Informatica

Informatica® Multidomain MDM
10.5

Data Controls Implementation Guide

Informatica Multidomain MDM Data Controls Implementation Guide
10.5
September 2022

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2022-09-30

Table of Contents

Preface

Refer to the Informatica® *Multidomain MDM Data Controls Implementation Guide* to learn how to configure user interface controls to expose and embed MDM Hub data in third-party applications. Learn how to implement and configure tasks in IDC. Also, learn how to configure IDC using the Informatica Data Director Configuration Manager.

This guide assumes that you have read the *Multidomain MDM Overview Guide* and have a basic understanding of MDM Hub architecture and key concepts.

Attention: To use IDC, your Informatica MDM Hub implementation must have a license for the IDC feature.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

IDC Concepts

This chapter includes the following topics:

Usage of IDD

IDC controls are tightly bound to an IDD application. IDD is a generic application framework which, when combined with an IDD application configuration, provides a user interface customized for a customer's data model and data governance needs. In the same way, IDC controls provide generic functionality that becomes specific when bound to an IDD application configuration.

From an IDD application configuration, an IDC control can use subject area definitions, layouts, search, cleanse and validation, customizations, data security, and localization.

IDC Control Type

An IDC control type is a generic control. Once a control type is bound to an IDD application, the control is available for use. The Informatica MDM Hub provides three built in control types - History View, Hierarchy Manager, and Duplicate Prevention.

The IDC framework provides the ability to deploy additional control types. If you need additional control types, contact Informatica Global Customer Support.

IDC Control

An IDC control is an instance of an IDC control type that has been created and bound to an IDD application. Throughout this document, an IDC control is simply referred to as a control .

Levels of Integration

Integration of a control and a third-party application can range from loose to tight:

Integration Level	Description
Loose	The control is unaware of the containing application and simply embeds within the containing application. The only communication between the application and the control is through the invoking URL.
Tight	The control is aware of the containing application and can interact directly with that application.

The Hierarchy Manager and History View controls, provided with Informatica MDM Hub, provide loose integration.

The Duplicate Prevention control require tight integration with calling application.

CHAPTER 2

Implementation Process

This chapter includes the following topics:

Overview

This section describes the recommended high-level process for configuring IDC controls. This process should be used as a template for creating IDC implementation plans. The main goal is to outline the steps in the build/test cycle that would provide an efficient model for rapid IDC development. Such an incremental approach allows you to use the intermediate stages of the configuration process for getting additional feedback and validating requirements with the customer.

Before You Begin

This section assumes the following prerequisites:

- The MDM Hub, cleanse adapters, and Process Servers are already configured and operational in your environment. For more information, see the *Multidomain MDM Installation Guide*.
- The Operational Reference Store (ORS) schemas are configured and contain some test data. The Hub Console is used to create the configuration elements in the target ORS, such as base objects, packages, lookups, match path controls, and so on.
- All base objects and associated metadata need to be configured as SECURE in the Secure Resources tool in the Hub Console.
- The application for Data Director has been configured and tested as described in the *Multidomain MDM Data Director Implementation Guide*.
- Configuration and initial testing should be done using an Informatica MDM Hub user account with unrestricted privileges for the target ORS schemas. You can either use the admin account or any other account that is configured with all privileges to the ALL_GLOBAL_RESOURCES group. The ALL_GLOBAL_RESOURCES group does not include the custom resources added as part of the application. Custom resources must be configured individually.

For more information about Hub Console tools, see the Hub Console online help or the *Multidomain MDM Configuration Guide*.

Configuration Process

The IDC configuration process involves the following steps. Bear in mind that this is an iterative process, not a linear, one-time procedure.

Step 1. Build, Deploy, and Test the IDD Application

An IDC control is entirely dependent on a Data Director application for the structure and behavior of subject areas. Before creating the IDC control, the Data Director application should be built, deployed, and tested according to the instructions in the *Multidomain MDM Data Director Implementation Guide* or the Data Director Configuration Manager online help. Configured IDC controls are deployed in the same Java Memory Model (JMM) as the Data Director application instance within the same application server environment.

Step 2. Create the IDC Control

Use the IDD Configuration Manager (see “Configuring Controls in the IDD Configuration Manager” later in this document) to create IDC controls as part of a deployed IDD application. An IDD application can contain multiple IDC controls. Each control is given a unique name. Once created, the URL used to invoke the control is available (see “Format of the Control URL” later in this document). The URL contains parameters that specify the subject area and key to the data. Each control can be used with any subject area in the IDD application.

Step 3. Configure IDC Control Overrides

By default, an IDC control uses the configuration of the IDD application to determine its visualization. As described in “IDC Control Overrides” later in this document, layouts, custom actions, and user exits can be overridden for individual controls.

Step 4. Test the IDC Control in a Browser

The control’s URL can be used directly in a web browser, with appropriate values provided for the key. This provides a simple way to verify the control’s configuration.
Note: If you are using the Safari browser, in the browser's toolbar click **Preferences**, click **Privacy** and select **Never** for the **Block cookies:** option to allow and remember all cookies.

Step 5. Embed the IDC Control in a Third-party Application

Once a control has been tested, configure a third-party application to embed the web content of the control (see “Embedding Controls” later in this document). The details of doing this are specific to each third-party application. Embedding includes the invocation of the control URL with the data key for the currently-shown data.

CHAPTER 3

Configuring Controls in the IDD Configuration Manager

This chapter includes the following topics:

Creating a Control

The IDD Configuration Manager is used to create, edit and manage IDC controls. To create a control:

1. In the IDD Configuration Manager, edit the IDD application.
2. On the Controls tab, click **Add Control**.
3. Specify the name and display name for the control and the control type. The name must be unique among all controls defined in this IDD application.
4. Click **OK**.
5. Click **Save**. If the IDD application is already deployed, the control is ready to use. If not, first deploy the IDD application.
6. Select the control in the tree, then click **Show URL**. This displays the template for the URL that is needed to invoke the control.
7. Test the URL by running it in a browser.
8. Copy the URL and integrate it into the third-party application that will be used to invoke the control.

Format of the Control URL

The format of the control URL is:

```
http://<host>[:<port>]/bdd/bdc/<controlName>/[sag:<sagName>|sa:<saName>],<key>/<controlType>/
component.jsf[?username=<username>&password=<password>&bdd_name=<bddName>]
```

where:

Parameter	Description
Host	Name of the machine where the Informatica MDM Hub is hosted.
Port	Port number (defaults to port 80 if not specified in the URL).
controlName	Name of the control.
sagName	Name of the subject area group for the data to be displayed. This is required for HM and History controls.
saName	Name of the subject area. This is required for Duplicate Prevention control.
key	Key to the data to be displayed. This has different meaning depending on the control type. - Hierarchy Manager and History Controls: This can be either of the following: “rowid:<rowidValue>” OR “sourceKey:<sourceKeyValue>,systemName:<systemNameValue>” - Duplicate Prevention control: This should contain search criteria to perform search for duplicates, values of Hub match columns that will be used for matching: “fieldName:<fieldValue>[,fieldName:<fieldValue>]” "mc.<matchColumnName>:<*value*>[, mc.<matchColumnName>:<*value*>]" If match column is based on column of DATE type, then its value must be specified as string in the format 'M-d-yyyy H-m-s' (for example date 'March 17, 2010' should be passed as '3-17-2010 0-0-0')
controlType	For the default control types: hm, history or duplicate prevention.
username	User name to use to authenticate this request.
password	Password to use to authenticate this request.
bdd_Name	The name of the IDD application to which the control is bound.

Note:

- Any characters in the parameters that are not allowed in a URL must be double encoded (see HTML URL Encoding Reference for details on URL encoding). Double encoding (running the encoding process twice) is requested on purpose as it is needed to allow web server to accept requests containing slashes ("/" and "\") in parameters, when single-encoded, requests containing single-encoded slashes are thrown back by web servers. Only the parameter values should be double encoded.
- The username, password, and bdd_name parameters are optional and can be used to automatically log a user into the control. This should be used with caution, however, because the URL is not encrypted (even if using HTTPS). Supplying login credentials as parameters can be useful when there is a read-only user account that can be accessed by all users.

Localization

All of the resources for the built-in controls are included in the standard Data Director IDD resource bundles. Localization of these controls is done as part of the process of localizing the IDD application. IDD applications and IDC controls share context in the web browser. When a user selects a language in the IDD application, that selection is saved as a preference that will carry over to IDC controls. For more information about localizing IDD applications, see the *Multidomain MDM Data Director Implementation Guide*.

User Authentication and Password Encryption

You can configure authentication for URLs in Data Director (IDD). When enabled, when users log in, they pass their user name and password to the URL in Data Director. You can also configure Data Director to pass the user name and password in an encrypted format.

By default, as a security precaution, the automatic login functionality is disabled. To enable this functionality so that login credentials can be passed as parameters in an URL, add the following setting to `cmxserver.properties` and restart the application server:

```
cmx.bdd.enable_url_authentication=true
```

If you use a password encryption tool to encrypt a password, you must use the same key to encrypt and decrypt the password.

For more information about configuring the `cmxserver.properties` file, see the "MDM Hub Properties" appendix in the *Multidomain MDM Configuration Guide*.

For more information about encrypting the user name and password, see the "User Interface Extensions" section in the *Multidomain MDM Data Director Implementation Guide*.

For more information about the password encryption tool, see the *Multidomain MDM Resource Kit Guide*.

CHAPTER 4

IDC Control Overrides

This chapter includes the following topics:

Overview

By default, the IDD application configuration determines how subject areas are displayed in the controls. For individual controls, you can manually override certain display settings and behaviors. Overrides do not affect the underlying data structure of the subject areas in the IDD application.

By default, IDC controls have no custom actions or user exits defined (these are not inherited from the IDD application). You can use IDC overrides to extend functionality by adding custom actions and user exits.

XML Files and Root Elements

Overrides are managed by creating a set of XML files. Each control has one set of files. Therefore, each control can have individual overrides. One or all of these files are combined in a ZIP file that can be uploaded to the IDD Configuration Manager to configure the control.

The MDM Hub Resource Kit includes the XML schema, `siperian-bdd-config-6.xsd`, that defines the IDD configuration file and these IDC configuration files. The HTML documentation for the XML schema describes individual elements and attributes in the schema. The following table describes the names of the XML files along with the XML root element that each file must contain.

Element	File Name	XML Root Element	Description	Applicability
Layout	layout-config.xml	layoutConfig	Controls which fields from each base object are displayed, as well as field sizes and number of columns for each row.	HM, History, Duplicate Prevention
Custom actions	ela-config.xml	externalLinkActionsConfig	Controls the custom actions that are shown for subject areas and HM objects.	HM
User exits	ue-config.xml	userExitsConfig	Controls the user exits that are applied to subject area and HM operations.	HM
Properties	bdc-config.xml	bdcConfiguration	Specifies control-specific configuration properties.	Duplicate Prevention
Mapping	pmc-config.xml	pmcConfig	Specifies target system name and import settings for Duplicate Prevention control	Duplicate Prevention

Steps to Override IDC Controls

To override the default IDD visualization and behavior for an IDC control:

1. Create the set of override XML files for the control.
2. Create a ZIP file containing the override XML files. You can use any name, as it will be stored in the IDC configuration section of the database. All configuration files must be in the root directory of the ZIP file.
3. In the IDD Configuration Manager, edit the IDD application associated with the control you want to override.
4. On the Controls tab, select the IDC control you want to override.
5. Click Edit.
6. In the Edit dialog, click Browse, select the ZIP file containing the overrides, and then choose OK.

   Note: Only when uploading the ZIP file containing the overrides for duplicate prevention control, a validation check will be performed on the selected ZIP file of pmc-config.xml and validation error messages are displayed, if any.
7. Save changes to the IDD application. If the IDD application is already deployed, the IDC control changes, including the overrides, are applied automatically.

Using XML Tools to Configure Configuration XML Files

The Informatica MDM Hub Resource Kit includes an XML schema, `siperian-bdd-config-6.xsd`, that defines the format for the IDD and IDC configuration XML files. This XSD file is very useful when working with XML editors. It can guide you in editing the file and, most importantly, it is used by the editor to verify the correctness of the XML in the IDD or IDC configuration file. Every configuration XML file should pass this test before being imported into the IDD Configuration Manager.

While a simple text editor can be used to modify the IDD configuration, there are many XML editing tools that make working with XML much easier, including:

Editor	URL
XML Copy Editor	http://xml-copy-editor.sourceforge.net/
XML Spy	http://www.altova.com/products/xmlspy/xmlspy.html
oXygen	http://www.oxygenxml.com/

Layout Overrides

The following example shows a layout override for a Person subject area. This override allows you to choose a subset of the columns and child objects for the subject area to be displayed when the IDC control is invoked.

- The name attribute in the *sagLayout* and saLayouts elements refers to the name of the subject area group and subject area in the IDD application configuration. The XML structure allows for more than one tabLayout for each subject area. For IDC controls, only the first one is used.
- The tabLayout configures a set of columns for a primary object, child, or grandchild. Child and grandchild tabLayouts are nested in their parent *tabLayout* . The attribute on a child or grandchild tabLayout refers to the name of a child or grandchild object in the IDD application configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<layoutConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              name="test"
              xsi:noNamespaceSchemaLocation="./siperian-bdd-config-6.xsd">

    <sagLayout name="Party">
        <saLayouts name="Person">
            <saLayout name="One" compactMode="true">
                <tabLayout columnNum="3">
                    <column columnUid="C_PARTY|FIRST_NAME"/>
                    <column columnUid="C_PARTY|MIDDLE_NAME"/>
                    <column columnUid="C_PARTY|LAST_NAME"/>
                    <column columnUid="C_PARTY|BIRTHDATE"/>
                    <column columnUid="C_PARTY|DISPLAY_NAME"/>
                    <tabLayout name="Name">
                        <column columnUid="C_PARTY_NAME|NAME"/>
                    </tabLayout>
                    <tabLayout name="Phone" columnNum="3">
                        <column columnUid="C_PARTY_PHONE|PHONE_COUNTRY_CD"/>
                        <column columnUid="C_PARTY_PHONE|PHONE_NUM"/>
                        <column columnUid="C_PARTY_PHONE|IS_VALID_IND"/>
                    </tabLayout>
                </tabLayout>
            </saLayout>
        </saLayouts>
    </sagLayout>
</layoutConfig>
```

Adding Custom Actions

The following example shows custom action (external link) overrides for a Person subject area and HM configuration.

- The name attribute in the sagExternalLinkActions and saExternalLinkActions elements refers to the name of the subject area group and subject area in the IDD application configuration.
- The usage of the externalLinkAction element is the same as in the IDD application configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<externalLinkActionsConfig name="test" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
                           xsi:noNamespaceSchemaLocation="./siperian-bdd-config-6.xsd">

    <sagExternalLinkActions name="Party">
        <saExternalLinkActions name="Person">
                <externalLinkAction callback="false" displayName="Google search"
                         name="person_google_search_action_bdc">
                    <externalLink name="person_google_search_child_link" type="IFRAME"
                             url="http://www.google.com/search">
                        <param bddParamName="C_PARTY|DISPLAY_NAME" name="q" />
                        <param name="hl" staticValue="en" />
                    </externalLink>
                </externalLinkAction>
        </saExternalLinkActions>
    </sagExternalLinkActions>
    <hmExternalLinkActions uid="Default|Master" logicalOrsGroupName="Test">
            <externalLinkAction callback="false" displayName="Google Search"
                     name="hm_google_search_action_bdc">
                <externalLink name="hm_google_search_link" type="IFRAME"
                         url="http://www.google.com/search">
                    <param bddParamName="SELECTED_GRAPH_OBJECTS" name="q" />
                    <param name="hl" staticValue="en" />
                </externalLink>
             </externalLinkAction>
    </hmExternalLinkActions>
</externalLinkActionsConfig>
```

Adding User Exits

The example below shows user exit overrides for a Person subject area and HM configuration.

- The name attribute in the sagUserExits and saUserExits elements refers to the name of the subject area group and subject area in the IDD application configuration.
- The usage of the userExits element is the same as in the IDD application configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<userExitsConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 name="test"
                 xsi:noNamespaceSchemaLocation="./siperian-bdd-config-6.xsd">

    <sagUserExits name="Party">
```

```
        <saUserExits name="Person">
            <userExits className="com.siperian.bdd.userexits.sample.SaveHandler"/>
            <userExits
className="com.siperian.bdd.userexits.sample.CustomActionProvider"
                 actionName="Launch CustomActionProvider from BDC"/>
        </saUserExits>
    </sagUserExits>
    <hmUserExits uid="Default|Master" orsUid="Test">
        <hmRelationshipTypeUserExits uid="HM_RELATIONSHIP_TYPE.employes">
            <userExit
className="com.siperian.bdd.userexits.sample.HMRelationshipHandler"/>
        </hmRelationshipTypeUserExits>
        <userExits className="com.siperian.bdd.userexits.sample.GraphUserExit"
             actionName="BDC Graph Action"/>
    </hmUserExits>
</userExitsConfig>
```

Properties

IDC controls can expose properties that are used to modify their behavior or appearance. The Hierarchy Manager and History View controls do not expose any properties. Duplicate Prevention control exposes `parentUrl` property that can be used to enable JavaScript messages for IE7 browser.

The following example shows the specification of two sample properties.

```
<?xml version="1.0" encoding="UTF-8"?>
<bdcConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="./siperian-bdd-config-6.xsd">

    <property name="sampleProperty1">value1</property>
    <property name="sampleProperty2">value2</property>
</bdcConfiguration>
```

Duplicate Prevention Configuration

Duplicate Prevention control requires configuration file to work properly. This is used to specify the name of the target Source System and to control the behavior of the import functionality.

The following is a sample `pmc-config.xml` file, with a minimal Duplicate Prevention configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<pmcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="siperian-bdd-config-6.xsd"
systemName="SFDC">
</pmcConfig>
```

Associate a Match Rule Set

When you create a new record in an external source system such as Salesforce.com, you can associate a match rule set to perform a search for duplicates. This allows you to leverage the cleansed and corrected data that already exists in the MDM server and prevent the creation of duplicate data at the external source.

The following code snippet is an example on how to configure `pmc-config.xml` file, to associate a match rule set for Duplicate Prevention control.

```
<?xml version="1.0" encoding="UTF-8"?>
<pmcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="siperian-bdd-config-6.xsd" systemName="Admin"
```

```
import="true">

    <matchRuleSet saName="Person" uid="C_PARTY|IDL" type="AUTO"/>

    <matchRuleSet saName="Organization" uid="C_PARTY|IDL" type="BOTH"/>

    <mdmEntity name="Person" sourceEntity="PersonEntity">
        <columnMapping columnUid="C_PARTY|FIRST_NAME" sourceColumn="FirstName"/>
        <columnMapping columnUid="C_PARTY|LAST_NAME" sourceColumn="LastName"/>

        <mdmEntity name="ShipAddresses" sourceEntity="ShipAddressesEntity" maxOccur="10">
            <columnMapping columnUid="C_ADDRESS|CITY_NAME" sourceColumn="City"/>

            <mdmEntity name="ShipAddressService" sourceEntity="ShipAddressServiceEntity">
                <columnMapping columnUid="C_ADDRESS_CHILD5|COLUMN1"
sourceColumn="Column1"/>
            </mdmEntity>
        </mdmEntity>
    </mdmEntity>
</pmcConfig>
```

In this example, the saName attribute is the name of subject area, where duplicate search will be performed. The uid attribute is the match rule set name (IDL) defined for the base object (C_PARTY). The type attribute is the matchtype for which you can define values either AUTO or BOTH.

System Name

The systemName attribute contains the name of the target Source System, which is the Source System configured in the Hub that is used to provide to the Hub data from the application embedding Duplicate Prevention control. This is a required configuration attribute that effects actions available for matched records found by Duplicate Prevention control.

Related Topics:

- "Duplicate Prevention" on page 24

Import Configuration

Duplicate Prevention control lets you create new records using data available in the Hub, in the application embedding the control. The control sends data of the selected matched record as content of ON_IMPORT event. The matched record is then serialized into a string in JSON format (in this string fields of numeric types are converted into string using Java String.valueOf method, values of date/time columns are converted into string using format 'M-d-yyyy H-m-s'). The format of JSON string and the list of imported objects and fields are configured using the mdmEntity element:

```
<mdmEntity name="<IDD object name>" sourceEntity="<object name in JSON>" maxOccur="<max
objects to export>">
        <columnMapping columnUid="<column UID>" sourceColumn="<column name in JSON>"/>
        <columnMapping columnUid="<column UID>" sourceColumn="<column name in JSON>"/>
       ...
</mdmEntity>
```

The mdmEntity element has following attributes:

name

Name of the IDD object (SubjectArea, Child or Grandchild) that must be imported.

sourceEntity

Name of the object in the generated JSON string.

If the embedding application converts this string into a JavaScript object, then this name must be a valid JavaScript variable name and must not start with a number.

maxOccur

Maximum number of objects to be imported for children and grandchildren controls.

If this attribute is not specified, then only 10 objects are imported. To import all objects, the value must be set to `-1`. This attribute is ignored for Logical One:One children, where only 1 child is imported.

Only objects and fields that are explicitly specified in the configuration are imported. If import configuration is not specified, then all fields of the SubjectArea's PrimaryObject are imported.

Consider a scenario where the SubjectArea Perso, has children Telephones and Addresses, and the Addresses child has the AddressesSerivces grandchild. If the imported data must contain only PrimaryObject's data, all the Addresses children and no more than five AddressesServices grandchildren, then the following import configuration can be used:

```
<?xml version="1.0" encoding="UTF-8"?>
<pmcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="siperian-bdd-config-6.xsd" import=”true”
systemName="SFDC">
    <mdmEntity name="Person" sourceEntity="Person">
        <columnMapping columnUid="C_PARTY|FIRST_NAME" sourceColumn="FirstName"/>
        <columnMapping columnUid="C_PARTY|LAST_NAME" sourceColumn="LastName"/>

        <mdmEntity name="Addresses" sourceEntity="Addresses" maxOccur=”-1”>
            <columnMapping columnUid="C_ADDRESS|CITY" sourceColumn="City"/>
            <columnMapping columnUid="C_ADDRESS|STATE" sourceColumn="State"/>

            <mdmEntity name="AddressesServices" sourceEntity="AddressesServices "
maxOccur=”5”>
              <columnMapping columnUid="C_ADDRESS_SERIVE|PHONE" sourceColumn="Phone"/>
            </mdmEntity>

        </mdmEntity>
    </mdmEntity>
</pmcConfig>
```

following sample code handles ON_IMPORT event and works with JSON string:

```
function handleImportEvent(event) {
  var importedRecordString = event[‘record’];

  // convert JSON string into JavaScript object
  var importedRecord = eval(‘(’ + importedRecordString +’)’);

  // get FIRST_NAME field of PrimaryObject
  var firstName = importedRecord.Person.FirstName

  // get CITY field of first child Addresses
  var city = event[record].Addresses[0].City

  // get PHONE field of AddressServices grandchild
  var phone = event[record].Addresses[0]. AddressServices[0].Phone;

}
```

The `import` attribute can be used to disable the import functionality. If it is disables, then the import button is not displayed. By default, the import functionality is enabled). To disable the import functionality, set the import attribute to `false`, as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<pmcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="siperian-bdd-config-6.xsd" import=”false”
systemName="SFDC">
</pmcConfig>
```

JavaScript Messaging in Internet Explorer 7

The Duplicate Prevention control sends events to the embedding application by using postmessage plugin. This plugin uses the `window.postMessage` JavaScript function that is not supported by Internet Explorer 7. You must use a workaround based on window location hash polling, by providing the current URL of the parent window embedding Duplicate Prevention control (full URL string displayed in the browser's navigation bar) to the postmessage plugin. This is required for setting the location hash of the target window. You must pass the value of this URL to the Duplicate Prevention control by using one of the following methods:

- If the URL does not change in your environment, then specify the URL in the bdc-config.xml properties file using the parentUrl parameter, as shown in the following example:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <bdcConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="siperian-bdd-config-6.xsd">
  <property name="parentUrl">http://c.na7.visual.force.com/apex/SomePage</property>
  </bdcConfiguration>
  ```

  Note: The value specified in the properties file must exactly match the URL in the navigation bar of the browser. This value is case-sensitive.

  You cannot use this method if single control instance needs to be embedded in multiple pages with different URLs.

- If the URL changes in your environment, or if single control instance needs to be embedded in multiple pages with different URLs, then dynamically detect the URLs on the embedding application side and pass it to the control as the parentUrl parameter inside the URL of the control. The value of the parentUrl parameter must be double encoded.

  The following example shows a JavaScript code that detects the current URL displayed in the browser and constructs URL of Duplicate Prevention control with parentUrl parameter to embed it into the HTML page iFrame with control:

  ```
  // get current URL
  var parentUrl = document.location.href;
  // double encode URL using standard JavaScript function ‘encodeURIComponent’
  parentUrl = encodeURIComponent(encodeURIComponent(parentUrl));
  // embed into the page iframe with Duplicate Prevention control
  document.writeln('<iframe src="http://<host>:<port>/bdd/bdc/<component name>/
  sa:<subject area>,<match parameters>,parentUrl:' + parentUrl + '/proactive_match/
  component.jsf" </iframe>');
  ```

Also, Internet Explorer 7 has restrictions on the size of the imported data. In Internet Explorer 7, an URL cannot be more than 2083 characters in size. The string with data is truncated and a JavaScript error is displayed if the URL and the imported data do not comply with the size restriction.

CHAPTER 5

IDC Controls

This chapter includes the following topics:

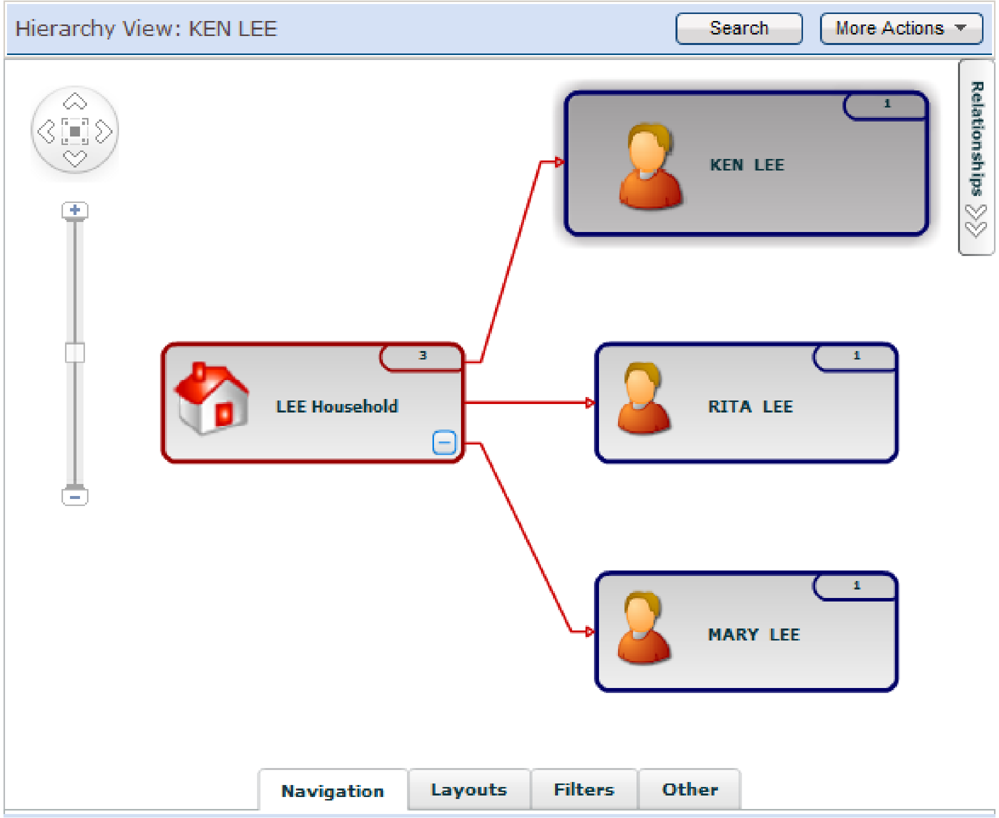
Overview

This section describes the features and functionality of the built-in IDC controls and the ways in which they might differ from their IDD counterparts.

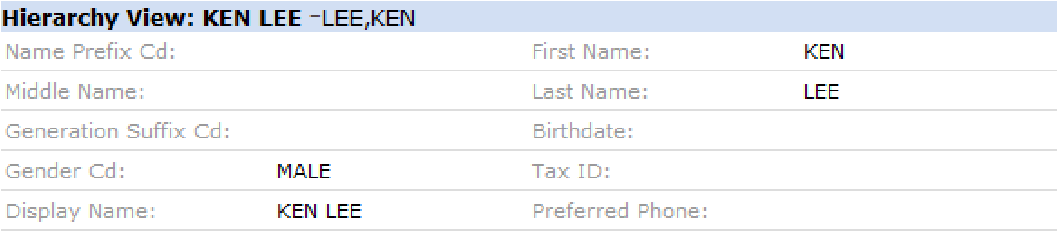
Clickable Path

The amount of space available on the screen is more limited with IDC controls than with IDD applications. Large dialog boxes in particular do not work well with the limited space. In IDC controls, large dialog boxes are replaced with overlay panels that use the full control space and a clickable path in the title bar to navigate back to the starting panel.

As an example, here is the main display of the Hierarchy Manager control.



When the user selects an entity in the graph and chooses View Detail, a full panel overlay is used to display the entity information. The title bar of this panel consists of a clickable path showing the title of the main page and the current page. Clicking on the title of the main page returns to that view.



Hierarchy View: KEN LEE -LEE,KEN

Name Prefix Cd:		First Name:	KEN
Middle Name:		Last Name:	LEE
Generation Suffix Cd:		Birthdate:	
Gender Cd:	MALE	Tax ID:	
Display Name:	KEN LEE	Preferred Phone:	

Hierarchy Manager

The Hierarchy Manager control provides functionality similar to the Hierarchy View in an IDD application. It has been modified to provide capabilities that are appropriate for the context of an embedded control. When the component is invoked, it displays the specified entity and directly related entities (one hop).

The following list describes differences between the Hierarchy View in IDD applications and the IDC Hierarchy View:

Title bar – the IDC control does not include:

> Add New Entity button – this control is for viewing and managing relationships. Entities must be added in IDD.

Full Screen Mode checkbox – not needed.

More Actions menu – the IDC control does not include:

Show History – the Hierarchy History View is not included.

Show Bookmark – not needed.

Entity Actions – the IDC control does not include:

View / Cross References

Edit / Edit Entity

Edit / Delete Entity

Edit / Create Task

Find / Duplicates

Find / Merge Candidates

History View

The History View control provides the same features as the History View in IDD. The display layout is rearranged to accommodate the more limited display space.

- In IDD applications, the history timeline displays at the top of the form with the point-in-time history view of the subject area at the bottom of the form.
- In IDC controls, the Event and Entity Details are displayed in a full panel overlay, with a clickable path to return to the timeline.

Duplicate Prevention

The goal of the Duplicate Prevention control is to bring the potential match capabilities currently available within the IDD to business users, who create master data in external source system within enterprise applications such as Salesforce.com, SAP, and Oracle ERP.

When a user creates a new record in the external source system, the component displays the potential matches and provides the user with the following options:

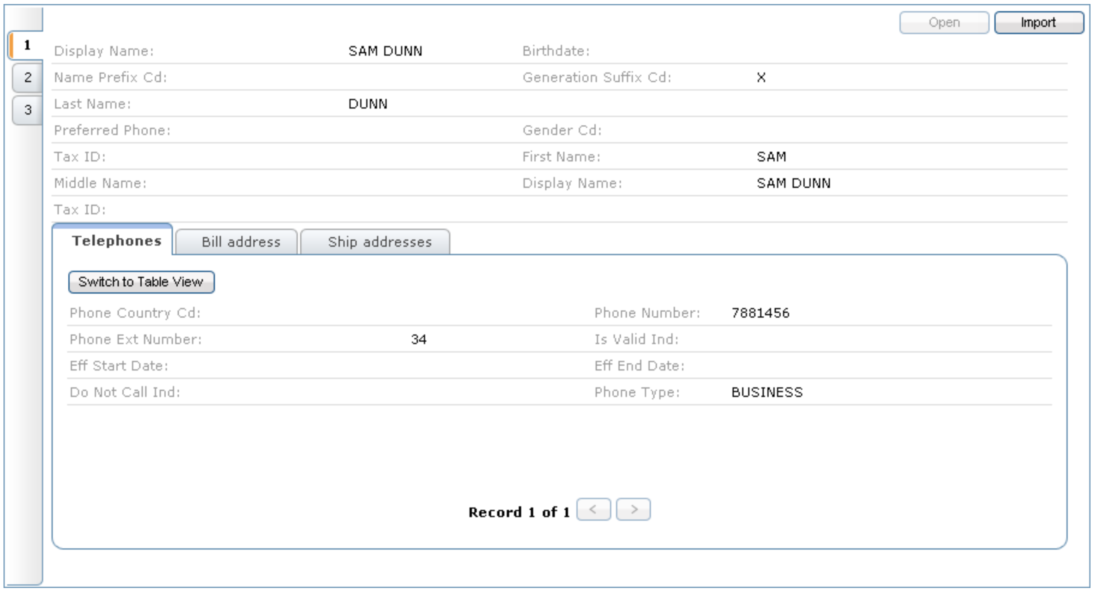
- Navigate to the matched record, if it exists in source system.
- Ability to create a new record in the source system, if it exists in Hub but does not have an XREF for the calling application.

This allows the business users to leverage the cleansed and correct data that may already exist in the MDM server and prevent the creation of duplicate data at the external source.

The control is not intended to address the overall integration or synchronization requirements between the external application and the Hub server. It is assumed that this is in place (batch and/or real-time).

Note: This control is designed to be application agnostic. There must be a dedicated connector for the specific application that embeds the control.

The Duplicate Prevention control performs search for duplicates using search criteria passed in the control's URL. If matched records are found, it displays them in a view similar to the Potential Matches IDD dialog, as shown:



Display Name:	SAM DUNN	Birthdate:	
Name Prefix Cd:		Generation Suffix Cd:	X
Last Name:	DUNN		
Preferred Phone:		Gender Cd:	
Tax ID:		First Name:	SAM
Middle Name:		Display Name:	SAM DUNN
Tax ID:			

Phone Country Cd:		Phone Number:	7881456
Phone Ext Number:	34	Is Valid Ind:	
Eff Start Date:		Eff End Date:	
Do Not Call Ind:		Phone Type:	BUSINESS

Record 1 of 1

Note: Search criteria are passed to the Duplicate Prevention control as values of Hub match columns. To perform search for duplicates, control uses the searchMatch API with matchType=NONE. This matchType option is intended for searching, and therefore does not use a predefined match rule set; instead a dynamic rule set based on the match columns passed to the control is generated. Also, if base object is configured to use a Fuzzy match/search strategy, then the searchMatch API requires the value of the fuzzy match key to be specified.

If multiple matched records are found, then a user can navigate between them using the scrolling control to the left. The following buttons are available for each matched:

Open

> The Open button is intended to be used to open matched record in the application embedding the control. This button is enabled if a matched record has an XREF record from the Source System that is configured as target Source System for the Duplicate Prevention control.

Import

> The Import button is intended to create new record, in the application embedding the control, by using data of matched record found in Hub. This button is enabled if a matched record does not have an XREF record from the Source System configured as target Source System for the Duplicate Prevention control. If the import functionality is not needed, then control can be configured not to display the Import button.

Related Topics:

- "Import Configuration" on page 19

Messaging

You need to establish connectivity from a component embedded in iFrame to the embedding application, as the embedded component contains action buttons that trigger some logic in the external application. The connectivity is implemented as messaging by using the window.postMessage JavaScript function, which allows cross-window or cross-domain messaging.

Messaging is supported by browsers such as Safari 4, Firefox 3, and Internet Explorer 8. To use messaging for Internet Explorer 7, additional configuration steps are required. The Duplicate Prevention control sends events to the embedding application by using postmessage plugin. Duplicate Prevention component sends messages in a predefined format, which the embedding application needs to handle. Each messaging event is a JavaScript object with an action field identifying the type of event.

The Duplicate Prevention component generates the following types of events:

ON_LOAD

The ON_LOAD event is generated when search for duplicates and rendering of the component is completed. This event has a `duplicatesFound` field that provides an indication if matched records are found.

ON_OPEN

The ON_OPEN event is generated when a user clicks the Open button. This event has an `id` field that identifies selected matched records. It contains the value of the PKEY_SRC_OBJECT field of XREF record from the target Source System (primary key value from the source system). Code handling this event can use this `id` to find and open matched records in the embedding application.

ON_IMPORT

The ON_IMPORT event is generated when a user clicks the Import button. This event has a `record` field that contains data of selected matched records serialized into the string in JSON format. Code handling this event can use a `record` data to create new records in the embedding application.

The Duplicate Prevention control uses the `idd_pmc_event` string as the postmessage type of event, and this string should be used to register handler for events generated by the control.

The following is a sample HTML page embedding a Duplicate Prevention control that shows you how to register a handler for events generated by the component:

```
<html>
<head>
 <!-- include postmessage JavaScript plugin -->
 <script type="text/javascript" src="http://postmessage.freebaseapps.com/
postmessage.js"></script>

 <script type="text/javascript">

 // function handling events
 function handleEvent(data) {
    switch(data.action) {
    case 'ON_LOAD':
      // handle ON_LOAD event
      break;
        case 'ON_IMPORT':
          // handle ON_IMPORT event
          break;
        case 'ON_OPEN':
          // handle ON_OPEN event
          break;
      }
 }

 // register function 'handleEvent' as handler for events generated by Duplicate
Prevention component
 function bindHandler() {
    pm.bind("idd_pmc_event", function(data) {
       handleEvent(data);
   });
 }
</script>
```

```
</head>

<!-- register event handler from onload event -->
<body onload='bindHandler();'>
  <!-- embed Duplicate Prevent component -->
  <iframe src="http://<host>:<port>/bdd/bdc/<component name>/sa:<subject area>,<match
parameters>/proactive_match/component.jsf></iframe>
</body>

</html>
```

RELATED TOPICS:

- “Import Configuration” on page 19
- “JavaScript Messaging in Internet Explorer 7” on page 21

Error Handling

Search for matched records is not performed in situations such as when a Duplicate Prevention component is not active or a component is configured incorrectly.

If some error occurs during the initialization of Duplicate Prevention component or during the search for duplicates, then the component displays a standard IDC error page with a detailed error message and does not generate any event. It is recommended that you implement some user interface controls in the application embedding Duplicate Prevention so that a user is able to cancel an operation or skip the component screen in case of an unexpected error.

CHAPTER 6

Embedding Controls

This chapter includes the following topics:

Overview

At present IDC controls are loosely coupled with third-party application. The URL that invokes the control is the only interface between the control and the third-party application. Embedding an IDC control in a third-party application depends on the particular application.

Note: Embedding more than one IDC component in the same html page is not supported.

Loose Coupled Controls

The Hierarchy Manager and History IDC controls are loosely coupled with the third-party applications in which they are embedded. The URL that invokes the control is the only interface between the control and the third-party application. The control is unaware of the containing third-party application and does not communicate with it.

The details of embedding an IDC control in a third-party application depend on the particular application. As an example, here is a Salesforce.com VisualForce page for an Account that includes a Hierarchy Manager control.

```
<apex:page standardController="Account" showHeader="true" tabStyle="account" >
   <style>
      .activeTab {background-color: #236FBD; color:white; background-image:none}
      .inactiveTab { background-color: lightgrey; color:black; background-image:none}
   </style>
   <apex:tabPanel switchType="client" selectedTab="tabdetails"
                  id="AccountTabPanel" tabClass="activeTab"
                  inactiveTabClass="inactiveTab">
      <apex:tab label="Details" name="AccDetails" id="tabdetails">
         <apex:detail relatedList="false" title="true"/>

         <apex:iframe
             src="http://hostname:port/bdd/bdc/hmEn/sag:Party,systemName:SFA,sourceKey:
```

```
{!$CurrentPage.parameters.id}/hm/component.jsf"
             height="500" scrolling="false"
id="siperianHM1"/>

         <apex:relatedList subject="{!account}" list="contacts" />
         <apex:relatedList subject="{!account}" list="opportunities" />
         <apex:relatedList subject="{!account}" list="OpenActivities" />
      </apex:tab>
   </apex:tabPanel>
</apex:page>
```

In this example, the `http://hostname:port/bdd/bdc/hmEn/sag:Party,systemName:SFA,sourceKey: {!$CurrentPage.parameters.id}/hm/component.jsf` URL invokes the control. This URL uses the system name and source key to invoke the control. The system name is a constant – SFA. The source key is dynamic - {!$CurrentPage.parameters.id}. This is the VisualForce syntax used to substitute the ID for the current account.

Duplicate Prevention Control

The Duplicate Prevention IDC control is tightly coupled with the application in which it is embedded.

An external application must generate an URL invoking control by using record data entered by the user and handle events generated by the Duplicate Prevention control. The code required to embed a Duplicate Prevention control in the external application depends on this application. This section provides an example of Duplicate Prevention control integration into the Salesforce.com application. You must be familiar with the Salesforce platform, and understand the Visualforce Pages syntax and Apex Code.

Salesforce Scenario

In this scenario, an organization uses the Salesforce application to store information about customers.

The organization uses the Hub Server to maintain customer master data (data is synchronized with the Hub Server through batch processes) but continues to use the Salesforce application to enter new customer records. In this scenario, duplicate customer records can be created in the Salesforce application. The potential matches or the duplicate prevention capabilities available in IDD need to be extended to the Salesforce application, to detect duplicate data at the time of new record entry.

Requirements

The embedded Duplicate Prevention control must ensure that duplication of records is not allowed.

The implemented solution must help a business user to avoid creating duplicate customer records. When a user tries to save a new record, search for duplicate records already existing in the Hub Server must be executed. If no match is found, then the new record is saved. Otherwise, the matched records are displayed to the user. The following options must be available to the user:

- If a matched record found in the Hub Server exists in the Salesforce application, then a user should be able to open it.
- If a matched record found in the Hub Server does not exist in the Salesforce application, then a user should be able to create a new record using the data of the matched record.
- If an analysis of potentially matched records proves that the records are not duplicates, then a user should be able to save the new record.
- If an error occurs during the search for duplicate records, then a user should be able to save the new record or cancel the save.

Implementation

The following prerequisites are assumed for integration of Duplicate Prevention control into the Salesforce applications:

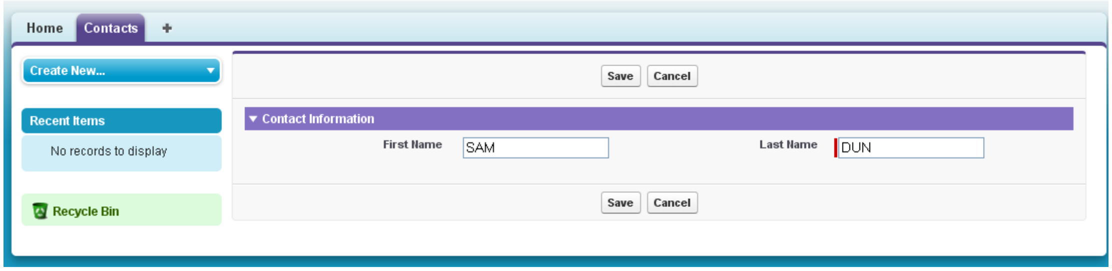
- Customer data is stored in the Salesforce application as a standard Salesforce object, Contact.
- Data from the Salesforce application is loaded into the Hub Server. Source System for the data from Salesforce is named SFDC and the internal field Contact.id, which are used to uniquely identify Salesforce objects mapped as PKEY_SRC_OBJECT column in the Hub. The Hub Server base object that stores customer records is named C_CUSTOMER.
- The C_CUSTOMER base object is configured to use a fuzzy match/merge strategy. It has a match column named Customer_Name, with type Person_Name, which is a Fuzzy Match Key. This match column is used to find matches on full names of people. For more information, see the section on configuring the match process in the *Multidomain MDM Configuration Guide*.
- The IDD application is configured and deployed. SubjectArea that uses C_CUSTOMER base object for its PrimaryObject is named Customer.
- An instance of Duplicate Prevention IDC control is created as part of the IDD application, with the name **dp.**

Firstly, control should be configured to use the Source System, SFDC, that was used to load data from Salesforce as the target system. Also, assume that for importing records from the Hub Server, we need values of PrimaryObject columns, FIRST_NAME and LAST_NAME only. During the import into Salesforce these columns are mapped to the corresponding columns of Contact object. The following configuration file, pmc-config.xml, must be uploaded to configure the control:

```
<?xml version="1.0" encoding="UTF-8"?>
<pmcConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="siperian-bdd-config-6.xsd"
systemName="SFDC">
    <mdmEntity name="Customer" sourceEntity=" Customer">
        <columnMapping columnUid=" C_CUSTOMER |FIRST_NAME" sourceColumn="FirstName"/>
        <columnMapping columnUid=" C_CUSTOMER |LAST_NAME" sourceColumn="LastName"/>
        </mdmEntity>
</pmcConfig>
```

On the Salesforce side, the standard process of new contact creation should be overridden to use Duplicate Prevention capabilities. In this example a two-step wizard is used instead of the default, new Contact page:

1. User enters new Contact data and clicks **Save.**

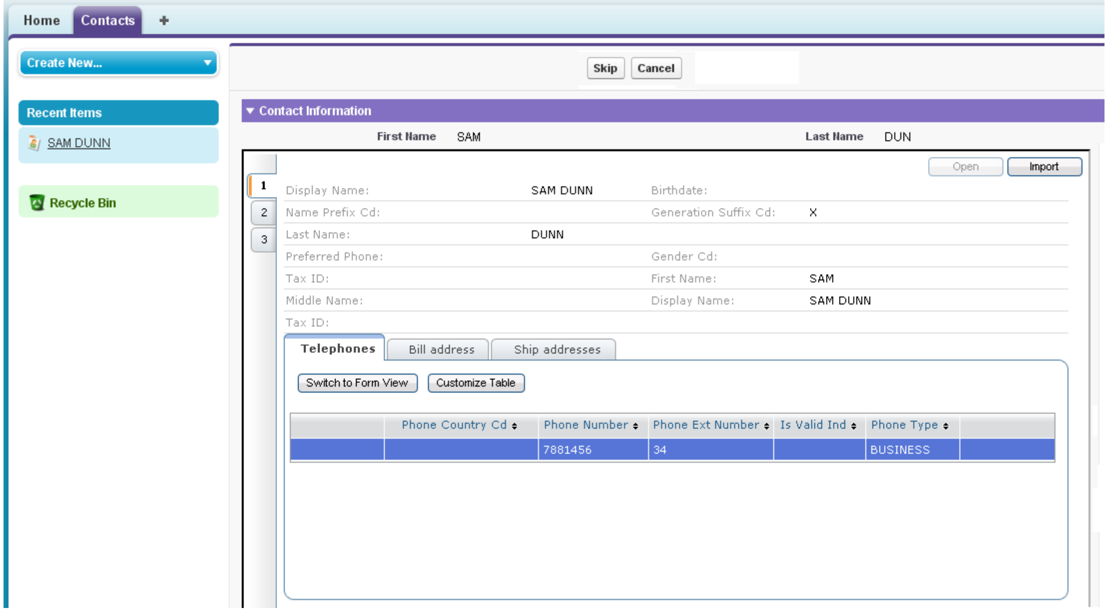


First Name SAM

Last Name DUN

The user is redirected to the second page, which contains Duplicate Prevention control displaying the potential matches found.

2. User can use this page to handle events generated by the control and if the control notifies the page that no matched records are found, then the new Contact is automatically saved. Otherwise, the user can open or import the found matched records, proceed with saving the new Contact or cancel the save operation:



First Name SAM Last Name DUN

Display Name:	SAM DUNN	Birthdate:	
Name Prefix Cd:		Generation Suffix Cd:	X
Last Name:	DUNN		
Preferred Phone:		Gender Cd:	
Tax ID:		First Name:	SAM
Middle Name:		Display Name:	SAM DUNN
Tax ID:			

	Phone Country Cd	Phone Number	Phone Ext Number	Is Valid Ind	Phone Type	
		7881456	34		BUSINESS	

Note: This approach to integrate Duplicate Prevention control into the Salesforce application is used only to demonstrate some aspect of work with the control. You can use other strategies for integration with Salesforce.

To implement this wizard, you must define two pages and a custom controller that keeps data entered by the user, coordinates navigation between pages, and handles Open and Import actions.

The following is a sample Apex code for the controller (which extends the standard Contact controller):

```
public class duplicatePreventionWizardController {
    /*
    field keeps id of the selected matched record, (field is set by JavaScript code
handling ON_OPEN event)
    */
    private String matchedRecordId;

    /*
    next two fields keep values of FIRST_NAME and LAST_NAME fields of the imported
matched record
    (fields are set by JavaScript code handling ON_OPEN event)
    */
    private String importedFirstName;
    private String importedLastName;

    // reference to the standard Contact controller
    private ApexPages.StandardController stdController;

    public duplicatePreventionWizardController(ApexPages.StandardController
stdController) {
        this.stdController = stdController;
    }

    // returns reference to the first wizard's page, used to control navigation
    public PageReference returnToEdit() {
        return Page.NewContact_Step1;
    }

    // returns reference to the second wizard's page, used to control navigation
    public PageReference searchForDuplicates() {
        return Page.NewContact_Step2;
    }

    // generate search criteria string needed to build URL invoking Duplicate Prevention
control
    public String getMatchParametersString() {
        // get data entered by user
        Contact contact = (Contact)stdController.getRecord();
        String firstname = contact.FirstName != null ? contact.FirstName : '';
        String lastname = contact.LastName != null ? contact.LastName : '';
        // construct full name used for matching as FirstName + LastName
        String fullName = firstname + ' ' + lastname;
        // values of IDC parameters should be double encoded
        fullName = doubleEncode(fullName);
        // search criteria should be passed in format 'mc.<match column name>:<match
column value>' (Customer_Name is name of match column configured in Hub)
        String matchParameters = 'mc.Customer_Name:' + fullName;
        return matchParameters;
    }

    // function implements double encoding for IDC parameters
    private String doubleEncode(String str) {
        return EncodingUtil.urlEncode(EncodingUtil.urlEncode(str, 'UTF-8'),
'UTF-8');
    }

    // open selected matched record
    public PageReference openMatch() {
        try {
            // try to find Contact using id passed in ON_OPEN event and redirect to the
Contact view page
            Contact matchedContact = [select id from Contact where id
= :matchedRecordId];
            PageReference matchViewPage = new
ApexPages.StandardController(matchedContact).view();
            matchViewPage.setRedirect(true);
            return matchViewPage;
        } catch (Exception e) {
          ApexPages.Message message = new ApexPages.Message(ApexPages.Severity.FATAL,
'Failed to load object with id ' + matchedRecordId);
          ApexPages.addMessage(message);
          return null;
        }
    }

    // import selected matched record
    public PageReference importObject() {
        try {
            // try to create Contact using data passed in ON_IMPORT event and redirect
to the Contact view page
            Contact importedContact  = new Contact(FirstName = importedFirstName,
LastName = importedLastName);
            insert importedContact;
            PageReference importedViewPage = new
ApexPages.StandardController(importedContact).view();
            importedViewPage.setRedirect(true);
            return importedViewPage;
        } catch (Exception e) {
          ApexPages.Message message = new ApexPages.Message(ApexPages.Severity.FATAL,
'Failed to save object');
          ApexPages.addMessage(message);
          return null;
        }
    }

    // getters and setters required to access controller's attributes using <apex:param>
tag

    public String getMatchedRecordId() {
        return matchedRecordId;
        }

        public void setMatchedRecordId(String matchedRecordId) {
            this.matchedRecordId = matchedRecordId;
        }

        public void setImportedFirstName(String value) {
            importedFirstName = value;
        }

        public String getImportedFirstName() {
            return importedFirstName;
        }

        public void setImportedLastName(String value) {
            importedLastName = value;
        }

        public String getImportedLastName() {
            return importedLastName;
        }

    }
```

The following code defines the first wizard page and contains input fields required to enter new Contact record. To simplify the example, only fields required to perform search for duplicates (contact's first name and last name) are used.

```
<!-- page NewContact_Step1 (uses custom duplicatePreventionWizardController) -->
<apex:page standardController="Contact" extensions="duplicatePreventionWizardController">
<apex:form >
    <apex:pageBlock >
        <apex:pageBlockButtons >
            <!-- Save button invokes 'searchForDuplicates' action, it redirects user to
the second page -->
            <apex:commandButton action="{!searchForDuplicates}" value="Save"/>
            <!-- Cancel button invokes standard 'cancel' action -->
            <apex:commandButton action="{!cancel}" value="Cancel" immediate="true"/>
        </apex:pageBlockButtons>
        <!-- fields to input Contact FirstName ane LastName -->
        <apex:pageBlockSection title="Contact Information">
            <apex:inputField value="{!contact.firstName}"/>
            <apex:inputField value="{!contact.lastName}"/>
        </apex:pageBlockSection>
    </apex:pageBlock>
</apex:form>
</apex:page>
```

The following code defines the second wizard page. It displays the Contact data entered by the user in read-only view and the found potential duplicates. This page has two additional buttons:

Skip

Skips this page and proceeds with save.

Cancel

Returns to the first page.

The URL invoking duplicate prevention control is generated dynamically using Contact data entered by user. The JavaScript code handling Duplicate Prevention events extracts data passed in events and invokes corresponding actions defined in the controller:

```
<!-- page NewContact_Step2 (uses custom duplicatePreventionWizardController) -->
<apex:page standardController="Contact" extensions="duplicatePreventionWizardController">

<!-- include postmessage JavaScript plugin  -->
<script type="text/javascript" src="http://postmessage.freebaseapps.com/
postmessage.js"></script>
```

```
<apex:form id="mainForm">
    <!-- panel displaying error messages -->
    <apex:outputPanel id="messages">
        <apex:messages style="font-weight:bold; color:red;"/>
    </apex:outputPanel>

    <script>
    // function handles events generated by Duplicate Prevention control
    function handleEvent(data) {
        switch(data['action']) {
        case 'ON_LOAD':
            if (data['duplicatesFound'] == 'false') {
                // if duplicates are not found proceed with save
                continueSave();
            }
            break;
        case 'ON_OPEN':
            // try to find and open Contact using id passed in event
            openMatchedRecord(data['id']);
            break;
        case 'ON_IMPORT':
            // convert JSON string passed in event into JavaScript object
            var record = eval('(' + data['record'] + ')');
            // try to save and open Contact using data from Hub passed in event
            importObject(record.Customer.FirstName, record.Customer.LastName);
            break;
        }
    }

    // register function handleEvent as handler for Duplicate Prevention events
    function bindHandler() {
        pm.bind("idd_pmc_event", function(data) {
            handleEvent(data);
        });
    }

    bindHandler();
    </script>

    <!-- JavaScript functions invoking methods defined in controller, used by code
handling events -->

    <apex:actionFunction name="openMatchedRecord" action="{!openMatch}"
reRender="messages">
        <apex:param name="matchedRecordId" assignTo="{!matchedRecordId}" value=""/>
    </apex:actionFunction>

    <apex:actionFunction name="importObject" action="{!importObject}"
reRender="messages">
        <apex:param name="firstName" assignTo="{!importedFirstName}" value=""/>
        <apex:param name="lastName" assignTo="{!importedLastName}" value=""/>
    </apex:actionFunction>

    <apex:actionFunction name="continueSave" action="{!save}" reRender="mainForm"/>

    <apex:pageBlock id="mainPanel">
        <apex:pageBlockButtons >
                               <!-- Skip button invokes standard save action -->
            <apex:commandButton action="{!save}" value="Skip" immediate="true"/>
           <!-- Cancel button redirects user to the first page -->
            <apex:commandButton action="{!returnToEdit}" value="Cancel"
immediate="true"/>
        </apex:pageBlockButtons>

        <apex:pageBlockSection title="Contact Information">
            <apex:outputField value="{!contact.firstName}"/>
            <apex:outputField value="{!contact.lastName}"/>
        </apex:pageBlockSection>

        <!-- IFRAME embedding Duplicate Prevention control, URL is constructed using
string generated by controller -->
```

```
        <iframe src="http://host:port/bdd/bdc/dp/sa:Customer,{!matchParametersString}/
proactive_match/component.jsf"  style="width:100%;height:500px"></iframe>

    </apex:pageBlock>
</apex:form>
</apex:page>
```

To enable JavaScript messaging in Internet Explorer 7, Duplicate Prevention control requires the parentUrl parameter. You must pass the value of this parameter to the Duplicate Prevention control. To achieve this code of NewContact_Step2 page, embedding component using HTML tag <iframe> can be changed for Internet Explorer 7 to dynamically detect current URL and insert it as a parameter into the URL invoking the control as shown:

```
<!-- IFRAME embedding Duplicate Prevention control, URL is constructed using string
generated by controller and dynamically detected URL of the current page-->
<script>
var parentUrl = encodeURIComponent(encodeURIComponent(document.location.href));
var url = 'http://host:port/bdd/bdc/dp/sa:Customer,' + '{!matchParametersString}' +
',parentUrl:' + parentUrl + '/proactive_match/component.jsf';
document.writeln('<iframe src="' + url + '"  style="width:100%;height:500px"></iframe>');
</scrip>
```

To configure Salesforce to use the implemented wizard for new Contact creation:

1. Navigate to **Setup** > **Customize** > **Contacts** > **Button** > **Links**.
2. Click the **Edit** link for the New action.
3. Select the first wizard's page, **NewContact_Step1** to override the value of the URL.

Related Topics:

- "JavaScript Messaging in Internet Explorer 7" on page 21

Accessing IDC Components Bound to Different IDD Configurations

You cannot access more than one IDC component bound to different IDD configurations from the same browser session.

Typically, different browser windows create different sessions, while different browser tabs may share the same session. This behavior depends on the browser configuration. If you need to simultaneously access IDC components that are bound to different IDD applications, then you must access them from different browser sessions.