflowTask](#)(see page 248)
- [Task Callback](#)(see page 249)
  - [Callback Payload for Finished state](#)(see page 250)
  - [Callback Payload when the process must be terminated](#)(see page 253)

### 7.12.8.1 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>

```

### 7.12.8.2 WorkflowTask

Field	Datatype	Description
<b>container</b>	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.
<b>entity</b>	String	Entity identifier of the task's content type.
<b>workflowServiceEndpoint</b>	String	The endpoint which is to be triggered on Informatica BPM side when a workflow task is marked as finished.
<b>workflowCommunicationMode</b>	QUEUE/REST	The communication mode which can define message queue or REST communication to BPM
<b>workflowQueueId</b>	String	The queue id from the server.properties which is defined by the syntax "queue.[queueid].name" which is used for the response message
<b>userGroup</b>	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>
<b>user</b>	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</p>



Field	Datatype	Description
		<p>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>
<b>supplier</b>	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>
template	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>
maximumDuration	Long	<p>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).</p>
name	String	<p>The name of the task. Ideally a meaningful short description of what is to do.</p>
description	String	<p>Optional long description of the task.</p>
priority	Integer	<p>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).</p>

### 7.12.8.3 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>
        </statusEntry>
      </processStatusInfo>
    </decisions>
  </rest:payload>
</rest:RESTRequest>
```

```

<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>
<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.
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.

The StatusEntry object		
Field	DataType	Description
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?)
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 correlationId of the workflow process
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

## 8 REST Enumeration API

The [.REST Enumeration API v8.1](#) (see page 254) provides read-only access to all enumerations of the Product Manager. The enum entry keys are used in field values as well as for the qualification of fields.

- [All Enumerations](#) (see page 254)
  - [Result](#) (see page 254)
- [Meta Data and Enumeration Entries](#) (see page 255)
  - [Result](#) (see page 255)
- [Examples](#) (see page 256)
  - [Retrieving all enumerations](#) (see page 256)
  - [Retrieving the entries of enumeration Enum.Languages](#) (see page 257)
  - [Retrieving the entries of enumeration Enum.SupplierWithMainSupplier](#) (see page 258)

### 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

#### 8.1.1 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 <a href="#">REST Datatypes</a> (see page 23).
name	String	Localized name of the enumeration
description	String	Localized description of the enumeration

## 8.2 Meta Data and Enumeration Entries

Returns the entries of the specified enumeration.

URL Pattern	/enum/{enumeration-identifier}
Method	GET
Parameters	No parameters are available
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

### 8.2.1 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 <a href="#">REST Datatypes</a> (see page 23).
	entries	Array	List of all entries of this enumeration

Each enumeration entry has the following properties:

Properties of a member of entries			
	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 used by the HPM export module. Note that the external code is not unique!
	keySynonyms	Array of Strings	List of registered synonyms

## 8.3 Examples

### 8.3.1 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)
{
    // ...
}
```

### 8.3.2 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();
```

**8.3.3 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 }

```

## 9 REST Media API

- i** The REST Media API provides access to all kinds of media assets in the Product Manager.
- [Single Media Asset File](#)(see page 259)
  - [Preview of Media Asset File](#)(see page 259)
  - [Media Asset File Location](#)(see page 260)
  - [Upload single Media Asset File](#)(see page 260)
  - [Examples](#)(see page 261)
    - [Retrieve single media asset file in 36dpi JPEG quality with identifier D123456](#)(see page 261)
    - [Retrieve media asset file preview](#)(see page 262)
    - [Retrieve media asset file location](#)(see page 262)
    - [Upload media asset file to media asset server](#)(see page 262)

### 9.1 Single Media Asset File

URL Pattern	/media/{quality/derivativeDefinition}/{identifier}	
Method	GET	
Parameters	lastModified	DATETIME
Content types	application/octet-stream	
Returns	the media asset file as binary data stream	

### 9.2 Preview of Media Asset File

URL Pattern	/media/{identifier}/preview	
Method	GET	
Parameters	size	All allowed values: small   medium   large
Content types	image/jpg	

Returns	the media asset preview as binary jpeg	
---------	--	--

### 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	

### 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 185) 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.

## 9.5 Examples

### 9.5.1 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 );
```

### 9.5.2 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 );
```

### 9.5.3 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");
```

### 9.5.4 Upload media asset file to media asset server

call [REST File API](#)(see page 185) 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

```
MediaRequest mediaRequest = restClient.createMediaRequest();

String mediaAssetIdentifier = mediaRequest.addMediaAsset( "feb3cd45-  
d27e-4700-912e-26014512a6e3",
```

[illegible]

## 9.6 REST List API for MediaAssetFile entity

 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 264)
  - **Internal id**(see page 264)
  - **Special reports**(see page 265)
    - **byCategory**(see page 265)
    - **bySearch**(see page 265)
      - **Search with query**(see page 266)
      - **Search with mediaQuery**(see page 266)
- **Examples**(see page 266)
  - **Retrieve media asset file with "byIdentifiers" report for external id**(see page 266)
  - **Retrieve media asset file with "byItems" report for internal id**(see page 267)
  - **Retrieve media asset files which belong (be assigned) to a category and are used in PIM CORE**(see page 268)
  - **Retrieve all media asset files which belong to a category or its child categories**(see page 269)
  - **Update media asset file (Root Entity)**(see page 270)
  - **Update media asset file language special attribute (Sub Entity)**(see page 271)
  - **Delete media asset file with identifier D11000100112532**(see page 273)
  - **Create media asset file**(see page 273)

### 9.6.1 Adjustments for MediaAssetFile entity

The REST List API access for general object is already described in [REST List API](#)(see page 46), which is also available for the `MediaAssetFile` entity with some adjustments.

#### 9.6.1.1 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.




## 9.6.1.2 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 278).

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 273).

## 9.6.2 Examples

### 9.6.2.1 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"
  ]
}
]
}

```

### 9.6.2.2 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"
      ]
    }
  ]
}

```

### 9.6.2.3 Retrieve media asset files which belong (be assigned) to a category and are used in PIM core

## Rest Call

[illegible]

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"
      ]
    }
  ]
}
```

```

    ],
    },
    ...
  ]
}

```

#### 9.6.2.4 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"
      ]
    }
  ]
  ...
}

```

### 9.6.2.5 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"
      ]
    }
  ]
}
```

#### 9.6.2.6 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"
    ]
  }
]
}

```

#### 9.6.2.7 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
```

#### 9.6.2.8 Create media asset file

Please visit the page [Upload single Media Asset File](#).

### 9.6.3 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 274)
- [Examples](#)(see page 277)
  - [Search all media asset files whose filename contain "test"](#)(see page 277)

### 9.6.3.1 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
F_ATTRIBUT.ATTR_NAME	is/contain/begin/empty	Character	My custom defined state (If 'empty' operator is used no	State

FIELD	OPERATOR	CONTENT	Example search content	Corresponding PIM Desktop search fieldname (en)
			search value is needed)	
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
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

FIELD	OPERATOR	CONTENT	Example search content	Corresponding PIM Desktop search fieldname (en)
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
F_IMGKOMP.PIMG_PREVIEW_STATE	equal/less	4 values	Special case only this 4 combinations are allowed:  No preview: OPERATOR 'less' with '5'  No read only preview: OPERATOR 'less' with '9'  Error creating preview: OPERATOR 'equal' with '0 or 6'  Preview generation queued: OPERATOR 'equal' with CONTENT '3 or 4 or 7 or 8'	Preview version

FIELD	OPERATOR	CONTENT	Example search content	Corresponding PIM Desktop search fieldname (en)
NUMBER_OF_GROUP_ASSIGNMENT	equal/greater	Number	1	Number of group assignments
F_IMGITEM.IMI_ITEM_Mxx (text type property field)	is/contain/begin	Character	My search value	Itemfield type text
F_IMGITEM.IMI_ITEM_Mxx (list type property field)	is	List values	My selected list value	Itemfield type list
F_IMGITEM.IMI_ITEM_Mxx (bool type property field)	is	True/False	true	Itemfield type bool
F_IMGITEM.IMI_ITEM_Mxx (date type property field)	equal/less/greater	Date	09.08.2013	Itemfield type date
F_IMGITEM.IMI_ITEM_Mxx (int type property field)	equal/less/greater/lessequal/greaterequal	Number	4	Itemfield type int
F_IMGITEM.IMI_ITEM_Mxx (real type property field)	equal/less/greater/lessequal/greaterequal	Number	3.5	Itemfield type real
F_IMGITEM.IMI_ITEM_Mxx (multiselect list type property field)	contain	Listvalues	My selected list value	Itemfield type multiselect list

### 9.6.3.2 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"
      ]
    },
    ...
  ]
}
```

### 9.6.4 Search query for MediaAssetFile entity

 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 279)
- [Examples](#)(see page 283)
  - [Retrieve all media asset files whose filename contain "test"](#)(see page 283)
  - [Retrieve all media asset files which belong to a category or its child categories](#)(see page 283)
  - [Retrieve all media asset files whose version is greater or equals 2](#)(see page 285)
  - [Retrieve all media asset files who are changed on the 2015-11-12 or later](#)(see page 286)
  - [Retrieve all media asset files whose medias type equals "images"](#)(see page 287)
  - [Retrieve all media asset files whose english memo contains "test"](#)(see page 288)
  - [Retrieve all media asset files whose english property field\(of MultiSelectList type\) contains "First"](#)(see page 289)

- Retrieve all media asset files whose name start with "test" or whose english memo equals "test"(see page 290)

#### 9.6.4.1 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 32).

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"> <li>• exact category id (<b>120-digit-STRING</b> )</li> <li>• id for category and all sub categories (STRING with suffix %)</li> </ul>	<ul style="list-style-type: none"> <li>• "009900010000..."</li> <li>• "0099%"</li> </ul>	

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.
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"> <li>• docus</li> <li>• images</li> </ul>	images	The allowed search contents are the labels of the Media Manager Enum <b>MediaDataTypes</b> whose id and the detailed



ENTITY FIELD	OPERATOR	CONTENT	Example search content	Notice
		<ul style="list-style-type: none"> <li>• videos</li> <li>• sounds</li> <li>• fonts</li> <li>• internet</li> <li>• profiles</li> <li>• common</li> <li>• drawings</li> </ul>		<p>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 <b>MediaDataTypes</b> enum entry, "JPEG" and "Graphics Interchange Format" are detailed medias type names.</p>
MediaAssetFileAttribute.State	=, equals, is empty	entry of Enum.MediaAssetFileState	"Job is finished"  2	
MediaAssetFileAttribute.Status	=, equals	entry of Enum.MediaAssetFileStatus	Free	
MediaAssetFileAttribute.Finished	=, equals	True False	True	
MediaAssetFileAttribute.InProgress	=, equals	True False	False	
MediaAssetFileAttribute.Lang.Name	=, != / <>, equals(IC), contains(IC), startsWith(IC), in, is empty	Name value (STRING)	nameTest	
MediaAssetFileAttribute.Lang.Memo	=, != / <>, equals(IC), contains(IC),	Name value (STRING)	memoTest	

ENTITY FIELD	OPERATOR	CONTENT	Example search content	Notice
	startsWith(IC), is empty			
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	
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	

### 9.6.4.2 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:

[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.LastModified
```

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 (MediaAssetFileAttributeLang.Memo(en) =
test )&fields=MediaAssetFile.Identifier,
MediaAssetFileAttribute.Filename,MediaAssetFileAttributeLang.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"
    ]
  },
  ...
]
}

```

## 9.7 REST Media Asset Category API

**i** 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 291)
- [Get top Media Asset Categories](#)(see page 292)
- [Get direct child Media Asset Categories](#)(see page 292)
- [Add Media Asset Category](#)(see page 292)
- [Update Media Asset Category](#)(see page 293)
- [Delete Media Asset Category](#)(see page 293)
- [Examples](#)(see page 293)
  - [Retrieve single media asset category with identifier](#)(see page 293)
  - [Retrieve all top media asset categories](#)(see page 294)
  - [Retrieve all direct child media asset categories of the category with identifier](#)(see page 295)
  - [Add a top media asset category](#)(see page 295)
  - [Add a child media asset category under the category with identifier](#)(see page 296)
  - [Update the name of the media asset category with identifier](#)(see page 297)
  - [Delete a media asset category with identifier](#)(see page 298)

### 9.7.1 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

### 9.7.2 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

### 9.7.3 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

### 9.7.4 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 an new top category will be created, otherwise a child category will be created under the category with parentId.

### 9.7.5 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

### 9.7.6 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

### 9.7.7 Examples

#### 9.7.7.1 Retrieve single media asset category with identifier

## Rest Call

[illegible]

An object including an ID and an attributes which contains German name is returned:

[illegible]

```

    "attributes": {
      "name": "category_german",
      "completeCategoryPath": "/category_german",
      "level": 1,
      "hasChildren": true
    }
  }
}

```

## Rest Client Java Code

[illegible]

#### 9.7.7.2 Retrieve all top media asset categories

## Rest Call

```
curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1501/rest/V1.0/media/categories
```

The following JSON object is returned:

[illegible]

```

    "level": 1,
    "hasChildren": false
  }
}
]
```

## Rest Client Java Code

```
MediaAssetCategoryRequest mediaAssetCategoryRequest =
restClient.createMediaRequest().createCategoryRequest();
List< MediaAssetCategory > topCategories =
mediaAssetCategoryRequest.getCategories( );
```

#### 9.7.7.3 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/  
000100000000000000000000000000000000000000000000000000000000000000000000000000000000000  
00000000000000000000000000000000/categories
```

## Rest Client Java Code

[illegible]

#### 9.7.7.4 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 );
```

#### 9.7.7.5 Add a child media asset category under the category with identifier

## Rest Call

[illegible]

## Rest Client Java Code

```
MediaAssetCategoryRequest mediaAssetCategoryRequest =
restClient.createMediaRequest().createCategoryRequest();
```



[illegible]

#### 9.7.7.6 Update the name of the media asset category with identifier

## Rest Call

[illegible]

The following JSON object is provided as content:

```
{
  "name": "newCategoryName"
}
```

The following JSON object is returned:

[illegible]

## Rest Client Java Code

```
MediaAssetCategoryRequest mediaAssetCategoryRequest =  
restClient.createMediaRequest().createCategoryRequest();
```

[illegible]

#### 9.7.7.7 Delete a media asset category with identifier

## Rest Call

[illegible]

## Rest Client Java Code

[illegible]

## 10 REST Meta API

The [R\(see page 298\)EST Meta API\(see page 298\)](#) 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 299)
  - **Result**(see page 299)
- **Metadata for a Root Entity**(see page 300)
  - **Parameters**(see page 300)
  - **Result**(see page 300)
- **All Fields of a Root Entity**(see page 301)
  - **Result**(see page 301)
- **Metadata for a Sub Entity**(see page 302)

- [Parameters](#)(see page 302)
- [Result](#)(see page 302)
- [Metadata for a Field](#)(see page 303)
  - [Parameters](#)(see page 303)
  - [Result](#)(see page 303)
- [Metadata for a Qualifier](#)(see page 305)
  - [Result](#)(see page 305)
- [Examples](#)(see page 306)
  - [Retrieving all root entities](#)(see page 306)
  - [Retrieving meta data of the root entity Article](#)(see page 306)
  - [Retrieving all fields of the root entity Article](#)(see page 308)
  - [Retrieving meta data of the sub entity ArticlePriceValuePurchase](#)(see page 308)
  - [Retrieving metadata for the field ArticleLang.DescriptionLong](#)(see page 309)
  - [Retrieving meta data of the field ArticlePriceVa.DescriptionLong with qualification](#)(see page 310)
  - [Retrieving meta data of the qualifier Language for entity ArticleLang](#)(see page 311)

## 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

### 10.1.1 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

## 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.

### 10.2.1 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.

### 10.2.2 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 <a href="#">qualifier properties</a> (see page 305).
	fields	Array	List of direct fields. The available properties for each field are described at <a href="#">field properties</a> (see page 303).
	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.

## 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.

### 10.3.1 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 <code>fields</code> 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

## 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.

### 10.4.1 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.

### 10.4.2 Result

The returned object has the same properties as the result object for a root entity. See [Properties of a root entity\(see page 300\)](#).

## 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.

### 10.5.1 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 <a href="#">REST Field Qualification</a> (see page 27) 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.

### 10.5.2 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
	qualified Name	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 <a href="#">here(see page 0)</a>
	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 <a href="#">String(see page 0)</a> .
	maxLength	Integer	Defines the maximum length. Only meaningful if the data type is <a href="#">String(see page 0)</a> .
	richText	Boolean	Defines if this field can hold formatting markers. Only meaningful if the data type is <a href="#">String(see page 0)</a> .
	scale	Integer	Defines the number of digits after the decimal separator. Only meaningful if the data type is <a href="#">Decimal(see page 0)</a> .
	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 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 <a href="#">String</a> (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.

## 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

### 10.6.1 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 <a href="#">REST List API Write for Sub Entities</a> (see page 101).
description	String	Localized description of this field.
dataType	String	Data type of this qualifier. The possible data types are limited and documented <a href="#">here</a> (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)

## 10.7 Examples

### 10.7.1 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 }
```

### 10.7.2 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 }

```

### 10.7.3 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     ]
21 }
```

### 10.7.4 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 }

```

### 10.7.5 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
item has been recorded",
25         "dataType": "INTEGER",
26         "enumeration": {
27             "name": "All languages",
28             "identifier": "Enum.Language"
29         }
30     }
31 ]
32 }

```

### 10.7.6 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 }

```

### 10.7.7 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 }

```

## 10.8 REST Meta Media API

e

 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 312)
  - [Get single derivative definition](#)(see page 312)
    - [Content](#)(see page 312)
  - [Get all derivative definitions](#)(see page 313)
- [Media Asset File Property field definition](#)(see page 313)
  - [Get single property field definition for one supported language](#)(see page 313)
    - [Content](#)(see page 314)
  - [Get single property field definition for all supported languages](#)(see page 316)
  - [Get all property field definitions for all supported languages](#)(see page 317)
- [Examples](#)(see page 317)
  - [Retrieve single derivative definition with identifier](#)(see page 317)
    - [REST call for JSON](#)(see page 317)
    - [REST call for XML](#)(see page 318)

- [REST Client Java](#)(see page 318)
- [Retrieve all derivative definitions](#)(see page 318)
  - [REST call for JSON](#)(see page 318)
  - [REST call for XML](#)(see page 319)
  - [REST Client Java](#)(see page 320)
- [Retrieve single property field definition for one supported language](#)(see page 320)
  - [REST call for JSON](#)(see page 320)
  - [REST call for XML](#)(see page 321)
  - [REST Client Java](#)(see page 322)
- [Retrieve single property field definition for all supported languages](#)(see page 322)
  - [REST call for JSON](#)(see page 322)
  - [REST call for XML](#)(see page 323)
  - [REST Client Java](#)(see page 323)
- [Retrieve property field definitions for all supported languages](#)(see page 324)
  - [REST call for JSON](#)(see page 324)
  - [REST call for XML](#)(see page 325)
  - [REST Client Java](#)(see page 325)

### 10.8.1 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 259).



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).

#### 10.8.1.1 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
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

#### 10.8.1.2 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

### 10.8.2 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.

#### 10.8.2.1 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)
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

Field	Required	Datatype	Parameter description	Example
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 definitoin 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"
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"]
regularExpressi on	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 filed defintion type to the corresponding Product 360 DataType inclusive additional notes

Property field definiton type in Media Manager	DataType in Product 360	additional notes
Boolean	BOOLEAN	
Date	DATE	
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]

#### 10.8.2.2 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
--------	--

### 10.8.2.3 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

## 10.8.3 Examples

### 10.8.3.1 Retrieve single derivative definition with identifier

REST call for JSON

Rest Call for JSON
<pre>curl -u rest:heiler -H "Accept: application/json" -X GET http://localhost:1512/rest/V1.0/meta/media/derivativeDefinitions/1</pre>

A MediaAssetDerivativeDefinition JSON object including id and other attributes of the desired derivative definition is returned:

Returned JSON object
<pre>{   "id": 1,   "shortDescription": "JPEG 36dpi",   "longDescription": "Derivative with JPEG extension and 36 dpi resolution",   "volumeId": 0 }</pre>

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 );
```

### 10.8.3.2 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();

```

## 10.8.3.3 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 selelction 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" );
```

## 10.8.3.4 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 selection 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();
```

### 10.8.3.5 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 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_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();
```

## 11 REST Security API

The REST Security API provides access to securities of the Product Manager.

- [List all Security ACL APIs](#)(see page 326)
  - [Result](#)(see page 326)
- [Read ACL object](#)(see page 327)
  - [Content](#)(see page 327)
- [Create ACL object](#)(see page 328)
- [Examples](#)(see page 330)
  - [Read ACL object for an id](#)(see page 330)
    - [REST call for JSON](#)(see page 330)
    - [REST call for XML](#)(see page 332)
    - [REST Client Java](#)(see page 333)
  - [Create ACL object](#)(see page 333)
    - [REST call for JSON](#)(see page 333)
    - [REST call for XML](#)(see page 334)
    - [REST Client Java](#)(see page 335)

### 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

#### 11.1.1 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	-

Description	URL Pattern	Method	Parameters	Result	Content
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

## 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

### 11.2.1 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
<b>Principal properties</b>			
	<b>Field</b>	<b>Data type</b>	<b>Description</b>
	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

### 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



Flag "deleted" of principal in content	Principal is deleted on the system	Result
		EntityItemReference does not exist.
true	true (principal deleted)	AclEntry with that principal is <b>NOT</b> 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
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

fullPermission	deletePermission	writePermission	readPermission	adjusted permissions of created object
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.

## 11.4 Examples

### 11.4.1 Read ACL object for an id

#### 11.4.1.1 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
  }
]
}

```

## 11.4.1.2 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>
  </entries>
</acl>
```

```

    <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>

```

#### 11.4.1.3 REST Client Java

##### Rest Client Java Code

```

SecurityRequest securityRequest = getRestClient().createSecurityRequest();
Long actualAclId = 3L;
ACL actualAcl = securityRequest.getAcl( actualAclId );

```

#### 11.4.2 Create ACL object

##### 11.4.2.1 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
    }
  ]
}

```

#### 11.4.2.2 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>
      <writePermission>true</writePermission>
      <readPermission>true</readPermission>
    </entry>
  </entries>
</acl>

```

#### 11.4.2.3 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()

```

## 12 REST API Tutorial

### 12.1 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*.

### 12.2 Tools

Postman for Google Chrome as REST client ( <http://www.getpostman.com/> )

### 12.3 Recording

The recording (one hour long) is available at

<https://informaticameeting.webex.com/informaticameeting/lr.php?RCID=161c81bc0d1743fbaff6fc2cafec1d1>

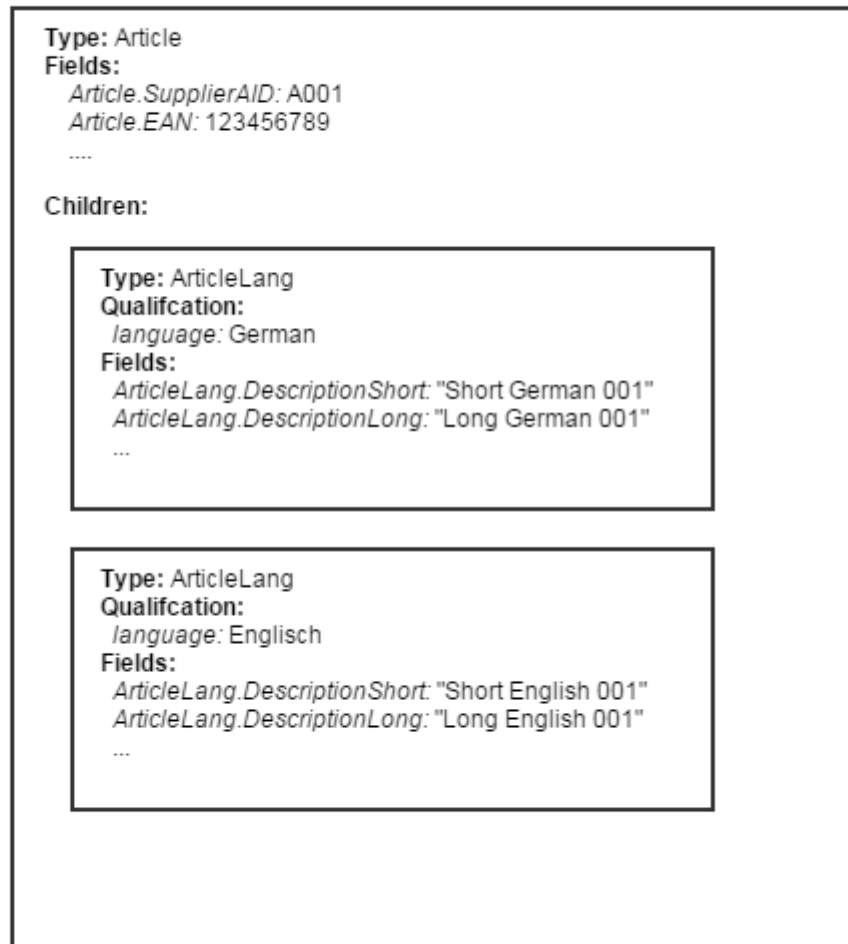
### 12.4 REST API Tutorial - Data Model

#### 12.4.1 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 23)

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>

## 12.5 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:

### 12.5.1 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 32)).

General information about reports: Entity Reporting

### 12.5.2 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 27)

### 12.5.3 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 53)

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*).

### 12.5.4 Examples

#### 12.5.4.1 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
```

#### 12.5.4.2 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,&nbsp;ArticleLang.DescriptionShort(German),ArticleLang.DescriptionShort(English),ArticleAttributeValue.Value(Length,"Language independent",DEFAULT),ArticleAttributeValue.Value("Front Color",English,DEFAULT)&metaData=true
```

**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)

#### 12.5.5 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 31).

**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
```

## 12.5.6 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&pageSize=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&pageSize=1&cacheId=<Returned cacheId>
```

## 12.6 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.

### 12.6.1 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.

#### 12.6.1.1 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 23)).

```
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
8        Color\",English,DEFAULT)"
9      }
10     ],
11     "rows": [
12       {
13         "object": {
14           "id": "'A001'@'RestTutorial'"
15         },
16         "values": [
17           "Updated Short English A001",
18           "light blue"
19         ]
20       },
21       {
22         "object": {
23           "id": "'A002'@'RestTutorial'"
24         },
25         "values": [
26           null,
27           "red"
28         ]
29       }
30     ]
31   }

```

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.

#### 12.6.1.2 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 }

```

## 12.6.2 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 23).

### 12.6.2.1 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   }

```

## 12.7 REST API Tutorial - List Read for Sub Entities

### 12.7.1 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

### 12.7.2 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.

#### 12.7.2.1 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
&metaData=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
```

### 12.7.3 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*

#### 12.7.3.1 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(English)&
metaData=true&pageSize=-1
```

## 12.8 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

#### 12.8.1 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 REST API How To's

- [Updating Product's Structure Group with Rest API](#)(see page 346)
- [Assigning User Groups to a User](#)(see page 347)
- [Monitor all REST requests in the log file](#)(see page 347)
- [Query the list of items assigned to multiple structure groups of a structure](#)(see page 348)

### 13.1 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": "MGRTN0000003381699"
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.

## 13.2 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      }]
    }
```

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.

## 13.3 Monitor all REST requests in the log file

Add the following lines to the log4j.xml:

```

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

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.AcIProxy,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

```

## 13.4 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

```

## 13.5 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 46) before you read this how-to.

### 13.5.1 Definitions

Name	
Characteristic	The model definition. Name, Datatype, Lookup Values, etc.
CharacteristicRecord	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.
ArticleCharacteristicValue, Product2GCharacteristicValue, VariantCharacteristicValue	The repository entity for a characteristic record of the corresponding root entity
SimpleArticleCharacteristicValue, SimpleProduct2GCharacteristicValue, SimpleVariantCharacteristicValue	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.

### 13.5.2 Qualifications

Qualification	Datatype	
characteristic	ENTITY_ITEM	The characteristic to which this value belongs as <a href="#">ENTITY_ITEM</a> (see page 23)
rootCharacteristic	ENTITY_ITEM	The root characteristic (not necessarily the parent) of the characteristic as <a href="#">ENTITY_ITEM</a> (see page 23). <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.

### 13.5.3 Datatypes

Characteristic Datatype	Corresponding Rest Datatype
TEXT	<a href="#">STRING</a> (see page 23)
INTEGER	<a href="#">INTEGER</a> (see page 23)
DECIMAL	<a href="#">DECIMAL</a> (see page 23)
BOOLEAN	<a href="#">BOOLEAN</a> (see page 25)
DATE	<a href="#">DATE</a> (see page 24)
DATETIME	<a href="#">DATETIME</a> (see page 24)

Characteristic Datatype	Corresponding Rest Datatype
MIME	<a href="#">MIME_VALUE</a> (see page 26)
LOOKUP	<a href="#">ENTITY_ITEM</a> (see page 25)
NONE	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.

### 13.5.4 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.

#### 13.5.4.1 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.

### 13.5.5 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.

#### 13.5.5.1 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),SimpleArti
cleCharacteristicValueLang.Value(STATEMENT,en),SimpleArticleCharacteristicValueLang.V
alue(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=SimpleProduct2GCharacteristicValueLang.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\\
\\logo.908f8b16002fbfb2_4955b4f5_170d4d6aaae_-7e4a.png"
23          }
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\\
\\logo_global.908f8b16002fbfb2_4955b4f5_170d4d6aaae_-7e3b.png"
39          }
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)&includ
eLabels=true

```

As result a zip file containing the MIME files will be provided

#### 13.5.5.2 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
22 [TextRestTestCharacteristic]",
23                     "entityId": 8000
24                 },
25                 "parentRecordKey": "root",
26                 "language": "-1",
27                 "characteristic": {
28                     "id": "10759",
29                     "label": "TextRestTestCharacteristic
30 [TextRestTestCharacteristic]",
31                     "entityId": 8000
32                 }
33             },
34             "values": [
35                 "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
22 [TextRestTestCharacteristic]",
23                     "entityId": 8000
24                 },
25                 "parentRecordKey": "root",
26                 "language": "-1",
27                 "characteristic": {
28                     "id": "10759",
29                     "label": "TextRestTestCharacteristic
30 [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     "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
t2GCharacteristicValue.Lang.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         }
66     ]
67 }

```

```

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     }
38   ]
39 }
```

```

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
27 [marketingInformation]",
28                     "entityId": 8000
29                 },
30                 "parentRecordKey": "0000.0000.RK",
31                 "language": "9",
32                 "characteristic": {
33                     "id": "39",
34                     "label": "Key words [tradeItemKeyWords]",
35                     "entityId": 8000
36                 }
37             },
38             "values": [
39                 "0000.0000.RK",
40                 [
41                     "Item versioned with variant and product in
42 ProductVersion2"
43                 ]
44             ]
45         },
46         {
47             "object": {
48                 "id": "95@1",
49                 "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         "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":
"880x495_cmsv2_298e3b01-877d-57e3-9ce0-0542084c5af4-3217366.jpg",
36               "mimeType": "image/jpeg",
37               "relativeFilePath": "35\\18\\14\\
880x495_cmsv2_298e3b.b245cc1ddda0ddd7_1023f19d_16bad3f75bb_-7bf7.jpg"
38             }
39           ]
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

## 13.5.6 Write Access

### 13.5.6.1 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     {
15       "values": [
16         "Another EN statement",
17         "Another DE statement",
18         "Another FR statement"
19       ]
20     }
21   ]
22 }
```

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  }

```

### 13.5.6.2 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
[LookupRestArticleTestCharacteristic]",
22             "entityId": 8000
23         }
24     },
25     "values": [
26     [
27         {
28             "id": "10617@3",
29             "label": "ActualLookupValueLabel",
30             "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 185).

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 }
```



```

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  }

```

## 14 REST API Troubleshooting

- [Characteristics](#)(see page 378)
- [Structure Groups](#)(see page 378)
  - [Access to structure group fails](#)(see page 378)
  - [StructureGroup levels](#)(see page 379)
  - [Filtering on structure group attribute](#)(see page 379)
  - [Not possible to search by Structure Group Attribute Value](#)(see page 380)
- [Media Assets](#)(see page 380)
  - [Get media asset document with attributes fails](#)(see page 380)

### 14.1 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

### 14.2 Structure Groups

#### 14.2.1 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
```

## 14.2.2 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.

## 14.2.3 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)
```

## 14.2.4 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] [CoreExceptionHandler] 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.hsq.HSQParser.parseSearchQuery(HSQParser.java:62)
```

See also <https://mysupport.informatica.com/message/89002?et=watches.email.thread>

## 14.3 Media Assets

### 14.3.1 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
```

## 15 PUBLIC Server Health Status

### 15.1 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.

#### 15.1.1 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.

##### 15.1.1.1 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.

### 15.1.2 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.

#### 15.1.2.1 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"
  }
}
```

# Copyright

Copyright (c) 1993-2020 Informatica LLC. All rights reserved.

This Software and documentation contain proprietary information of Informatica LLC and are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright law. Reverse engineering of the software is prohibited. 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. This Software may be protected by [U.S. and/or](#) international Patents and other Patents Pending.

Use, duplication, or disclosure of the Software by the U.S. Government is subject to the restrictions set forth in the applicable software license agreement and as provided in DFARS 227.7202-1(a) and 227.7702-3(a) (1995), DFARS 252.227-7013©(1)(ii) (OCT 1988), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14

(ALT III), as applicable.

The information in this product or documentation is subject to change without notice. If you find any problems in this product or documentation, please report them to us in writing.

Informatica, Informatica Platform, Informatica Data Services, PowerCenter, PowerCenterRT, PowerCenter Connect, PowerCenter Data Analyzer, PowerExchange, PowerMart, Metadata Manager, Informatica Data Quality, Informatica Data Explorer, Informatica B2B Data Transformation, Informatica B2B Data Exchange Informatica On Demand, Informatica Identity Resolution, Informatica Application Information Lifecycle Management, Informatica Complex Event Processing, Ultra Messaging and Informatica Master Data Management are trademarks or registered trademarks of Informatica LLC in the United States and in jurisdictions throughout the world. All other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties, including without limitation: Copyright DataDirect Technologies. All rights reserved. Copyright © Sun Microsystems. All rights reserved. Copyright © RSA Security Inc. All Rights Reserved. Copyright © Ordinal Technology Corp. All rights reserved. Copyright © Aandacht c.v. All rights reserved. Copyright Genivia, Inc. All rights reserved. Copyright Isomorphic Software. All rights reserved. Copyright © Meta Integration Technology, Inc. All rights reserved. Copyright © Intalio. All rights reserved. Copyright © Oracle. All rights reserved. Copyright © Adobe Systems Incorporated. All rights reserved. Copyright © DataArt, Inc. All rights reserved. Copyright © ComponentSource. All rights reserved. Copyright © Microsoft Corporation. All rights reserved. Copyright © Rogue Wave Software, Inc. All rights reserved. Copyright © Teradata Corporation. All rights reserved. Copyright © Yahoo! Inc. All rights reserved. Copyright © Glyph & Cog, LLC. All rights reserved. Copyright © Thinkmap, Inc. All rights reserved. Copyright © Clearpace Software Limited. All rights reserved. Copyright © Information Builders, Inc. All rights reserved. Copyright © OSS Nokalva, Inc. All rights reserved. Copyright Edifecs, Inc. All rights reserved.

Copyright Cleo Communications, Inc. All rights reserved. Copyright © International Organization for Standardization 1986. All rights reserved. Copyright © ej-technologies GmbH. All rights reserved. Copyright © Jaspersoft Corporation. All rights reserved. Copyright © International Business Machines Corporation. All rights reserved. Copyright © yWorks GmbH. All rights reserved. Copyright © Lucent Technologies. All rights reserved. Copyright (c) University of Toronto. All rights reserved. Copyright © Daniel Veillard. All rights reserved. Copyright © Unicode, Inc. Copyright IBM Corp. All rights reserved. Copyright © MicroQuill Software Publishing, Inc. All rights reserved. Copyright © PassMark Software Pty Ltd. All rights reserved. Copyright © LogiXML, Inc. All rights reserved. Copyright © 2003-2010 Lorenzi Davide, All rights reserved. Copyright © Red Hat, Inc. All rights reserved. Copyright © The Board of Trustees of the Leland Stanford Junior University. All rights reserved. Copyright

© EMC Corporation. All rights reserved. Copyright © Flexera Software. All rights reserved. Copyright © Jinfonet Software. All rights reserved. Copyright © Apple Inc. All rights reserved. Copyright © Telerik Inc. All rights reserved. Copyright © BEA Systems. All rights reserved. Copyright © PDFlib GmbH. All rights reserved. Copyright © Orientation in Objects GmbH. All rights reserved. Copyright © Tanuki Software, Ltd. All rights reserved. Copyright © Ricebridge. All rights reserved. Copyright © Sencha, Inc. All rights reserved. Copyright © Scalable Systems, Inc. All rights reserved. Copyright © jqWidgets. All rights reserved. Copyright © Tableau Software, Inc. All rights reserved. Copyright © MaxMind, Inc. All Rights Reserved. Copyright © TMate Software s.r.o. All rights reserved. Copyright © MapR Technologies Inc. All rights reserved.

This product includes software developed by the Apache Software Foundation ( <http://www.apache.org/>), and/or other software which is licensed under various versions of the Apache License (the "License"). You may obtain a copy of these Licenses at <http://www.apache.org/licenses/>. Unless required by applicable law or agreed to in writing, software distributed under these Licenses is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the Licenses for the specific language governing permissions and limitations under the Licenses.

This product includes software which was developed by Mozilla ( <http://www.mozilla.org/>), software copyright The JBoss Group, LLC, all rights reserved; software copyright © 1999-2006 by Bruno Lowagie and Paulo Soares and other software which is licensed under various versions of the GNU Lesser General Public License Agreement, which may be found at [http:// www.gnu.org/licenses/lgpl.html](http://www.gnu.org/licenses/lgpl.html). The materials are provided free of charge by Informatica, "as-is", without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

The product includes ACE(TM) and TAO(TM) software copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University, Copyright (©) 1993-2006, all rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (copyright The OpenSSL Project. All Rights Reserved) and redistribution of this software is subject to terms available at <http://www.openssl.org> and <http://www.openssl.org/source/license.html>.

This product includes Curl software which is Copyright 1996-2013, Daniel Stenberg, < [daniel@haxx.se](mailto:daniel@haxx.se) >. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at [http:// curl.haxx.se/docs/ copyright.html](http://curl.haxx.se/docs/copyright.html). Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

The product includes software copyright 2001-2005 (©) MetaStuff, Ltd. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at [http://www.dom4j.org/ license.html](http://www.dom4j.org/license.html).

The product includes software copyright © 2004-2007, The Dojo Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at [http://dojotoolkit.org/ license](http://dojotoolkit.org/license).

This product includes ICU software which is copyright International Business Machines Corporation and others. All rights reserved. Permissions and limitations regarding this software are subject to terms available at <http://source.icu-project.org/repos/icu/icu/trunk/license.html>.

This product includes software copyright © 1996-2006 Per Bothner. All rights reserved. Your right to use such materials is set forth in the license which may be found at [http://www.gnu.org/software/kawa/ Software-License.html](http://www.gnu.org/software/kawa/Software-License.html).

This product includes OSSP UUID software which is Copyright © 2002 Ralf S. Engelschall, Copyright © 2002 The OSSP Project Copyright © 2002 Cable & Wireless Deutschland. Permissions and limitations regarding this software are subject to terms available at <http://www.opensource.org/licenses/mit-license.php>.

This product includes software developed by Boost ( <http://www.boost.org/>) or under the Boost software license. Permissions and limitations regarding this software are subject to terms available at [http:// www.boost.org/LICENSE\\_1\\_0.txt](http://www.boost.org/LICENSE_1_0.txt).

This product includes software copyright © 1997-2007 University of Cambridge. Permissions and limitations regarding this software are subject to terms available at <http://www.pcre.org/license.txt>. This product includes software copyright © 2007 The Eclipse Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at [http:// www.eclipse.org/org/documents/ epl-v10.php](http://www.eclipse.org/org/documents/epl-v10.php) and at <http://www.eclipse.org/org/documents/edl-v10.php>.

This product includes software licensed under the terms at <http://www.tcl.tk/software/tcltk/license.html>, <http://www.bosrup.com/web/overlib/?License>, [http:// www.stlport.org/doc/ license.html](http://www.stlport.org/doc/license.html), <http://asm.ow2.org/license.html>, <http://www.cryptix.org/LICENSE.TXT>, <http://hsqldb.org/web/hsqLicense.html>, <http://unit.sourceforge.net/doc/license.html>, <http://jung.sourceforge.net/license.txt>, [http://www.gzip.org/zlib/zlib\\_license.html](http://www.gzip.org/zlib/zlib_license.html), [http://www.openldap.org/software/release/ license.html](http://www.openldap.org/software/release/license.html), <http://www.libssh2.org>, <http://slf4j.org/license.html>, <http://www.sente.ch/software/OpenSourceLicense.html>, [http:// fusesource.com/downloads/license- agreements/fuse-message-broker-v-5-3- licenseagreement](http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3-licenseagreement); [http:// antlr.org/license.html](http://antlr.org/license.html); <http://aopalliance.sourceforge.net/>; <http://www.bouncycastle.org/licence.html>; <http://www.jgraph.com/jgraphdownload.html>; <http://www.jcraft.com/jsch/LICENSE.txt>; [http://jotm.objectweb.org/ bsd\\_license.html](http://jotm.objectweb.org/bsd_license.html); [http://www.w3.org/Consortium/Legal/ 2002/copyright-software-20021231](http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231); [http://](http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231)



[www.slf4j.org/license.html](http://www.slf4j.org/license.html); <http://nanoxml.sourceforge.net/orig/copyright.html>; <http://www.json.org/license.html>; <http://forge.ow2.org/projects/jaservice/>; <http://www.postgresql.org/about/licence.html>; <http://www.sqlite.org/copyright.html>; <http://www.tcl.tk/software/tcltk/license.html>; <http://www.jaxen.org/faq.html>; <http://www.jdom.org/docs/faq.html>; <http://www.slf4j.org/license.html>; <http://www.iodbc.org/dataspace/iodbc/wiki/iODBC/License>; <http://www.keplerproject.org/md5/license.html>; <http://www.toedter.com/en/jcalendar/license.html>; <http://www.edankert.com/bounce/index.html>; <http://www.net-snmp.org/about/license.html>; <http://www.openmdx.org/#FAQ>; [http://www.php.net/license/3\\_01.txt](http://www.php.net/license/3_01.txt); <http://srp.stanford.edu/license.txt>; <http://www.schneier.com/blowfish.html>; <http://www.jmock.org/license.html>; <http://xsom.java.net>; <http://benalman.com/about/license/>; <https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js>; <http://www.h2database.com/html/license.html#summary>; <http://jsoncpp.sourceforge.net/LICENSE>; <http://jdbc.postgresql.org/license.html>; <http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto>; <https://github.com/rantav/hector/blob/master/LICENSE>; <http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html>; <http://jibx.sourceforge.net/jibx-license.html>; <https://github.com/lyokato/libgeohash/blob/master/LICENSE>; <https://github.com/hjiang/jsonxx/blob/master/LICENSE>; <https://code.google.com/p/lz4/>; <https://github.com/jedisct1/libsodium/blob/master/LICENSE>; <http://one-jar.sourceforge.net/index.php?page=documents&file=license>; <https://github.com/EsotericSoftware/kryo/blob/master/license.txt>; <http://www.scala-lang.org/license.html>; <https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt>; and <http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html>.

This product includes software licensed under the Academic Free License (<http://www.opensource.org/licenses/afl-3.0.php>), the Common Development and Distribution License (<http://www.opensource.org/licenses/cddl1.php>) the Common Public License (<http://www.opensource.org/licenses/cpl1.0.php>), the Sun Binary Code License Agreement Supplemental License Terms, the BSD License (<http://www.opensource.org/licenses/bsd-license.php>), the new BSDLicense (<http://opensource.org/licenses/BSD-3-Clause>), the MIT License (<http://www.opensource.org/licenses/mitlicense.php>), the Artistic License (<http://www.opensource.org/licenses/artistic-license-1.0>) and the Initial Developer's Public License Version 1.0 (<http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/>).

This product includes software copyright © 2003-2006 Joe Walnes, 2006-2007 XStream Committers. All rights reserved. Permissions and limitations regarding this software are subject to terms available at <http://xstream.codehaus.org/license.html>. This product includes software developed by the Indiana University Extreme! Lab. For further information please visit <http://www.extreme.indiana.edu/>.

This product includes software Copyright (c) 2013 Frank Balluffi and Markus Moeller. All rights reserved. Permissions and limitations regarding this software are subject to terms of the MIT license.

This Software is protected by U.S. Patent Numbers 5,794,246; 6,014,670; 6,016,501; 6,029,178; 6,032,158; 6,035,307; 6,044,374; 6,092,086; 6,208,990; 6,339,775; 6,640,226; 6,789,096; 6,823,373; 6,850,947; 6,895,471; 7,117,215; 7,162,643; 7,243,110; 7,254,590; 7,281,001; 7,421,458; 7,496,588; 7,523,121; 7,584,422; 7,676,516; 7,720,842; 7,721,270; 7,774,791; 8,065,266; 8,150,803; 8,166,048; 8,166,071; 8,200,622; 8,224,873; 8,271,477; 8,327,419; 8,386,435; 8,392,460; 8,453,159; 8,458,230; 8,707,336; 8,886,617 and RE44,478, International Patents and other Patents Pending.

DISCLAIMER: Informatica LLC provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica LLC does not warrant that this software or documentation is error free. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.iv>

## NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.

2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY,

MISREPRESENTATION AND OTHER TORTS.