How-To Library

Informatica

MDM Hub Java User Exits

Abstract

The MDM Hub user exits are based on Java code. This article describes the Java user exits, explains how to implement Java user exits, and describes best practices to follow when you implement user exits.

Supported Versions

- Multidomain MDM 10.1 - 10.3

Table of Contents

User Exits Overview

A user exit is custom Java code that you develop that runs at specific points in the batch or SIF API processes to extend the functionality of Informatica MDM Hub. Informatica MDM Hub exits regular MDM Hub processing to run code that users develop.

For example, you can use a post-landing user exit to perform pre-cleansing on addresses before the delta detection process.

You upload all the user exits to Informatica MDM Hub in a single JAR file. You can upload one JAR file to each Operational Reference Store.

Informatica MDM Hub supplies input parameter values when it calls a user exit. Follow the guidelines for implementing user exits to ensure that the user exits do not unnecessarily decrease performance.

You can implement user exits in the following MDM Hub processes:

Stage Process

The stage process moves data from a landing table to a staging table associated with a base object. You can run the following user exits during the stage process:

- Post-landing user exit. Use the post-landing user exit to refine data in a landing table after you populate the landing table through an ETL process.
- Pre-stage user exit. Use a pre-stage user exit to perform custom handling of delta processes.
- Post-stage user exit. Use a post-stage user exit to perform custom processing at the end of a stage job.

Load Process

The load process moves data from a staging table to a base object table. You can run the following user exit during the load process:

- Post-load user exit. Use a post-load user exit to perform custom processing after the load process.

Match Process

The match process identifies base object records that are potential duplicates. You can run the following user exits during the match process:

- Pre-match user exit. Use a pre-match user exit to perform custom processing before the match process.
- Post-match user exit. Use a post-match user exit to perform custom processing on the match table.

Merge Process

The merge process consolidates duplicate base object records into a single master base object record. You run the following user exit during the merge process:

- Post-merge user exit. Use a post-merge user exit to perform custom processing after the merge process.

Unmerge Process

The unmerge process unmerges a single master base object record into the individual base object records that existed before the records were merged. You can run the following user exits during the unmerge process:

- Pre-unmerge user exit. Use a pre-unmerge user exit to perform custom processing before the unmerge process.
- Post-unmerge user exit. Use a post-unmerge user exit to preform custom processing after the unmerge process.

User Exit Processing

Informatica MDM Hub processes the user exits as part of the block, batch, or SIF API transaction, as applicable. If the user exit generates an exception, Informatica MDM Hub rolls back the processing for the block, batch, or SIF API call. The SIF APIs and batch processes call the same user exits.

When user exits run during a batch process, the user exits run before or after the batch job runs, with the exception of the post-load and post-merge user exits. The post-load and post-merge user exits run after Informatica MDM Hub processes each block. You configure the block size in the cmxserver.properties file when you configure the process servers.

User Exit JAR Files

You can develop custom user exit classes based on the user exit interface classes. Implement custom user exit classes in a separate JAR file and then upload the JAR file to the Informatica MDM Hub to register the user exits.

The Informatica MDM Hub installation includes a JAR file called `mdm-ue.jar` found in `<`*`infamdm_install_dir`*`>\hub\server\lib`. The file `mdm-ue.jar` contains Java interface classes for each user exit.

Use the MDM Hub Console to upload the user exit JAR file with custom code to the Informatica MDM Hub. You can upload one user exit JAR file for each Operational Reference Store. If you want to change the implementation of a user exit in Informatica MDM Hub, first remove the user exits from the Informatica MDM Hub. Then upload a JAR file that contains the user exits with the latest implementation.

Implementing the User Exit JAR File

To implement the user exit JAR file, create custom Java code, and then package the code in a JAR file.

1. Create custom user exit classes.
2. Run the following Java command to package the implemented custom user exit classes in a JAR file:

```
<java_project_folder>\bin>jar -cf  <user_specified_JAR_file_name >  .\com\userexit
\*.class
```

Uploading User Exits to the MDM Hub

To upload user exits to an Operational Reference Store, use the user object registry in the MDM Hub Console.

1. Verify that you are connected to the Operational Reference Store that you want to upload user exits to.
2. In the **Utilities** workbench, select **User Object Registry**.
3. Acquire a write lock.
4. Select **User Exits** in the navigation pane.
5. Click the **Add** button.
6. From the **Add User Exit** window, select **Browse**.
7. From the **Open** window, browse to the JAR file that contains the user exits. Select **Open**.
8. From the **Add User Exit** window, optionally enter a description, and then select **Ok**.

   The user exits appear in the user exit table in the properties pane.

Removing User Exits from the MDM Hub

To delete all user exits from the Informatica MDM Hub, remove the user exits from the user object registry. You cannot remove individual user exits.

1. In the **Utilities** workbench, select **User Object Registry**.
2. Acquire a write lock.
3. Select **User Exits** in the navigation pane.
4. Select the user exit table in the properties pane, and then click the **Remove** button. Select **Ok**.

UserExitContext Class

The UserExitContext class contains the parameters that Informatica MDM Hub to the user exits.

The UserExitContext class passes the following parameters to the user exits:

batchJobRowid

Job ID for the batch job. The UserExitContext class passes the BatchJobRowid parameter to all user exits during batch processes.

connection

Database connection that the MDM Hub process uses.

stagingTableName

Source table for the load job. The UserExitContext class passes the stagingTableName parameter during the load and stage process.

tableName

Name of the table that the MDM Hub process uses.

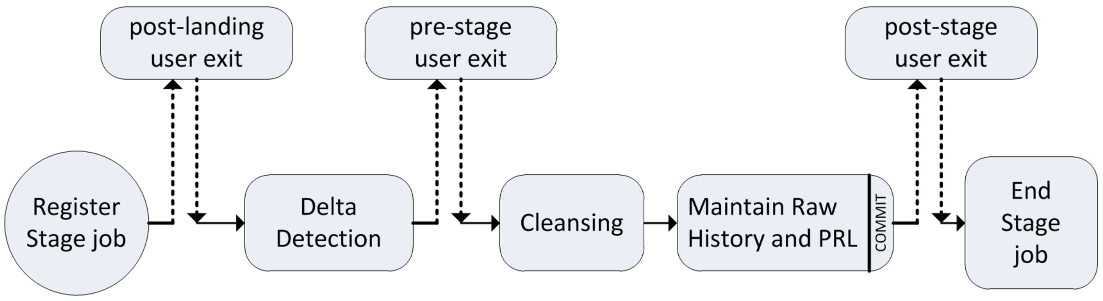
The following code shows the UserExitContext class:

```
package com.informatica.mdm.userexit;

import java.sql.Connection;

/**
 * Represents the hub context that is passed to the user exit interfaces.
 * This is a placeholder for data that is applicable to all user exits.
 *
 */
public class UserExitContext {
    String jobRowid;
    String tablName;
    String stagingTablName;
    Connection conn;

    /**
     * See the corresponding {@link #setBatchJobRowid(String) setter} method for details.
     *
     * @return the rowid of the batch job
     */
    public String getBatchJobRowid() {
        return this.jobRowid;
    }

    /**
     * See the corresponding {@link #setTableName(String) setter} method for details.
     *
     * @return the name of the table used in the hub process
     */
    public String getTableName() {
        return this.tablName;
    }

    /**
     * See the corresponding {@link #setDBConnection(String) setter} method for details.
     *
     * @return the database connection used in the hub process
     */
    public Connection getDBConnection() {
        return this.conn;
    }

    /**
     * Set the rowid of the batch job in the context. This is applicable to batch jobs only.
     *
     * @param batchJobRowid the rowid of the batch job
     */
    public void setBatchJobRowid(String batchJobRowid) {
        this.jobRowid = batchJobRowid;
    }

    /**
     * Set the name of the table used in the hub process.
     *
     * @param tableName the name of the table
     */
    public void setTableName(String tableName) {
        this.tablName = tableName;
    }

    /**
     * See the corresponding {@link #setStagingTableName(String) setter} method for details.
     *
     * @return the name of the staging table used in the hub load and stage process
     */
    public String getStagingTableName() {
        return this.stagingTablName;
    }

    /**
     * Set the name of the staging table used in the context. This is applicable to load and
stage process.
     *
     * @param stagingTableName the name of the staging table
     */
    public void setStagingTableName(String stagingTableName) {
        this.stagingTablName = stagingTableName;
    }

    /**
     * Set the database connection in the context.
     *
     * @param connection the database connection used in the hub process
     */
    public void setDBConnection(Connection connection) {
        this.conn = connection;
    }

}
```

Stage Process User Exits

The stage process can call the post-landing, pre-stage, and post-stage user exits.

The following figure shows the stage process and the user exits that the stage process can call:



The user exits run within the stage process in the following sequence:

1. Informatica MDM Hub registers the stage job.
2. The post-landing user exit runs.
3. Informatica MDM Hub performs delta detection.
4. The pre-stage user exit runs.
5. Informatica MDM Hub performs data cleansing.
6. Informatica MDM Hub populates the stage table.
7. If you enable audit trails, Informatica MDM Hub populates the raw table.
8. Informatica MDM Hub commits the stage table changes.
9. The post-stage user exit runs.
10. The stage job ends.

Post-landing User Exit

Informatica MDM Hub calls the post-landing user exit after Informatica MDM Hub registers the stage job.

Use the post-landing user exit to refine data in a landing table after you populate the landing table through an ETL process. You can use the post-landing user exit to perform custom processing on the landing table before delta detection. For example, you might perform hard delete detection, replace control characters with printable characters, or perform pre-cleansing processing on addresses.

Post-landing User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-landing user exit.

Interface name

The post-landing user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostLandingUserExit
```

Methods

The post-landing user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String stagingTableName, String
landingTableName,
     String previousLandingTableName) throws Exception;
```

Parameters

The post-landing user exit uses the following parameters:

landingTableName

Source table for the stage job.

previousLandingTableName

The previous landing table name that contains a copy of the source data mapped to the staging table from the previous time the stage job ran.

stagingTableName

Target table for the stage job.

userExitContext

Passes parameters to the user exit.

Pre-stage User Exit

Informatica MDM Hub calls the pre-stage user exit before it loads data into a staging table.

Use a pre-stage user exit to perform custom handling of delta processes. You might use a pre-stage user exit to determine whether delta volumes exceed predefined allowable limits. For example, you might use the user exit to stop the stage process if the number of deltas from the source system is greater than 500,000.

Pre-stage User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the pre-stage user exit.

Interface Name

The pre-stage user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PreStageUserExit
```

Methods

The pre-stage user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String stagingTableName, String
landingTableName,
     String deltaTableName) throws Exception;
```

Parameters

The pre-stage user exit uses the following parameters:

deltaTableName

The delta table name. The delta table contains the records that Informatica MDM Hub identifies as deltas.

landingTableName

Source table for the stage job.

stagingTableName

Target table for the stage job.

userExitContext

Passes parameters to the user exit.

Post-stage User Exit

Informatica MDM Hub calls the post-stage user exit after Informatica MDM Hub loads data into a staging table.

Use a post-stage user exit to perform custom processing at the end of a stage job. You might use a post-stage user exit to process rejected records from the stage job. For example, you might configure the user exit to delete records that Informatica MDM Hub rejects for known, noncritical conditions.

Post-stage User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-stage user exit.

Interface Name

The post-stage user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostStageUserExit
```

Methods

The post-stage user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String stagingTableName, String 
landingTableName,
     String previousLandingTableName) throws Exception;
```

Parameters

The post-stage user exit uses the following parameters:

landingTableName

Source table for the stage job.

previousLandingTableName

The previous landing table name that contains the copy of the source data mapped to the staging table from the previous time the stage job ran.

stagingTableName

Target table for the stage job.

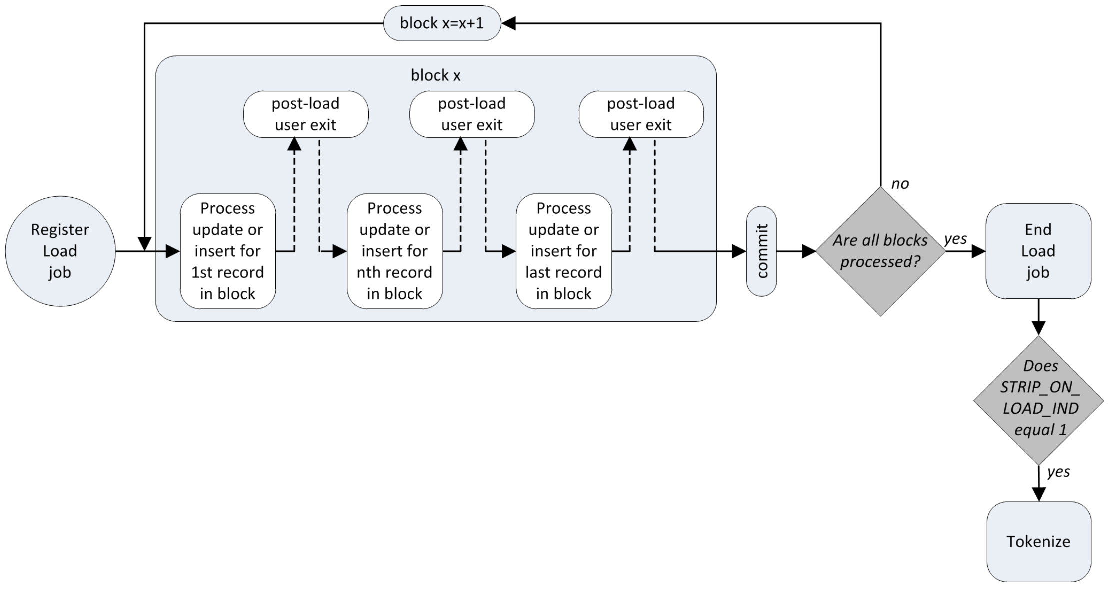
userExitContext

Passes parameters to the user exit.

Load Process User Exits

The load process can call the post-load user exit.

The post-load user exit runs after each block in processed instead of at the end of the batch process. The following figure shows the load process and the user exits that the load process can call:



block x=x+1
block x
post-load user exit
post-load user exit
post-load user exit
Register Load job
Process update or insert for 1st record in block
Process update or insert for nth record in block
Process update or insert for last record in block
commit
no
Are all blocks processed?
yes
End Load job
Does STRIP_ON_LOAD_IND equal 1
yes
Tokenize

The post-load user exit runs within the load process in the following sequence:

1. Informatica MDM Hub registers the load job.
2. Informatica MDM Hub updates or inserts the first record of the first block.
3. The post-load user exit runs.
4. Informatica MDM Hub repeats steps 2 and 3 for the remaining records in the block.
5. Informatica MDM Hub commits the changes.
6. If Informatica MDM Hub has more blocks to load, the process returns to step 2.
7. The load job ends after Informatica MDM Hub loads all blocks.
8. If strip_on_load_ind equals 1, the tokenize job generates match tokens required to perform fuzzy matching.

Post-load User Exit

Informatica MDM Hub calls the post-load user exit after the load process.

Use a post-load user exit to perform custom processing after the load process.

Post-load User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-load user exit.

Interface Name

The post-load user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostLoadUserExit
```

Methods

The post-load user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, ActionType actionType,
     Map<String, Object>baseObjectDataMap, Map<String, Object> xrefDataMap,
     List<Map<String, Object>> xrefDataMapList) throws Exception;
```

Parameters

The post-load user exit uses the following parameters:

actionType

Indicates whether the load job inserted the record or updated the record.

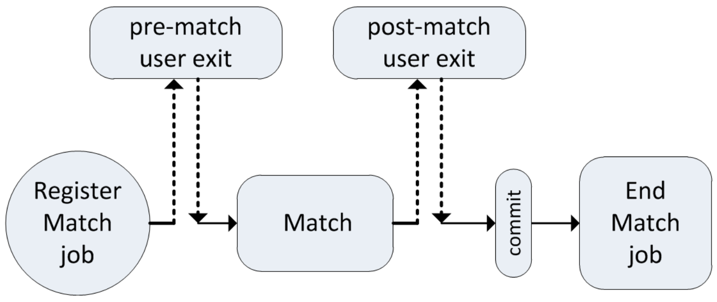
userExitContext

Passes parameters to the user exit.

Match Process User Exits

The match process can call the pre-match and post-match user exits.

The following figure shows the match process and the user exits the match process can call:



The user exits run within the match process in the following sequence:

1. Informatica MDM Hub registers the match job.
2. The pre-match user exit runs.
3. Informatica MDM Hub runs the match job.
4. The post-match user exit runs.
5. Informatica MDM Hub commits the match changes.
6. The match job ends.

Pre-match User Exit

Informatica MDM Hub calls the pre-match user exit before the match process.

Use a pre-match user exit to perform custom processing before the match process.

Pre-match User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the pre-match user exit.

Interface Name

The pre-match user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PreMatchUserExit
```

Methods

The pre-match user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String matchSetName) throws Exception;
```

Parameters

The pre-match user exit uses the following parameters:

matchSetName

The name of the match rule set.

userExitContext

Passes parameters to the user exit.

Post-match User Exit

Informatica MDM Hub calls the post-match user exit after the match process.

Use a post-match user exit to perform custom processing on the match table. For example, you might use a post-match user exit to manipulate matches in the match queue.

Post-match User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-match user exit.

Interface

The post-match user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostMatchUserExit
```

Methods

The post-match user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String matchSetName) throws Exception;
```

Parameters

The post-match user exit uses the following parameters:

matchSetName

The name of the match rule set that Informatica MDM Hub used to find the match.

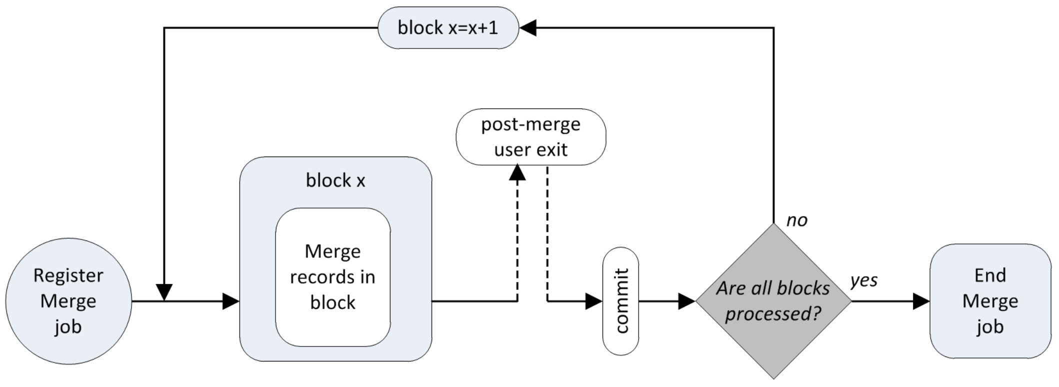
userExitContext

Passes parameters to the user exit.

Merge Process User Exits

The merge process can call the post-merge user exit.

The following figure shows the merge process and the user exits the merge process can call:



The user exits run within the merge process in the following sequence:

1. Informatica MDM Hub registers the merge job.
2. Informatica MDM Hub merges the first set of matched records in the first block.
3. The post-merge user exit runs.
4. Informatica MDM Hub commits the changes.
5. Informatica MDM Hub repeats steps 2 through 4 for the remaining records in the block.
6. If Informatica MDM Hub has more blocks to merge, the process returns to step 2.
7. The merge job ends after Informatica MDM Hub processes all blocks.

Post-merge User Exit

Informatica MDM Hub calls the post-merge user exit after the merge process.

Use a post-merge user exit to perform custom processing after the merge process. For example, you might use a post-merge user exit to match and merge child records affected by the match and merge of a parent record.

Post-merge User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-merge user exit.

Interface Name

The post-merge user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostMergeUserExit
```

Methods

The post-merge user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, List<String> baseObjectRowIds) throws
Exception;
```

Parameters

The post-merge user exit uses the following parameters:

baseObjectRowIds

List of base object row IDs involved in the merge. The first entry is the target base object record. The remaining entries in the list are the source base objects that merged into the target.

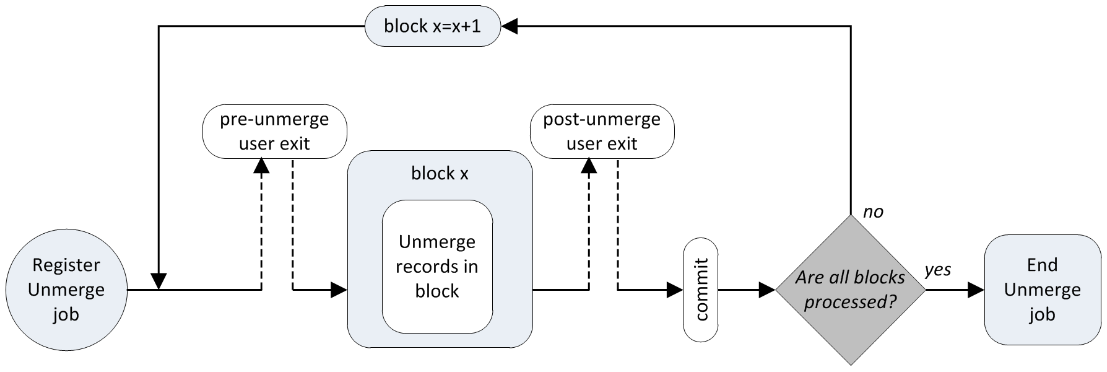
userExitContext

Passes parameters to the user exit.

Unmerge Process User Exits

The unmerge process can call the pre-unmerge and post-unmerge user exits.

The following figure shows the unmerge process and the user exits that the unmerge process can call:



The user exits run within the unmerge process in the following sequence:

1. Informatica MDM Hub registers the unmerge job.
2. The pre-unmerge user exit runs.
3. Informatica MDM Hub unmerges the first record of the first block.
4. The post-unmerge user exit runs.
5. Informatica MDM Hub repeats steps 3 and 4 for the remaining records in the block.
6. Informatica MDM Hub commits the changes.
7. If Informatica MDM Hub has more blocks to unmerge, the process returns to step 3.
8. The unmerge job ends after Informatica MDM Hub processes all blocks.

Pre-unmerge User Exit

Informatica MDM Hub calls the pre-unmerge user exit before the unmerge process.

Use a pre-unmerge user exit to perform custom processing before the unmerge process.

Pre-unmerge User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the pre-unmerge user exit.

Interface Name

The pre-unmerge user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PreUnmergeUserExit
```

Methods

The pre-unmerge user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String rowidSystem,
     String pkeySourceObject) throws Exception;
```

Parameters

The pre-unmerge user exit uses the following parameters:

pkeySourceObject

The primary key of the source object.

rowidSystem

The system row ID of the cross-reference record for the base object that the unmerge processes.

userExitContext

Passes parameters to the user exit.

Post-unmerge User Exit

Informatica MDM Hub calls the post-unmerge user exit after the unmerge process.

Use a post-unmerge user exit to preform custom processing after the unmerge process.

Post-unmerge User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the post-unmerge user exit.

Interface Name

The post-unmerge user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.PostUnmergeUserExit
```

Methods

The post-unmerge user exit uses the following methods:

```
void processUserExit(UserExitContext userExitContext, String rowidObject) throws Exception;
```

Parameters

The post-unmerge user exit uses the following parameters:

rowidObject

The Row ID of the reinstated base object.

userExitContext

Passes parameters to the user exit.

Task Management User Exits

You can use the AssignTasks user exit and the GetAssignableUsersForTask user exit for task management.

AssignTasks User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the AssignTasks user exit.

Interface Name

The AssignTasks user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.AssignTasksUserExit
```

Methods

The AssignTasks user exit uses the following methods:

```
void processAssignTasks(UserExitContext userExitContext, int maxTasks)
```

Parameters

The AssignTasks user exit uses the following parameters:

userExitContext

Passes parameters to the user exit.

maxTasks

Maximum number of tasks to assign to each user.

GetAssignableUsersForTask User Exit Interface

Use the appropriate interface name, methods, and parameters when you implement the GetAssignableUsersForTask user exit.

Interface Name

The GetAssignableUsersForTask user exit uses the following fully qualified interface name:

```
com.informatica.mdm.userexit.GetAssignableUsersForTaskUserExit
```

Methods

The GetAssignableUsersForTask user exit uses the following methods:

```
 public List<String> processGetAssignableUsersForTask(String taskType, String rowidSubjectArea,
UserExitContext userExitContext)
```

Parameters

The GetAssignableUsersForTask user exit uses the following parameters:

rowidSubjectArea

The Row ID of the subject area.

taskType

The type of task.

userExitContext

Passes parameters to the user exit.

Guidelines for Implementing User Exits

Implement user exits in a way that does not decrease Informatica MDM Hub performance unnecessarily.

Consider the following guidelines when you implement user exits:

Consider how user exits affects performance.

When you implement user exits, you affect Informatica MDM Hub performance. Consider if you need to use a user exit to accomplish what you want to do, or if you can do the same thing asynchronously without relying on user exits.

For example, when you load data, you might want to create tasks for the pending records. Although it is possible to create tasks with a user exit, it is unnecessary. You create a bottleneck in the process and unnecessarily decrease performance. You can create the tasks later because these tasks are not critical at that point in the process.

Use the database connection to query or update the database.

The Java user exit context provides a database connection. Use this database connection to query the database or update the database through direct JDBC calls or SIF API calls.

Consider how SIF API calls from a user exits affects batch performance.

When you call SIF APIs from a user exit, consider how these calls affect batch processes.

Use SIF APIs when possible.

Do not use the MDM Hub tables directly if you can use an equivalent SIF API.

Do not explicitly commit or roll back changes made in a user exit.

If you use the database connection to make SIF API calls or direct JDBC calls, the calls participate in the same transaction as the user exit caller. The user exit caller maintains the changes made in the transaction. Do not explicitly call commit or rollback on the connection passed to the user exits.

The MDM Hub does not authenticate or authorize user exits.

User exits run with full access to MDM Hub objects. User exits run with full access to MDM Hub Object regardless of the context. You can use the user exit SIF client to call a subset of supported SIF APIs. Only the supported SIF APIs can participate in the same transaction as the user exit caller. You cannot use the user exit SIF client to access unsupported SIF APIs.

Avoid recursive pre-unmerge user exit calls.

To avoid recursive pre-unmerge user exit calls, do not use the Unmerge SIF API with the post-unmerge user exit.

Avoid recursive post-merge user exit calls.

To avoid recursive post-merge user exit calls, do not use the MultiMerge SIF API with the post-merge user exit.

Avoid recursive post-load user exit calls.

To avoid recursive post-load user exit calls, do not use the MultiMerge SIF API with the post-load user exit.

When you use the Put API with the post-load user exit, set the Put API BypassPostLoadUE parameter to `true` to prevent recursive post-load user exit calls.

Author

MDM Documentation Team