precisely

Precisely EnterWorks

EnterWorks 10.4.8 Change Notification Administration Guide

Version 10.4.8

Notices

Table of Contents

Change Notification 5
Overview 5
Architecture 5
Installation and Initial Configuration 6
Integrate Change Notification Trigger 7
Registering the Change Notification Trigger 7
Updating an Existing Trigger for a Repository 7
Managing Multi-Language Attributes in Existing Trigger 9
Registering Change Notification as a PreSave Validation Rule 10
Activating Debug Logging 10
CN_Registry Repository 10
Summary Tab 11
Conditions Tab 12
Attributes Tab 13
Update Tab 16
Transformation Tab 18
Log Tab 20
Translation Tab 23
Email Tab 24
Workflow Tab 27
CN_Log Repository 29
Reporting Using Scheduled Exports 30
Transformation Extension 33
Overview 33
CN_Transformation_Repository 33
Summary Tab 33
Target Tab 35
Condition Tab 37
Future Tab 41
SQL Expressions 41
Example Transformations 42
Rollup Attribute 42

Cascade Attribute....47
Language Translation Extension....50
Overview....50
Change Notification Translation Registry....51
Summary Tab....51
Attributes Tab....53
Change Notification Translation Queue....54
Change Notification Translation Log Repository....56
Translation Request File....58
Translation Request Scheduled Export....60
Translation Request Log Update Scheduled Import....63
Translation Queue Cleanup Scheduled Export....65
Translation Queue Update Scheduled Import....65
Translation Queue Update Response Template....69
Translation Language Update Import Templates....70
SQL-Based Reports....70
Reports Using the CN_Reports Repository and Scheduled Export....71
CN_Reports Repository....72
Summary Tab....72
Attributes Tab....74
Report Scheduled Export....75
Launch Report Scheduled Export....75
Report Scheduled Export....76
Change Report Processing....76

Change Notification

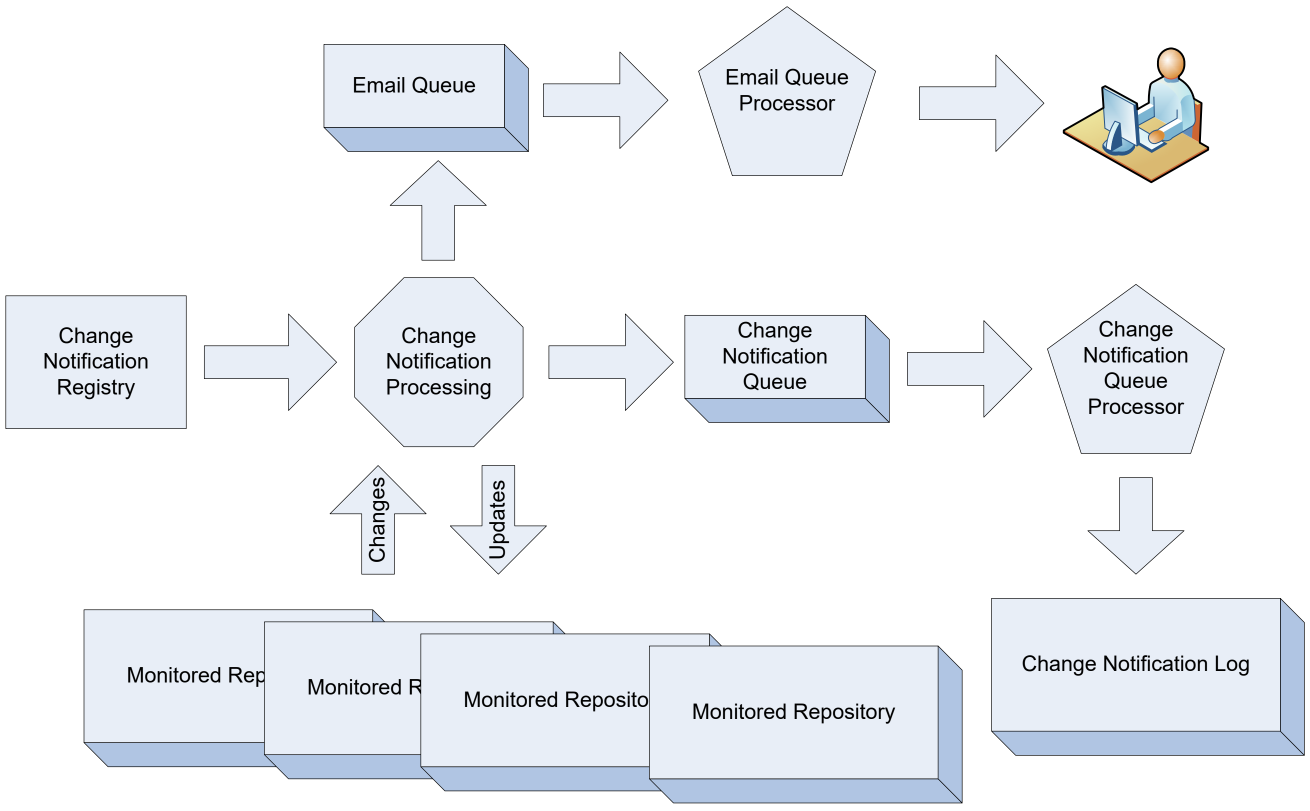
Overview

The Change Notification module provides the ability to track changes made to designated attributes in designated repositories and take one or more of the specified notification actions:

- Update designated attributes with the specified values in the changed record
- Log change details to the Change Notification Log
- Send e-mail to the designated addressee(s)
- Launch work items in designated workflows
- Facilitate sending Language Translation Requests to an external language translating service.
- Update designated attributes with the specified values in target repository records associated to the changed record
- Facilitate transformations to designated attributes based on values and conditions defined in a Change Notification Transformations repository
- Create, update, delete repository records based on the changed record

Architecture

The following diagram illustrates the Change Notification architecture:



The Change Notification Registry identifies which repositories and attributes are to be monitored and what actions are to be taken when changes are detected. The Change Notification Processing is invoked any time a repository record is created, modified or deleted. It uses the details in the Change Notification Registry to determine if any action needs to be taken.

- **Email** - notifications are placed into the Email Queue (table). The Email Queue Process (EPX workflow) retrieves the queued messages and packages them into e-mails based on common To address and Subject text.

- **Log** - The Change Notification Queue is used to log the details of the changes in order to minimize the performance impact of the original operation that invoked the Change Notification Processing. Since a single update to a repository record could result in multiple updates to the change log (one for each changed attribute), inserts into a SQL table are significantly faster than inserts into an Enable repository. A background (EPX) process monitors the Change Notification Queue for new records and moves them to the Change Notification Log repository(ies). In high-volume change scenarios (e.g., large file imports or mass edits), there may be a backlog of pending changes in the Change Notification Queue that takes a while to move to the Change Notification Log repository(ies).

- **Update** – the specified updates are made to the record that triggered the change notification

- **Work Item** – work item requests are placed in the Work Item Queue (table). The Change Notification Process (EPX Workflow) retrieves the queued requests and launches work items for each on based on the details specified in the queued request

- **Language Translation** – identifies an entry in the language translation registry (CN_Translation_Registry) that defines the language translation configuration for a repository.

- **Transform** – identifies transformations that are to be conditionally made to attributes in the record that triggered the change notification or to attributes in a record that is linked to the record that triggered the change notification. The conditions would be based off values in the changed record or the record linked to the changed record. Transformations can also be configured to create or delete records in a repository.

Installation and Initial Configuration

The following sections detail the steps necessary to install and initially configure the Change Notification processing. The necessary files, database tables and stored procedures, and data model (profiles, repositories, and code sets) should all be pre-installed with the Services Framework installation. The only configuration step required is to register the Change Notification trigger with each repository in the sharedConfig.properties file or on the Edit -> Trigger Properties operation in the Classic UI.

Integrate Change Notification Trigger

The Change Notification processing requires the Change Notification Trigger be invoked (directly or indirectly) for each repository for which Change Notification processing is to be enabled. If a repository already has a trigger defined for it, that trigger must be updated to call the Change Notification trigger as a final step before saving any updates to the changed record.

Registering the Change Notification Trigger

If a repository to be monitored does not currently have a trigger defined for it, the Change Notification Trigger must be registered by performing the following steps:

1. Edit each sharedConfig.properties file for each Enable Server Tomcat and JBoss services.
2. Make sure the property **allow.external.event.handler** is set to "true".
3. Add an entry to the property **external.event.handlers**. Entries must be separated by commas with no spaces. The name chosen will be used in the subsequent properties (referenced in this document as "<repo>"). For example, if the PIM_Item_Staging repository is to be monitored, an appropriate name would be "itemStaging" and the property: **external.event.<repo>.target.repository** would be defined as: **external.event.itemStaging.target.repository=PIM_Item_Staging**
4. Add the following properties below the **external.event.handlers** property:

   ```
   external.event.<repo>.type=0
   external.event.<repo>.sync=true
   external.event.<repo>.passInDBSessionInd=true
   external.event.<repo>.classname=com.enterworks.services.changenot
   ification.ChangeNotificationTrigger
   external.event.<repo>.target.repository=<actualRepositoryName>
   ```

   Note: In general, all triggers should be defines as synchronous (external.event.<repo>.sync=true). If there is a need for a Tomcat trigger to be asynchronous, set this flag to false. The JBoss triggers MUST be defined as synchronous.

5. Alternatively, edit the Trigger Properties for the repository in the Classic UI:
   a. Check the **Trigger Enabled** checkbox.
   b. Set the **External Handler Class** to: com.enterworks.services.changenotification.ChangeNotificationTrigger
   c. Set the **External Handler PassIn DBSession Indicator** to: True
   d. Set the **External Handler Sync** to True:
   e. Click **Save**.
6. Repeat the above steps for each repository to be monitored.

Updating an Existing Trigger for a Repository

If a repository to be monitored already has a trigger registered for it, that trigger should be updated to call the Change Notification trigger to keep the overhead processing to a minimum. A second trigger can be defined for the repository, but this will result possibly two new versions of the record being created for each change.

To integrate the Change Notification Trigger into an existing trigger, perform the following steps:

1. Edit the sharedConfig.properties file for each Enable Server Tomcat and JBoss service.
2. Make sure the existing trigger is configured to pass the DB session to the trigger by the **external.event.<repo>.passInDBSessionInd** property being set to true. If it is not defined or is set to false, change it to true. This will also require the trigger doWork() method to be updated.
3. Edit the source file for the existing trigger.
4. Make sure the doWork() method has the following signature:

```
        public void doWork(String repositoryName, long itemId, int action, ItemObject
currentItem, List diffList, org.hibernate.Session hibernateSession, Connection conn)
           throws Exception
```

5. Define a BsessionEpimHelper instance with the following code:

```
        CustomRepositoryItem itemData =
        (CustomRepositoryItem)currentItem;

        // Change the session to admin user
        Bsession bSession = EpimCustomHelper.getAdminBsession();
        itemData.setCurSession(bSession);
        HashMap<String, String> itemChanges = new HashMap<String,
        String>();

        // Create session helper using existing session and DB connection
        BsessionEpimHelper bsessionEpimHelper =
        BsessionEpimHelper.getInstance(bSession, conn, hibernateSession);

        // Retrieves all changes including to non-default languages in
        multi-language attributes:
        HashMap diffMap = ChangeNotificationTrigger.getCurDiffMap(
        diffList, itemData );
```

   Note: The current session should be changed to the admin user to ensure attribute security does not interfere with trigger operations.

6. After the existing New/Modify processing, add the following statement:

```
        // Honor any change notifications

        itemChanges.putAll(ChangeNotificationTrigger.processChangeNotific
ations(repositoryName, itemId, itemData, action, bsessionEpimHelper,
diffMap, itemChanges, null));
```

```
// Save any changes
if (itemChanges.size() > 0) {
    if (debugEnabled) appLogger.severe(
"AutoCalculateTrigger.doWork: saving changes: " + itemChanges);
    itemData.updateItemDataByFormatAttrHash(itemChanges, false);
}
```

7. Recompile the trigger source file
8. Repeat the above steps for each existing trigger.

Managing Multi-Language Attributes in Existing Trigger

If the existing trigger deals with non-default multi-language values, the getCurDiffMap() includes them using the following naming convention: F_<formatAttrId>_<langExt> where <formatAttrId> is the internal ID for the attribute and <langExt> is the language extension. This is the form the attribute name must be in order to update a non-default multi-language attribute. The Change Notification class has a method that provides lookup maps to facilitate dealing with multi-language attributes:

```
ChaneNotificationTrigger.getMultiLanguageAttributesForRepository(DBQue
ry dbQuery, String repositoryName, ArrayList<String>
multiLanguageSystemNames, HashMap<String,String>
extensionForSystemName, HashMap<String,String> attributeForSystemName,
HashSet<String>languageExtensions, HashMap<String,String>
systemNameForAttributeWithExtension);
```

- **dbQuery** – the caller must pass in a valid DBQuery object
- **repositoryName** – name of the repository being processed by the trigger
- **multiLanguageSystemNames** – empty ArrayList to be populated with the list of the system names (e.g., F_100000_es) for all multi-language attributes.
- **extensionForSystemName** – empty HashMap to be populated with a lookup for the extension for each system name (e.g, es for F_100000_es)
- **attributeForSystemName** – empty HashMap to be populated with a lookup for the attribute name for each system name (e.g., Brand Name for F_100000_es)
- **languageExtensions** – empty HashSet to be populated with a list of only the active multi-languages (except the default)

- **systemNameForAttributeWithExtension** – empty HashMap to be populated with a lookup for the system name for each user-friendly attribute name with language extension (e.g., F_100000_es for Brand Name_es)

Registering Change Notification as a PreSave Validation Rule

The Change Notification functionality can be invoked using a PreSave Validation rule by performing the following steps:

1. Edit the profile of the repository for which Change Notification processing is to be enabled.
2. Add a validation rule to an attribute in the profile.
3. Set the Type to Pre-Save Callout
4. Set the Class Path to:
   com.enterworks.services.changenotification.ChangeNotificationPreSaveValidation
5. Edit the Rule Properties for the repository.
6. Assign the new validation rule.
7. Clear the Data Cache

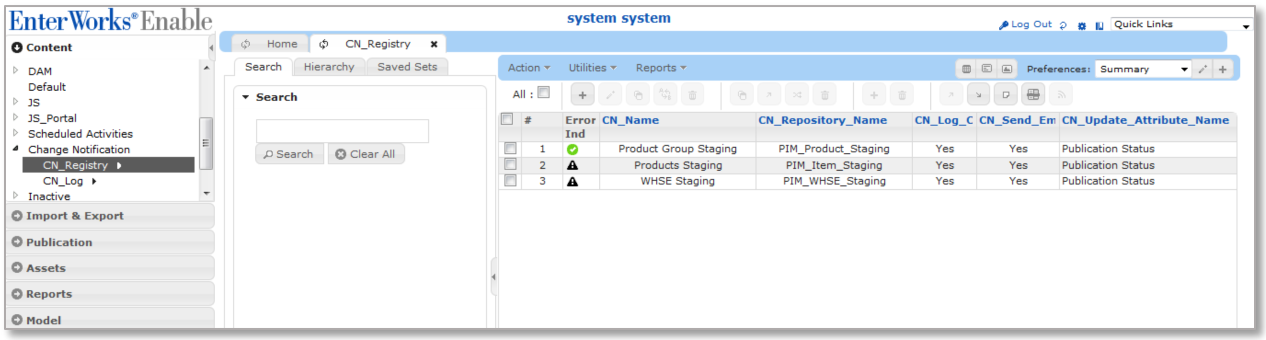
Activating Debug Logging

Debug logging for the Change Notification functions is controlled by the property **debug.ChangeNotificationEnabled** being set to **true** in the **Enterworks.properties** file. Services must be restarted for any property file changes to go into effect.

WARNING: Having the debug logging enabled may have a detrimental impact to performance. It is recommended to rely on the debug logging while developing a Change Notification-based solution, and then disabling the logging after development and sufficient testing has been completed.

CN_Registry Repository

The CN_Registry repository defines the registered change notifications and their actions. Each record in the repository represents changes being monitored and what actions are to be taken. Typically, there will be only one registration for each monitored repository but there may be situations where having more than one registration is warranted. For example, if different actions are needed depending upon which attributes are changed, each different set of attributes and their actions would be specified in separate CN_Registry records.



system system

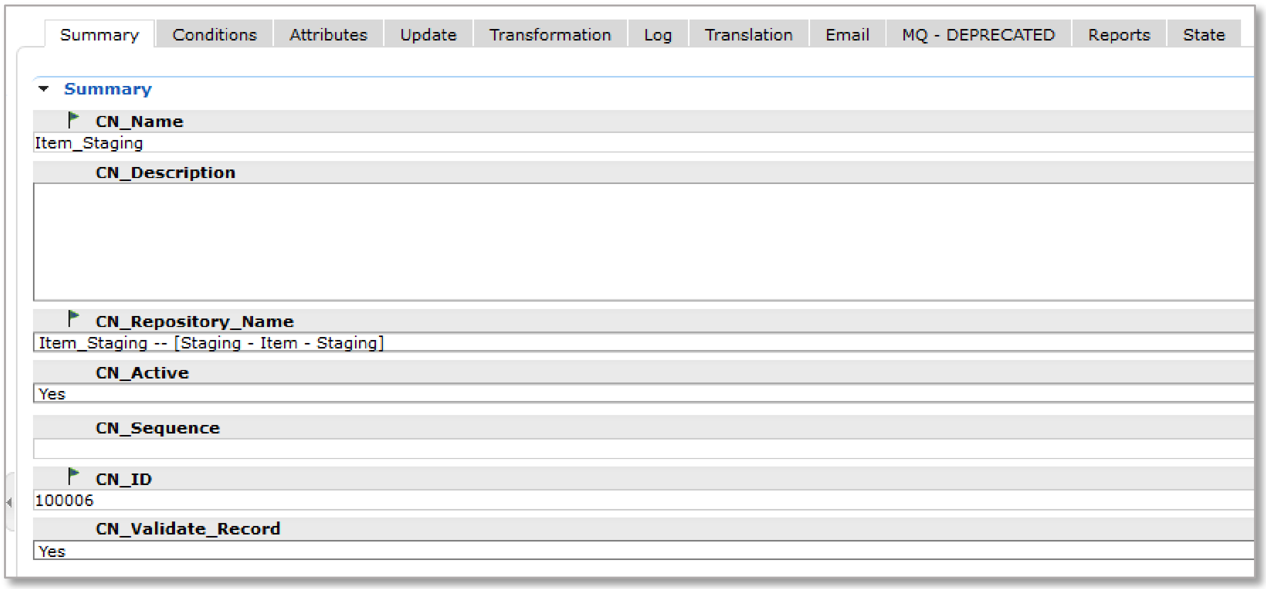
Preferences: Summary

#	Error Ind	CN_Name	CN_Repository_Name	CN_Log_C	CN_Send_Em	CN_Update_Attribute_Name
1	✓	Product Group Staging	PIM_Product_Staging	Yes	Yes	Publication Status
2	⚠	Products Staging	PIM_Item_Staging	Yes	Yes	Publication Status
3	⚠	WHSE Staging	PIM_WHSE_Staging	Yes	Yes	Publication Status

The following sections provides detailed information on each attribute in this repository, grouped by tabs.

Summary Tab

The Summary tab identifies the basic information for the registry entry, including registration name, repository being monitored, whether the registry entry is active and its processing order (when multiple registry entries are defined for the same repository):



CN_Name: Item_Staging
CN_Description:
CN_Repository_Name: Item_Staging -- [Staging - Item - Staging]
CN_Active: Yes
CN_Sequence:
CN_ID: 100006
CN_Validate_Record: Yes

CN_Name - is a logical name for the registry entry. If there is only one change notification registered for a repository, the name will likely be set to match the name of the repository. If a repository has multiple registrations, the name should reflect the nature of the registered notification. For example, if the Item Staging repository has a set of attributes that require manager review and another set of attributes that only need to be logged, two CN_Registry records would be created with the first having a name of "Item Staging Review" and the second having the name "Item Staging Log Only"

CN_Description - Optional description for the registry record. It is strongly recommended that a description be specified and kept up-to-date when changes are made to the record.

CN_Repository_Name - Name of the repository to which the registered change applies. The Drop-down selection list shows all repositories defined in the system, but only those that have been configured for the Change Notification trigger or pre-save validation rule will be processed.

CN_Active - The CN_Registry record is active if Yes and ignored if No.

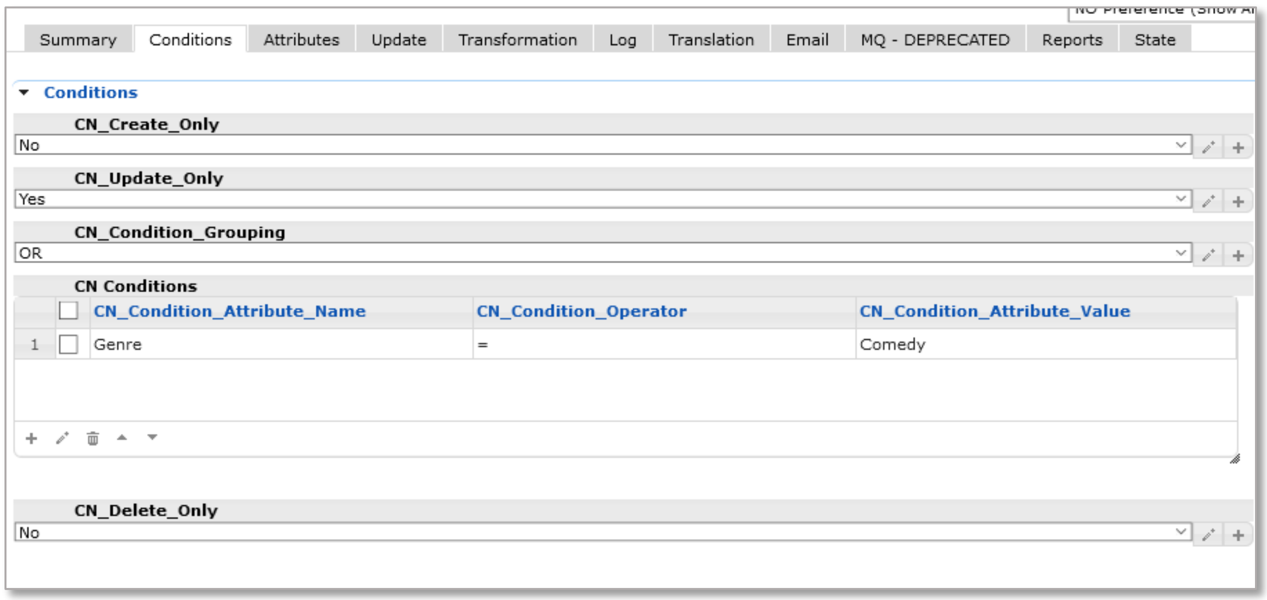
CN_ID - Internal ID for the registered change. This value will be specified in the CN_Registry_ID for any record in the CN_Log repository that originated from this registered change.

CN_Sequence - Determines the order registry records for the same repository are processed. The order may be significant. For example, if one registered change updates an attribute that needs to be logged by another registered change, the update change needs to have a lower sequence number than the log change.

CN_Validate_Record – Causes the trigger record to be validated one time (in cases where there are multiple CN_Registry records for the same repository) before performing any Change Notification processing if Yes. The intent of this option is to ensure the trigger record has been validated in case a Transformation rule needs to obtain the records validation status. If this is not needed in any Transformation rules, then this attribute should be set to No.

Conditions Tab

The Conditions tab specifies any conditions that must be met before any action is taken for a repository record.



CN_Create_Only: No

CN_Update_Only: Yes

CN_Condition_Grouping: OR

CN Conditions

	CN_Condition_Attribute_Name	CN_Condition_Operator	CN_Condition_Attribute_Value
1	Genre	=	Comedy

CN_Delete_Only: No

CN_Create_Only – Specifies that the change is only triggered on new records if Yes, otherwise it is triggered on updates and on new records (if CN_Update_Only is not Yes). Both CN_Create_Only and CN_Update_Only cannot be set to Yes.

CN_Update_Only – Specifies that the change is only triggered on updates if Yes, otherwise it is triggered on updates (if CN_Create_Only is not Yes) and on new records. Both CN_Create_Only and CN_Update_Only cannot be set to Yes.

CN_Condition_Grouping - Specifies whether the multiple conditions are grouped by AND, meaning all conditions must be met, or OR meaning any condition must be met.

CN_Condition_Attribute_Name - Name of the attribute for which a condition is defined. The condition must be evaluated as "true" in order for the actions in this registry record to be applied.

Conditions can only be defined on the default language – other languages cannot be specified. Generally, the conditions determine whether the particular record is subject to a registered change notification. This would typically be an attribute (or attributes) containing a value for the default language only.

CN_Condition_Operator - Specifies the condition operator for the CN_Condition_Attribute_Name. Must be one of the following:

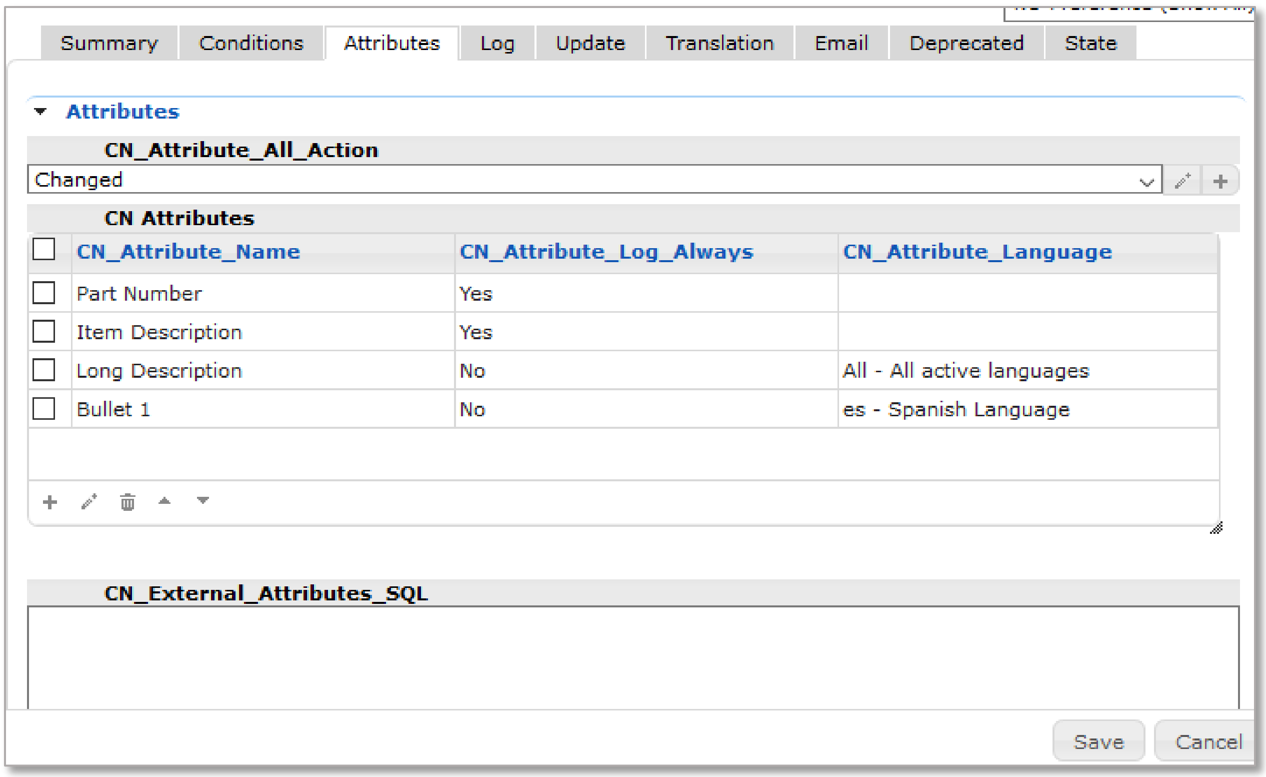
- = - The value of an attribute exactly matches the specified value
- <> - The value of an attribute does not exactly match the specified value.
- Begin With – the beginning of the value of an attribute matches the designated value.
- End With – the ending of the value of an attribute matches the designated value
- IN – the value of an attribute matches one of the values in the designated delimited list
- Is Empty – the designated attribute has no value
- Is Not Empty – the designated attribute as any value

CN_Condition_Attribute_Value - Value to be compared to the actual value of the attribute identified by CN_Condition_Attribute_Name when the CN_Condition_Operator is =, <>, Begin With, End With or IN. Multiple values can be specified for the IN operator by separating them with commas (no spaces).

CN_Delete_Only – specifies that the change is only triggered on deletes if Yes. Requires CN_Create_Only and CN_Update_Only to be set to No.

Attributes Tab

The Attributes tab identifies the attributes in the repository being monitored for changes or to be logged any time there is a change:



Attributes

CN_Attribute_All_Action: Changed

CN Attributes

CN_Attribute_Name	CN_Attribute_Log_Always	CN_Attribute_Language
Part Number	Yes	
Item Description	Yes	
Long Description	No	All - All active languages
Bullet 1	No	es - Spanish Language

CN_External_Attributes_SQL

This list of attributes excludes the multi-language attributes to be processed for language translation – these attributes are defined in the named CN_Translation_Registry record.

CN_Attribute_All_Action - Specifies the action to be taken on all attributes in the repository. This eliminates the need to list every attribute in the CN_Attribute_Name attribute. If a new attribute is added to the repository, it will automatically be included:

- **Changed** – take action on a change being made to any attribute in the repository but only log attributes that have actually changed. If specific attributes should always be logged, they must be listed in the CN_Attribute_Name list with the corresponding CN_Attribute_Log_Always set to Yes.
- **Changed with Multi-Language** – take action on a changed being made to any attribute in the repository including the non-default Multi-Language extensions.
- **Log Always** – log the values of each attribute in the repository. If this option is set, at least one attribute must be specified in the CN_Attribute_Name list to trigger the notification processing.
- **Log Always with Multi-Language** – log the values of each attribute in the repository including all active language extensions for Multi-Language attributes. If this option is set, at least one attribute must be specified in the CN_Attribute_Name list to trigger the notification processing.

CN_Attribute_Name - The attribute CN_Attribute_Name identifies each attribute to be monitored or logged. Changes to attributes not in this list will be ignored unless CN_Attribute_All_Action is set. Required if CN_Attribute_All_Action is not set.

CN_Attribute_Log_Always - Specifies whether the corresponding attribute in the CN_Attribute_Name list is always logged (if other attributes in the list have changed) or only when there is a change. If the log always attribute has not changed, the Old Value and New Value will be the same. If CN_Attribute_All_Action is set, attributes listed here will override the All Action setting. For example, if CN_Attribute_All_Action is set to "Changed", attributes will only be logged if changed, unless listed here with CN_Attribute_Log_Always being set to "Yes".

CN_Attribute_Language – Optionally defines the language extension for each attribute being monitored (if it is not the default language) for multi-language attributes. For example, to monitor changes to the Spanish version of Long Description, the CN_Attribute_Language would be set to "es". The "All" option indicates that all active languages are to apply. Unless the CN_Attribute_All_Action attribute is set to "Changed with Multi-Language" or "Log Always with Multi-Language", the only way to trigger or log non-default language values is by explicitly listing the attribute and language extension (with All indicating all active languages should be included):

CN_External_Attributes_SQL – Optional SQL Statement that returns a single row of additional data (from other repositories) that is connected to the changed record. Each column in the SELECT statement is captured and logged as additional attributes. This allows for contextual information to be recorded at the time of the original record was changed since retrieving the linked values at report time may be different from when the original change was logged.

The SQL statement may contain any valid SQL for a SELECT query. Data from the changed record can be included in the statement through named references (denoted by the attribute name surrounded by double pipe-characters). These references are resolved before the SQL statement is executed. If the value needs to be treated as a character value in the query, it must be surrounded by single quotes.

For example, if the Enable data model has a Product repository linked to an Item repository by an alphanumeric ID and changes to Item records need to record the Product Group Name (from the Product record), the following SQL can be used to retrieve that information:

```
SELECT Product_Group_Name FROM _PIM_Product_Staging WHERE
Product_ID = '||Product ID||'
```

All external attributes will show the same value for the Old and New values in the log.

The external attributes are populated prior to the processing of the conditions on the Conditions tab. This means an external attribute can be used to determine whether the CN_Registry is processed. It also means the external attribute SQL is always executed so its performance/efficiency needs to be a primary consideration.

If the name of a column in the query matches the name of an attribute in the monitored repository, the external attribute should be aliased to something unique. For example, if the Product repository and the Item repository both have a "Publication Status" attribute and both need to be logged, the SQL statement must alias the attribute:

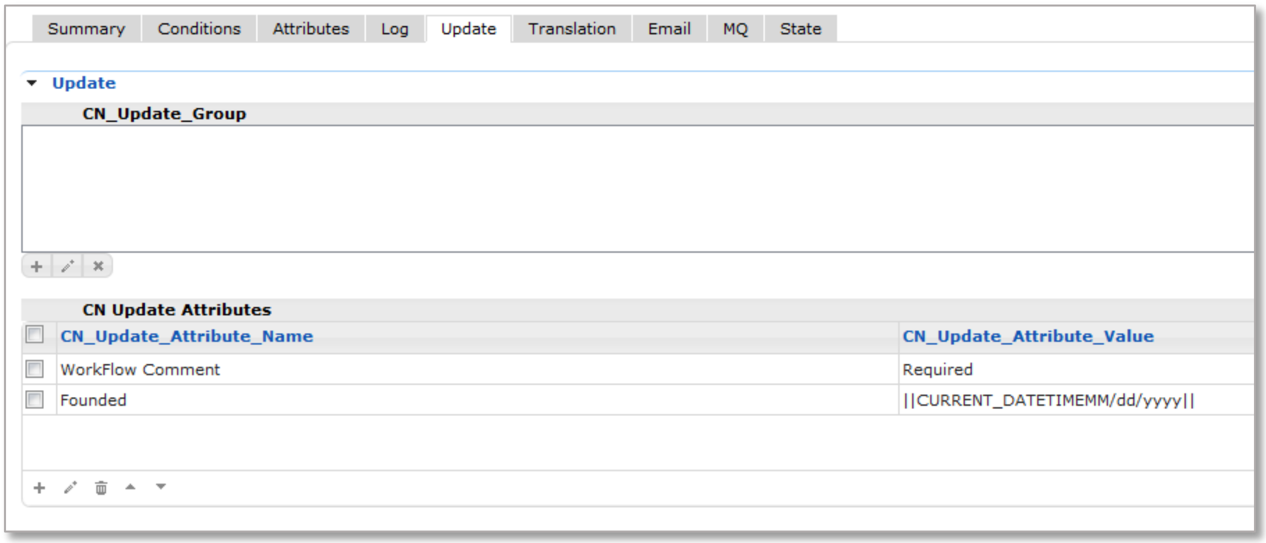
```
SELECT Publication_Status as [Product Publication Status],
Product_Group_Name FROM PIM_Product_Staging WHERE Product_ID =
'||Product ID||'
```

If the SELECT statement may return more than one row, only the first record retrieved will be used – the subsequent records will be ignored. If the values of the field(s) being retrieved may be different, the SQL must be structured such that the value from the desired record is returned first in the results.

In addition to referencing any attribute in the monitored repository, the SQL may also contain the date and time stamp of when the change was made by specifying the reserved reference ||CURRENT_DATETIME||. This is resolved to the current date and time in the format yyyyMMddHHmmssSS. Alternative formats can be specified by using the Java class SimpleDateFormat's notation. For example, to only include the date in mm/dd/yyyy format, the reference must be ||CURRENT_DATETIMEMM/dd/yyyy||.

Update Tab

The Update tab specifies any attributes in the changed record that are to be updated to specified values:



CN_Update_Attribute_Name	CN_Update_Attribute_Value
WorkFlow Comment	Required
Founded	||CURRENT_DATETIMEMM/dd/yyyy||

CN_Update_Group – List of Groups to which an update would be applicable if the user who triggered the change is a member of one or more the specified groups. If the user is not a member of any group, the update does not occur. If no groups are specified, then the update occurs for all users.

CN_Update_Attribute_Name - List of attributes in the changed record to be updated on change. This must exactly match the name (including case) in the repository.

CN_Update_Attribute_Value - The value to change the corresponding attribute. The value can be a literal (e.g., setting the attribute "Review Record" to "Yes") a field reference (surrounded by double pipe-characters) or a combination of the two.

If the value of the non-external attribute has been modified by a previous CN_Registry update or an implementation-specific trigger that was processed before invoking the Change Notification Processing, the updated value will be used. For example, if three CN_Registry records are defined for the same repository and the first one updates the attribute "Record Status", references to that attribute in subsequent CN_Registry records will use the new value. The final value will be saved to the repository record when all processing has completed.

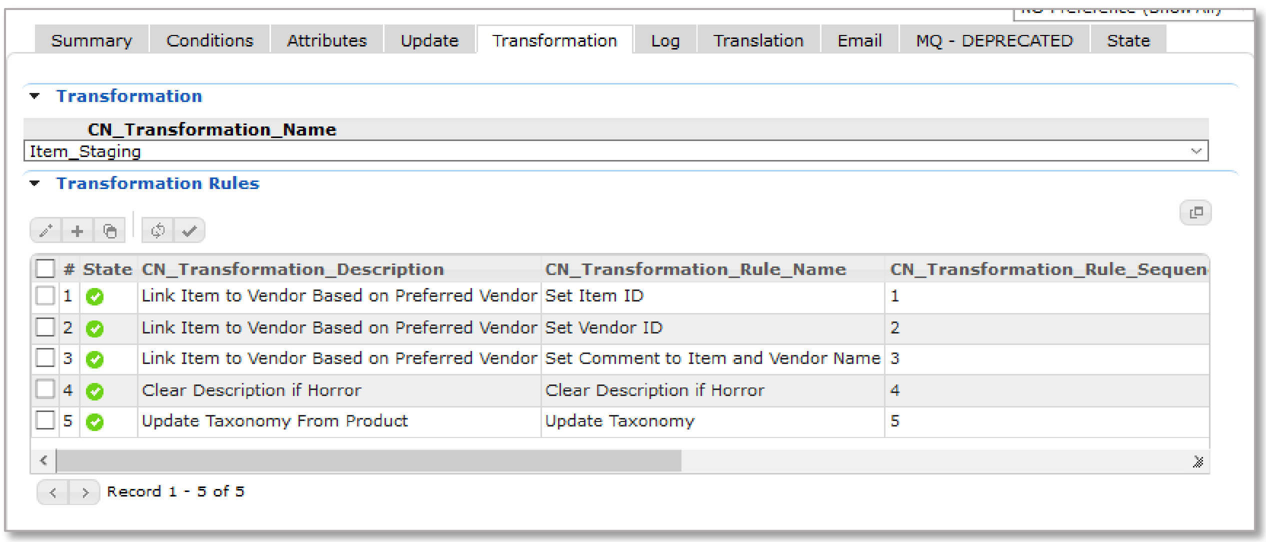
The following field references can be specified.

Reference	Description
||<attributeName>||	Replaced by the value of the specified attribute or external attribute.
||<attributeName>_NEW_VALUE||	Replaced by the new value for the specified attribute.
||<attributeName>_<langExt>_NEW_VALUE||	Replaced by the new value for the specified non-default multi-language attribute for the specified language extension.
||<attributeName>_OLD_VALUE||	Replaced by the old value for the specified attribute
||<attributeName>_<langExt>_OLD_VALUE||	Replaced by the old value for the specified non-default multi-language attribute for the specified language extension.
||CURRENT_DATETIME[<format>]||	Replaced by the current date and/or time. The default format is MMDDYYYYHHmmssSS. Any valid SimpleDtetimeFormat can be specified between "CURRENT_DATETIME" and the ending double-pipe characters. For example, to return just the date with no punctuation: ||CURRENT_DATETIMEyyyyMMdd||
||CHANGED_BY||	Replaced by the login ID of the user who made the change.
||CHANGE_DETAIL_CSV||	
||CHANGE_DETAIL_HTML||	
||CHANGE_DETAIL_PLAIN||	

Reference	Description
||CHANGE_LIST_<delimiter>||	Builds a delimited list using the specified delimiter character of the names of the attributes that have changed. This reference should only be used in an Update list. It is assumed the target attribute will contain a delimited list of attribute names that were previously changed since the last reset-event. A pipe delimiter must be specified as "PIPE" and the comma delimiter must be specified as "COMMA". If the target attribute already has a value, the new attributes are added to the list if not already present. The attribute should be cleared when the appropriate event occurs, such as a Delta Export of changed records. For example, to build a tilde-delimited list, specify: ||CHANGE_LIST_~||. To build a comma-delimited list, specify: ||CHANGE_LIST_COMMA||. To build a pipe-delimited list, specify: ||CHANGE_LIST_PIPE||.
||CHANGE_LIST_RESTRICTED_<delimiter>||	Same as CHANGE_LIST_<delimiter> except the list contains the restricted names for the attributes.
||ERROR_LIST_<delimiter>||	Builds a delimited list using the specified delimiter character of the names of the attributes that have validation errors (warning or severe). This reference should only be used in an Update list. This assumes the CN_Registry attribute CN_Validate_Record must be set to Yes, otherwise no attributes will be listed. A pipe delimiter must be specified as "PIPE" and the comma delimiter must be specified as "COMMA". For example, to build a tilde-delimited list, specify: ||ERROR_LIST_~||. To build a comma-delimited list, specify: ||ERROR_LIST_COMMA||. To build a pipe-delimited list, specify: ||ERROR_LIST_PIPE||.
||InternalRecordId||	Returns the internal record ID of the trigger record

Transformation Tab

The Transformation tab identifies the transformations that are to be performed in association with the registered change event.



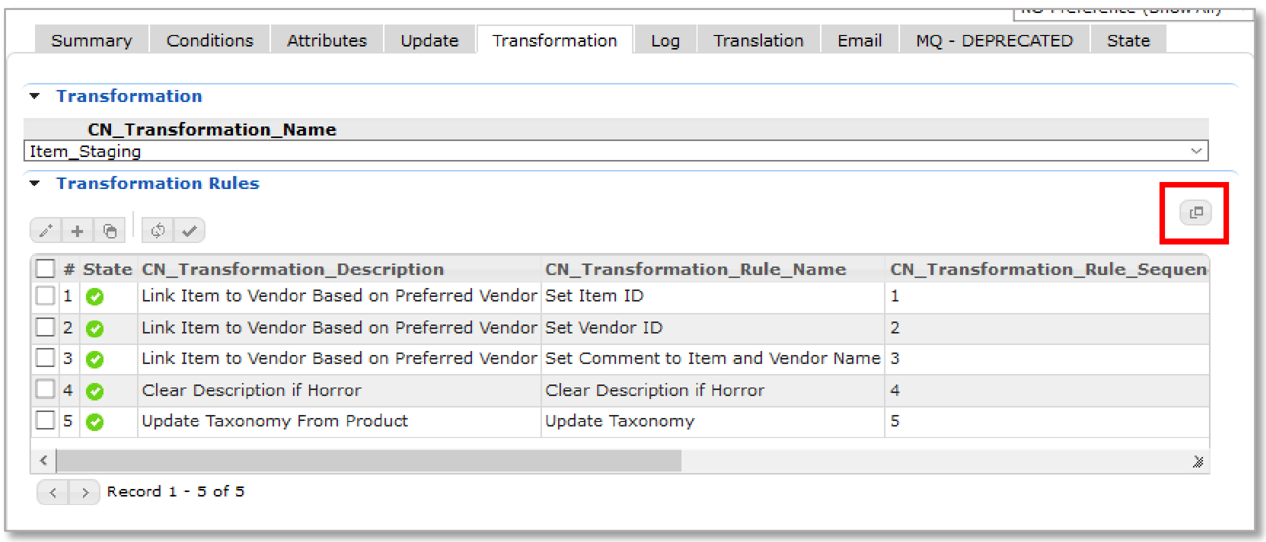
CN_Transformation_Name: Item_Staging

#	State	CN_Transformation_Description	CN_Transformation_Rule_Name	CN_Transformation_Rule_Sequen
1	✓	Link Item to Vendor Based on Preferred Vendor	Set Item ID	1
2	✓	Link Item to Vendor Based on Preferred Vendor	Set Vendor ID	2
3	✓	Link Item to Vendor Based on Preferred Vendor	Set Comment to Item and Vendor Name	3
4	✓	Clear Description if Horror	Clear Description if Horror	4
5	✓	Update Taxonomy From Product	Update Taxonomy	5

CN_Transformation_Name - Identifies the CN_Transformation_Registry entry to be processed. The drop-down selection list will show all defined CN_Transformation_Registry record names, which means the transformation registry records need to be created first. Multiple transformations may be identified on this tab – each named transformation will be processed independent of any other in the same Change_Notification_Registry record. For example, if three transformations are named and the first has no conditions met for performing any transformation, the other two will still be processed. Each named transformation may have a different target repository record that differs from the record that triggered the change event, which merely identifies when the set of transformations are to be performed. If a transformation target is a different repository record, it must be associated to the triggered record (i.e., it must be linked).

When the transformation rules reference the event trigger attributes and those attributes have been updated by CN_Registry Update operations or prior CN_Transformation_Registry rules (for the same event), the updated attribute values will be used. The CN_Registry Update is performed before the Transformation. For example, if the CN_Registry Update changes the value of the attribute "Approval Required" to "Yes", any linked CN_Transformation_Registry rule that references ||Approval Required|| will use the value "Yes".

Transformation Rules - The Transformation Rules list shows all linked transformation rules for the CN_Registry record being edited. These rules are listed in the order of the rule sequence. But since each rule might have more than one condition defined and the list can only sort on one attribute, the conditions may not be in the defined order. To view the rules in the proper order (and to more-easily edit them), click on the **Edit Transformation Rules** button above the list:



Transformation

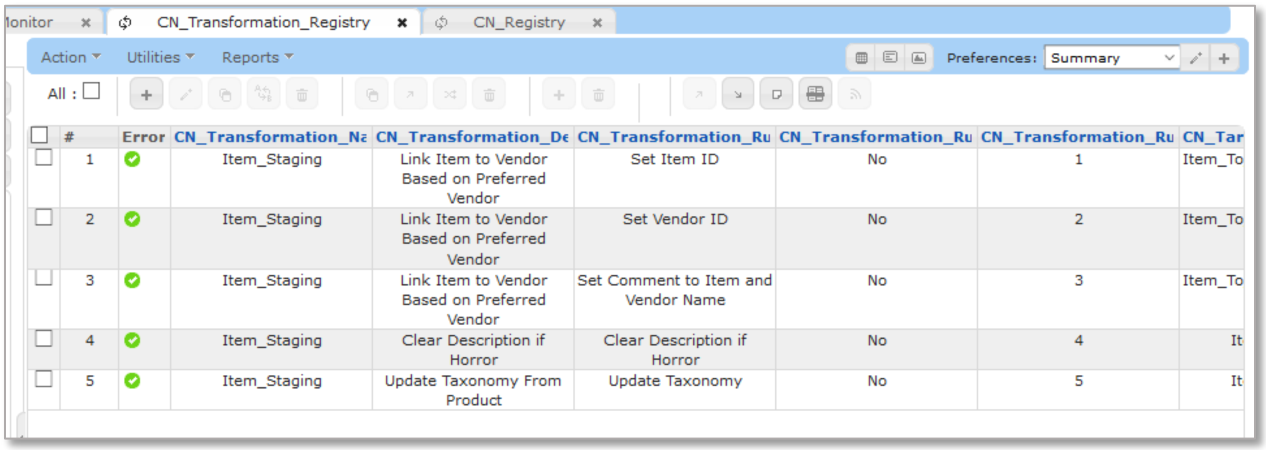
CN_Transformation_Name: Item_Staging

Transformation Rules

#	State	CN_Transformation_Description	CN_Transformation_Rule_Name	CN_Transformation_Rule_Sequen
1	✓	Link Item to Vendor Based on Preferred Vendor	Set Item ID	1
2	✓	Link Item to Vendor Based on Preferred Vendor	Set Vendor ID	2
3	✓	Link Item to Vendor Based on Preferred Vendor	Set Comment to Item and Vendor Name	3
4	✓	Clear Description if Horror	Clear Description if Horror	4
5	✓	Update Taxonomy From Product	Update Taxonomy	5

Record 1 - 5 of 5

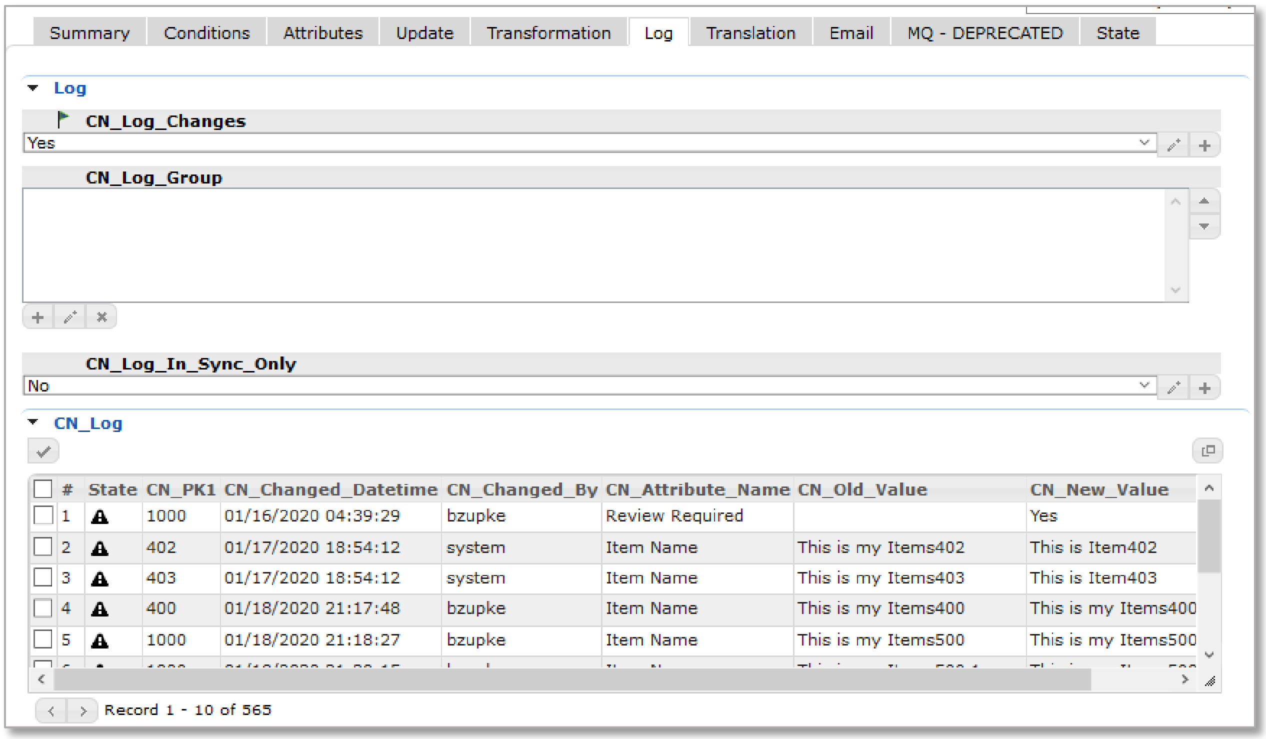
The CN_Transformation_Registry tab will show only the Transformation rules linked to the CN_Registry record being edited:



#	Error	CN_Transformation_Na	CN_Transformation_De	CN_Transformation_Ru	CN_Transformation_Ru	CN_Transformation_Ru	CN_Tar
1	✓	Item_Staging	Link Item to Vendor Based on Preferred Vendor	Set Item ID	No	1	Item_To
2	✓	Item_Staging	Link Item to Vendor Based on Preferred Vendor	Set Vendor ID	No	2	Item_To
3	✓	Item_Staging	Link Item to Vendor Based on Preferred Vendor	Set Comment to Item and Vendor Name	No	3	Item_To
4	✓	Item_Staging	Clear Description if Horror	Clear Description if Horror	No	4	It
5	✓	Item_Staging	Update Taxonomy From Product	Update Taxonomy	No	5	It

Log Tab

The Log tab controls whether changes are logged to the CN_Log repository:



CN_Log_Changes: Yes

CN_Log_Group:

CN_Log_In_Sync_Only: No

#	State	CN_PK1	CN_Changed_Datetime	CN_Changed_By	CN_Attribute_Name	CN_Old_Value	CN_New_Value
1	▲	1000	01/16/2020 04:39:29	bzupke	Review Required		Yes
2	▲	402	01/17/2020 18:54:12	system	Item Name	This is my Items402	This is Item402
3	▲	403	01/17/2020 18:54:12	system	Item Name	This is my Items403	This is Item403
4	▲	400	01/18/2020 21:17:48	bzupke	Item Name	This is my Items400	This is my Items400
5	▲	1000	01/18/2020 21:18:27	bzupke	Item Name	This is my Items500	This is my Items500

Record 1 - 10 of 565

CN_Log_Changes - Specifies changes are to be logged to the CN_Log repository if Yes.

CN_Log_Group - Lists the applicable groups for which the changes are to be logged. If the user is not a member of any of the specified groups, the change is not logged. If left blank, then all changes are logged, regardless of who made the change.

CN_Log_In_Sync_Only – Change events are logged to the CN_Log repository only when the Staging record is in Sync with Production if Yes. This has the effect of only logging the set of changes when the Staging record is promoted to Production. Normally the log messages are placed in the queue when changes are made to the Staging repository records and are soon dequeued and added to the CN_Log repository. Since the Change Notification cannot operate on Production repositories (due to triggers not being invoked during the promotion operation), delaying the dequeue and logging of change events provides a method of logging and reporting changes made to a promoted record. When a Staging record is promoted to production, its Record State will be changed to In Sync. At that point, any queued change events will be logged to the CN_Log repository. The next time a report export runs, it will pick up all of the changes made to the Production record since the last promotion. This assumes that the reporting is run after each promotion.

If a scheduled export is set up as a delta export on the CN_Log repository for events associated with the delayed sync option, all of the changes made to the record (with the final ones in the batch of records reflecting the final values) in the Production repository. The following table visually illustrates the events over a period of time:

Event	Sync State	Change Notification Queue Dequeue Action	Attr A Old Value	Attr A New Value	Attr B Old Value	Attr B New Value	Production Report Attr A Old Value	Production Report Attr A New Value	Production Report Attr B Old Value	Production Report Attr B New Value
Record in Sync with Production	In Sync	Dequeue	one	one	a	a	one	one	a	a
Update Staging	Not In Sync	Hold	one	two	a	a				
Update Staging	Not In Sync	Hold	two	three	a	b				
Update Staging	Not In Sync	Hold	three	four	b	b				
Promote to Production	In Sync	Dequeue					one	four	a	b
Update Staging	Not In Sync	Hold	four	five	b	c				
Update Staging	Not In Sync	Hold	five	six	c	c				
Promote to Production	In Sync	Dequeue					four	six	b	c

As the table shows, once a record in Staging has been modified, it’s sync status indicates not in-sync with Production. Any changes made to the staging record that are queued to the Change Notification log would remain in the queue until the record is promoted. Once the record is promoted, all the queued change events would be dequeued and inserted into the CN_Log repository.

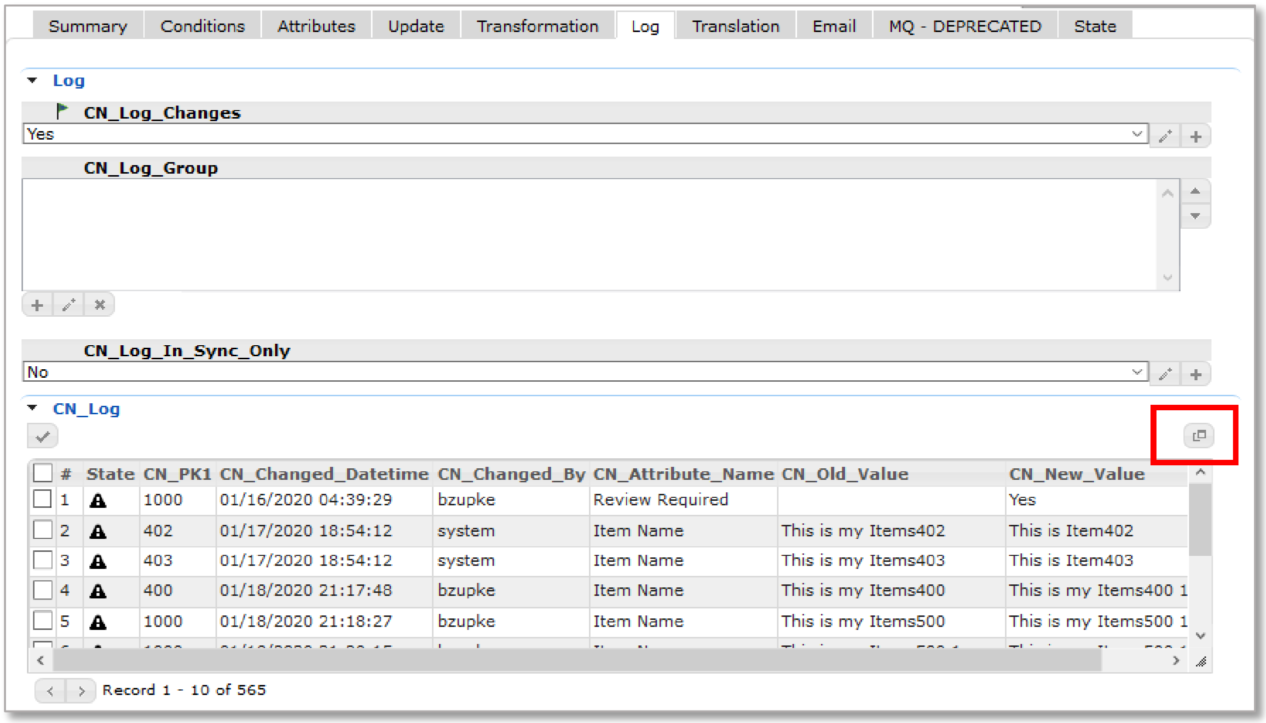
The Change Notification dequeue operation (which is launched every 5 minutes by default) is expected to run frequently enough that there would not be time for a staging record to have subsequent changes after it has been promoted, which would prevent the pending records from being dequeued.

Once the records have been stored in the CN_Log repository, a SQL-based delta export can extract the old value from the oldest new record and the new value from the newest new record for a given attribute to obtain an accurate picture of the changes of the records in Production since the last report was generated.

In order for all changes for a promoted record to be logged, it is essential that the Staging record not be modified before the dequeue operation has been able to retrieve all of the queued events for the record. This is best accomplished by either having the promotion operation run after normal business hours.

If a Staging record is promoted to Production multiple times during a single reporting period, the report will not identify the specific changes that were made for each promotion, only the changes made for all promotions since the last report was generated. If there is a requirement to report the changes for each promotion, the report must be run after each promotion (allowing sufficient time for all queued changes to be dequeued and logged).

CN_Log – Lists the records in the CN_Log repository that were generated for the CN_Registry record being edited. Since the number of records is likely to be quite large, it is not practical to view them in this list. To view the log messages, click the View Log Records button:



CN_Log_Changes: Yes

CN_Log_Group:

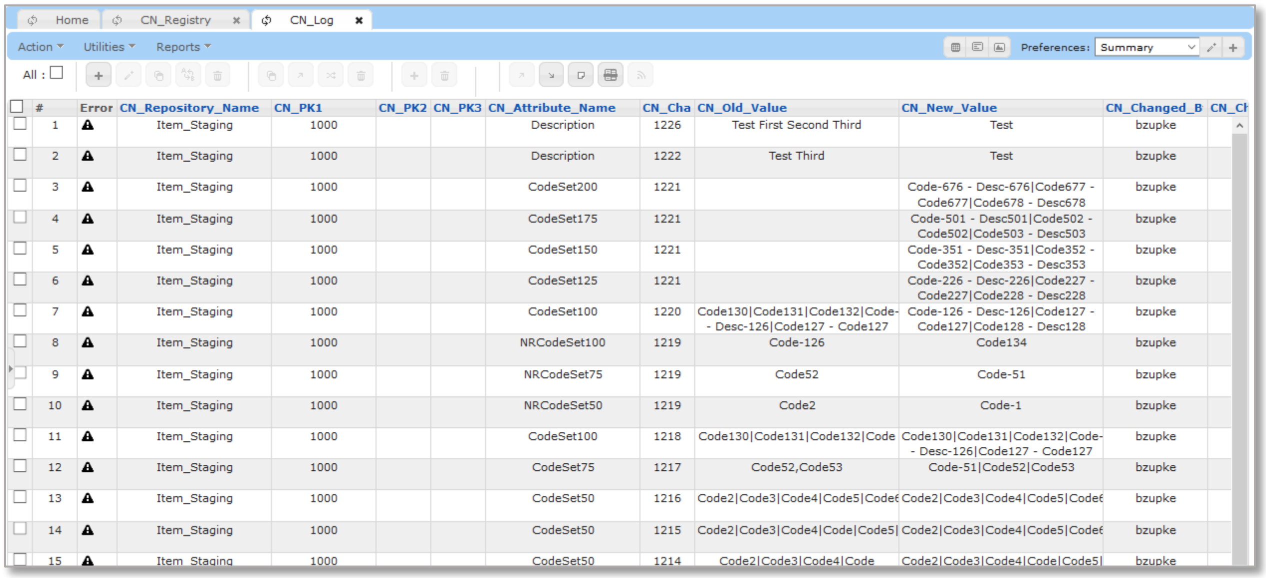
CN_Log_In_Sync_Only: No

CN_Log

#	State	CN_PK1	CN_Changed_Datetime	CN_Changed_By	CN_Attribute_Name	CN_Old_Value	CN_New_Value
1	⚠	1000	01/16/2020 04:39:29	bzupke	Review Required		Yes
2	⚠	402	01/17/2020 18:54:12	system	Item Name	This is my Items402	This is Item402
3	⚠	403	01/17/2020 18:54:12	system	Item Name	This is my Items403	This is Item403
4	⚠	400	01/18/2020 21:17:48	bzupke	Item Name	This is my Items400	This is my Items400 1
5	⚠	1000	01/18/2020 21:18:27	bzupke	Item Name	This is my Items500	This is my Items500 1

Record 1 - 10 of 565

The CN_Log tab will appear, showing only the log entries that were generated for the CN_Registry record being edited:

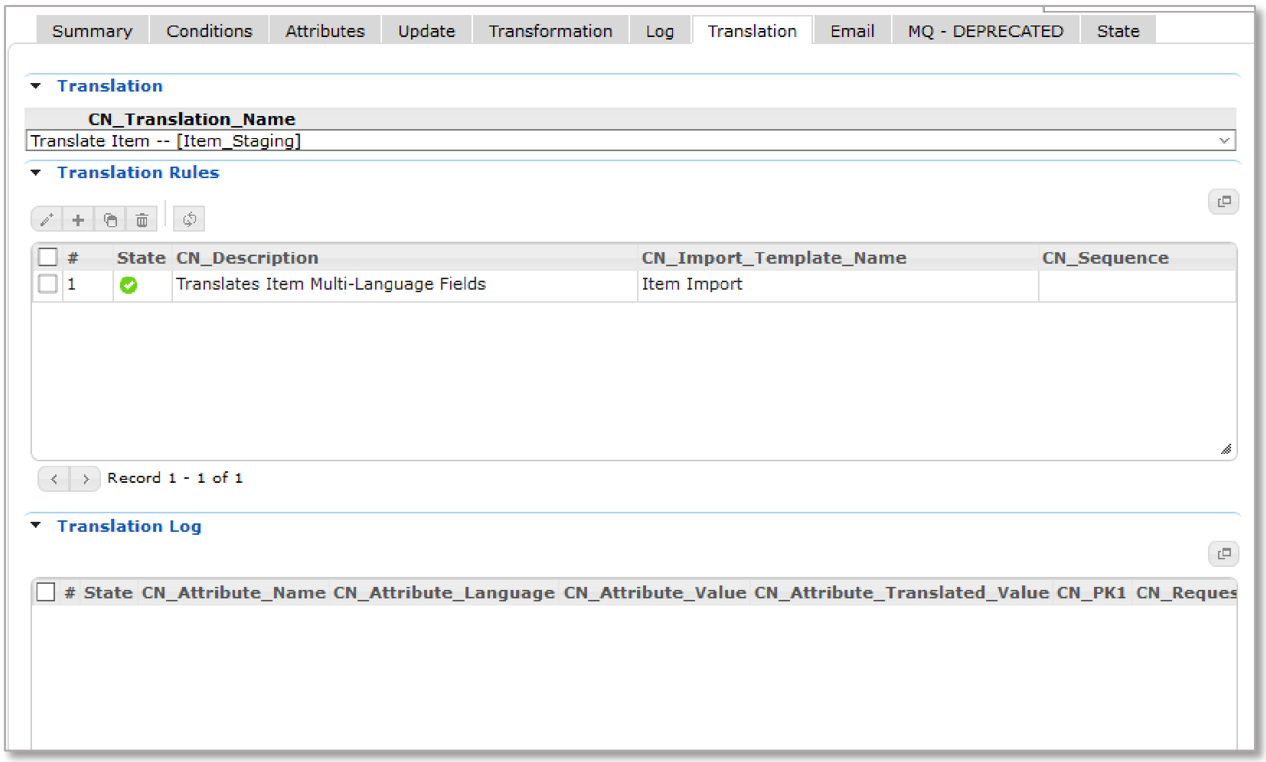


#	Error	CN_Repository_Name	CN_PK1	CN_PK2	CN_PK3	CN_Attribute_Name	CN_Cha	CN_Old_Value	CN_New_Value	CN_Changed_B	CN_Ch
1	⚠	Item_Staging	1000			Description	1226	Test First Second Third	Test	bzupke	
2	⚠	Item_Staging	1000			Description	1222	Test Third	Test	bzupke	
3	⚠	Item_Staging	1000			CodeSet200	1221		Code-676 - Desc-676|Code677 - Code677|Code678 - Desc678	bzupke	
4	⚠	Item_Staging	1000			CodeSet175	1221		Code-501 - Desc501|Code502 - Code502|Code503 - Desc503	bzupke	
5	⚠	Item_Staging	1000			CodeSet150	1221		Code-351 - Desc-351|Code352 - Code352|Code353 - Desc353	bzupke	
6	⚠	Item_Staging	1000			CodeSet125	1221		Code-226 - Desc-226|Code227 - Code227|Code228 - Desc228	bzupke	
7	⚠	Item_Staging	1000			CodeSet100	1220	Code130|Code131|Code132|Code- - Desc-126|Code127 - Code127	Code-126 - Desc-126|Code127 - Code127|Code128 - Desc128	bzupke	
8	⚠	Item_Staging	1000			NRCodeSet100	1219	Code-126	Code134	bzupke	
9	⚠	Item_Staging	1000			NRCodeSet75	1219	Code52	Code-51	bzupke	
10	⚠	Item_Staging	1000			NRCodeSet50	1219	Code2	Code-1	bzupke	
11	⚠	Item_Staging	1000			CodeSet100	1218	Code130|Code131|Code132|Code	Code130|Code131|Code132|Code- - Desc-126|Code127 - Code127	bzupke	
12	⚠	Item_Staging	1000			CodeSet75	1217	Code52,Code53	Code-51|Code52|Code53	bzupke	
13	⚠	Item_Staging	1000			CodeSet50	1216	Code2|Code3|Code4|Code5|Code6	Code2|Code3|Code4|Code5|Code6	bzupke	
14	⚠	Item_Staging	1000			CodeSet50	1215	Code2|Code3|Code4|Code|Code5|	Code2|Code3|Code4|Code5|Code6	bzupke	
15	⚠	Item_Staging	1000			CodeSet50	1214	Code2|Code3|Code4|Code	Code2|Code3|Code4|Code|Code5|	bzupke	

The list displays the entries in reverse-chronological order, with the latest entry at the top.

Translation Tab

The Translation tab identifies whether language translation is to be performed for changes to the record and identifies the translation registry entry to be used:



CN_Translation_Name
Translate Item -- [Item_Staging]

	#	State	CN_Description	CN_Import_Template_Name	CN_Sequence
☐	1	✓	Translates Item Multi-Language Fields	Item Import	

	#	State	CN_Attribute_Name	CN_Attribute_Language	CN_Attribute_Value	CN_Attribute_Translated_Value	CN_PK1	CN_Reques

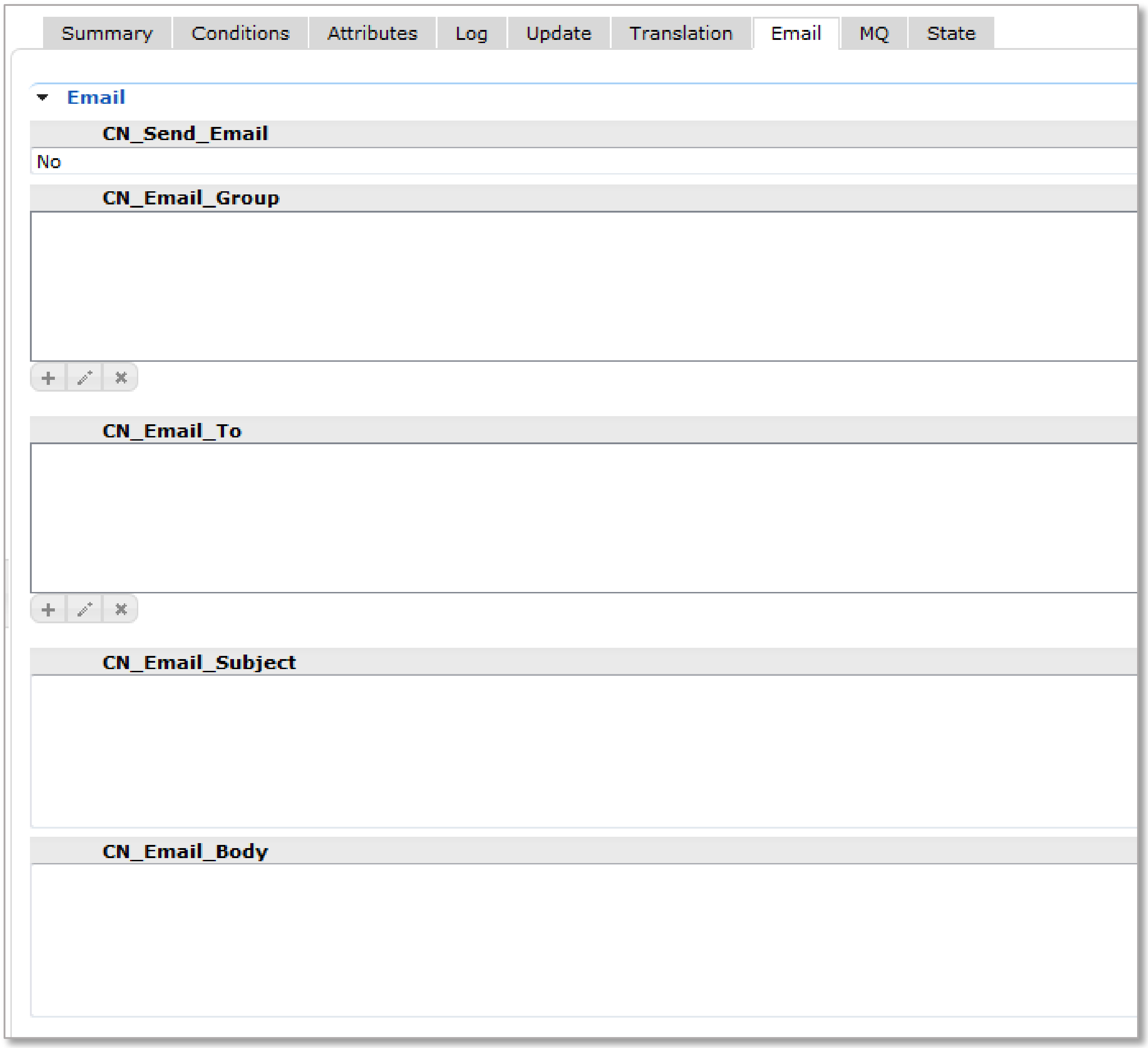
CN_Translation_Name - Identifies the CN_Translation_Registry entry to be processed. The drop-down selection list will show all defined CN_Translation_Registry record names, which means the translation registry record needs to be created first.

Translation Rules – Lists the translation rules defined for this CN_Registry record.

Translation Log – Lists the translation log messages for this CN_Registry record. It is not practical to view the log entries from the detail editor. Clicking the View Translation Log button opens the CN_Translation_Log repository for this CN_Registry record.

Email Tab

The Email tab specifies whether a notification e-mail should be generated and to whom it should be sent and what it should contain:



CN_Send_Email
No
CN_Email_Group
CN_Email_To
CN_Email_Subject
CN_Email_Body

CN_Send_Email - Specifies e-mail notification should be sent if Yes

CN_Email_Group - Lists the groups to which the user making the change must be a member in order for an e-mail to be sent. If the list is empty, then the e-mail will be sent regardless of who made the change.

CN_Email_To - List of e-mail addresses to which the e-mail is to be sent.

CN_Email_Subject - Subject line for the e-mail.

CN_Email_Body - Message body in the e-mail.

The To, Subject and Body fields can have literal text, references to attributes (or external attributes) surrounded by double-pipe characters. They can also include special fields including the ones defined under the Update Tab as well as the ones below.

- **CHANGE_DETAIL_PLAIN** – replaces the reference with the list of changes in the format:

  ```
  <attribute1Name>: old=<oldValue>,
  new=<newValue>;<attribute2Name>: old=<oldValue>, new=<newValue>
  ```

- **CHANGE_DETAIL_CSV** – replaces the reference with a CSV-formatted list:

  ```
  <attribute1Name>,<oldValue>,<neValue>
  <attribute2Name>,<oldValue>,<newValue>
  ```

- **CHANGE_DETAIL_HTML** – replaces the reference with a block of HTML table rows:

  ```
  <tr><td><attribute1Name></td><td><oldValue></td><td><newValue></t
  d></tr>
  <tr><td><attribute2Name></td><td><oldValue></td><td><newValue></t
  d></tr>
  ```

  The HTML reference should be surrounded by <table></table> tags as well as <html></html> tags. By default the generated e-mail is in plain text unless the body contains the literal value "<html>" then HTML formatting will be processed.

  For example, the following CN_Email_Body value:

  ```
  <html>
  <pre>
  The Product Group with ID=||Product Group ID|| has the following
  changes:  ||CHANGE_DETAIL_PLAIN||

  In CSV Form:

  Attribute Name,Old Value,New Value
  ||CHANGE_DETAIL_CSV||
  </pre>

  In HTML Form:
  <table border="1">
  <tr><th>Attribute Name</th><th>Old Value</th><th>New
  Value</th></tr>
  ||CHANGE_DETAIL_HTML||
  </table>
  </html>
  ```

  Produces the following e-mail message:

  ```
  The Product Group with ID=9916 has the following changes:  Prod Grp
  Hdr: old=Oil Burner Combustion Test Kit, new=Oil Burner Combustion Test
  Kits;  Prod Grp Subhdr: old=headers here, new=headers here with more
  text;  Applications: old=apps here, new=applications list here;
  ```

```
Specifications: old=specs here, new=specifications;  Features:
old=features here, new=list of features
```

In CSV Form:

```
Attribute Name,Old Value,New Value
Prod Grp Hdr,Oil Burner Combustion Test Kit,Oil Burner Combustion Test
Kits
Prod Grp Subhdr,headers here,headers here with more text
Applications,apps here,applications list here
Specifications,specs here,specifications
Features,features here,list of features
```

In HTML Form:

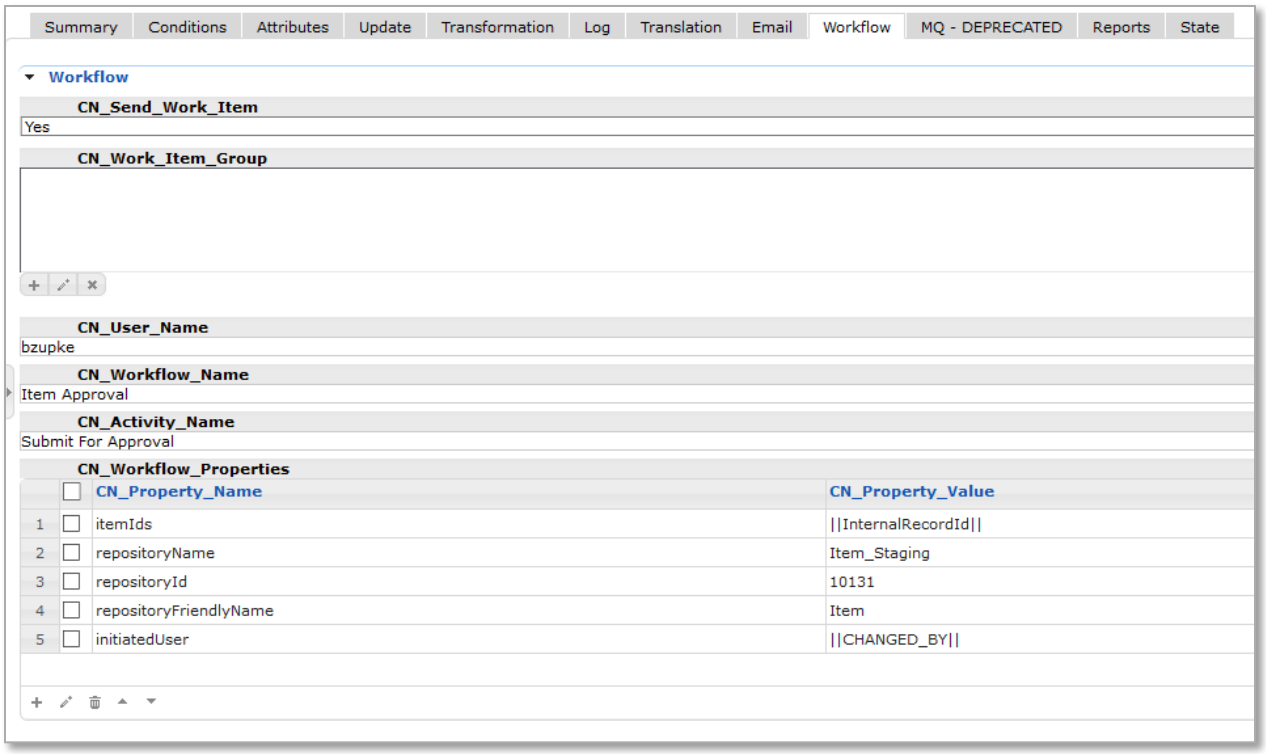
Attribute Name	Old Value	New Value
Prod Grp Hdr	Oil Burner Combustion Test Kit	Oil Burner Combustion Test Kits
Prod Grp Subhdr	headers here	headers here with more text
Applications	apps here	applications list here
Specifications	specs here	specifications
Features	features here	list of features

Email messages generated in rapid succession with the same recipient list and subject line may be combined into a single e-mail message with the contents of the message bodies concatenated into a single body.

In general, it is recommended to only use e-mail notification when the frequency of change is relatively low.

Workflow Tab

The Workflow tab specifies whether a work item should be initiated on the designated workflow and include the designated properties:



Workflow

CN_Send_Work_Item: Yes

CN_Work_Item_Group:

CN_User_Name: bzupke

CN_Workflow_Name: Item Approval

CN_Activity_Name: Submit For Approval

CN_Workflow_Properties

	CN_Property_Name	CN_Property_Value
1	itemIds	||InternalRecordId||
2	repositoryName	Item_Staging
3	repositoryId	10131
4	repositoryFriendlyName	Item
5	initiatedUser	||CHANGED_BY||

CN_Send_Work_Item - Specifies a work item should be initiated if Yes

CN_Work_Item_Group - Lists the groups to which the user making the change must be a member in order for a work item to be initiated. If the list is empty, then the e-mail will be sent regardless of who made the

CN_User_Name – login of the user on whose behalf the work item should be initiated. If this should be the user who made the record change, then ||CHANGED_BY|| can be specified for the value. This user must be identified as a participant in the designated starting point activity.

CN_Workflow_Name – name of the EPX workflow in which the work item is to be launched

CN_Activity_Name – name of the starting point activity in the designated workflow on which the work item is to be launched

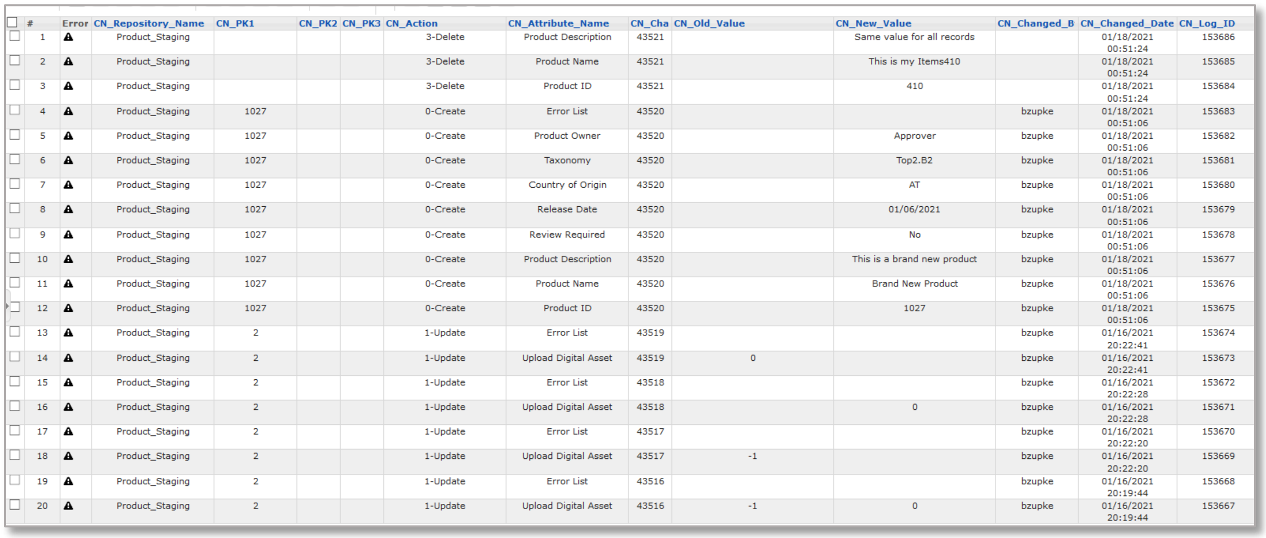
CN_Property_Name – name of a property to be defined in the work item. A maximum of 20 properties can be defined. Any beyond the first 20 will be ignored.

CN_Property_Value – value for the designated property to be defined in the work item

WARNING – Given that a work item will be created for each qualifying record change, it would be very easy to overwhelm the EPX workflow engine with events that create or update many records in a batch, such as an import or mass edit operation. This should be mitigated by either ensuring the EPX Workflow implementation can handle high-volumes of concurrent work *items or the conditions and attribute settings on the CN_Registry record limit the changes that cause a work item to be launched.*

CN_Log Repository

If logging is enabled for one or more registered change notification, records will be produced in the CN_Log repository for each attribute that has changed (or is designated to be displayed alongside a change):



☐	#	Error	CN_Repository_Name	CN_PK1	CN_PK2	CN_PK3	CN_Action	CN_Attribute_Name	CN_Cha	CN_Old_Value	CN_New_Value	CN_Changed_B	CN_Changed_Date	CN_Log_ID
☐	1	⚠	Product_Staging				3-Delete	Product Description	43521		Same value for all records		01/18/2021 00:51:24	153686
☐	2	⚠	Product_Staging				3-Delete	Product Name	43521		This is my Items410		01/18/2021 00:51:24	153685
☐	3	⚠	Product_Staging				3-Delete	Product ID	43521		410		01/18/2021 00:51:24	153684
☐	4	⚠	Product_Staging	1027			0-Create	Error List	43520			bzupke	01/18/2021 00:51:06	153683
☐	5	⚠	Product_Staging	1027			0-Create	Product Owner	43520		Approver	bzupke	01/18/2021 00:51:06	153682
☐	6	⚠	Product_Staging	1027			0-Create	Taxonomy	43520		Top2.B2	bzupke	01/18/2021 00:51:06	153681
☐	7	⚠	Product_Staging	1027			0-Create	Country of Origin	43520		AT	bzupke	01/18/2021 00:51:06	153680
☐	8	⚠	Product_Staging	1027			0-Create	Release Date	43520		01/06/2021	bzupke	01/18/2021 00:51:06	153679
☐	9	⚠	Product_Staging	1027			0-Create	Review Required	43520		No	bzupke	01/18/2021 00:51:06	153678
☐	10	⚠	Product_Staging	1027			0-Create	Product Description	43520		This is a brand new product	bzupke	01/18/2021 00:51:06	153677
☐	11	⚠	Product_Staging	1027			0-Create	Product Name	43520		Brand New Product	bzupke	01/18/2021 00:51:06	153676
☐	12	⚠	Product_Staging	1027			0-Create	Product ID	43520		1027	bzupke	01/18/2021 00:51:06	153675
☐	13	⚠	Product_Staging	2			1-Update	Error List	43519			bzupke	01/16/2021 20:22:41	153674
☐	14	⚠	Product_Staging	2			1-Update	Upload Digital Asset	43519	0		bzupke	01/16/2021 20:22:41	153673
☐	15	⚠	Product_Staging	2			1-Update	Error List	43518			bzupke	01/16/2021 20:22:28	153672
☐	16	⚠	Product_Staging	2			1-Update	Upload Digital Asset	43518		0	bzupke	01/16/2021 20:22:28	153671
☐	17	⚠	Product_Staging	2			1-Update	Error List	43517			bzupke	01/16/2021 20:22:20	153670
☐	18	⚠	Product_Staging	2			1-Update	Upload Digital Asset	43517	-1		bzupke	01/16/2021 20:22:20	153669
☐	19	⚠	Product_Staging	2			1-Update	Error List	43516			bzupke	01/16/2021 20:19:44	153668
☐	20	⚠	Product_Staging	2			1-Update	Upload Digital Asset	43516	-1	0	bzupke	01/16/2021 20:19:44	153667

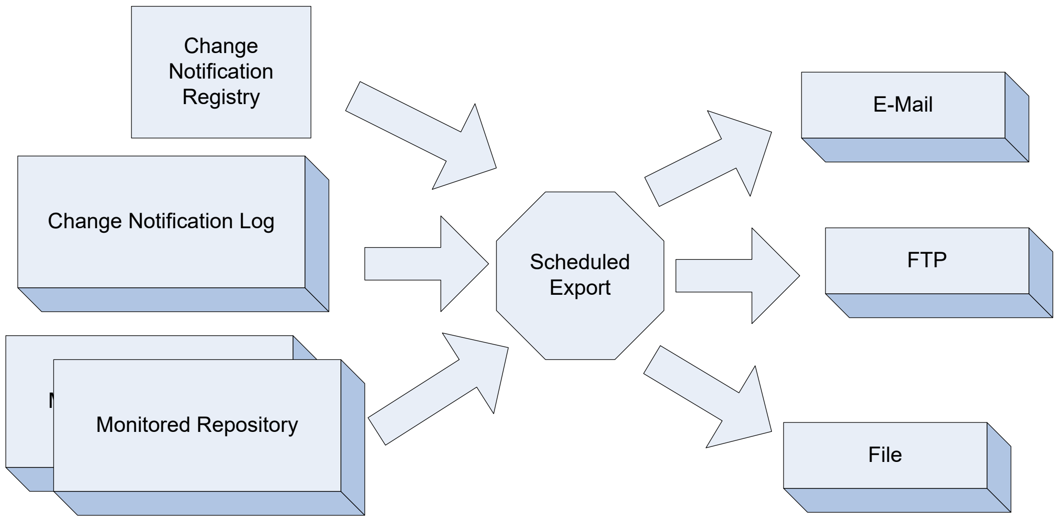
The following table provides detailed information on each attribute in this repository:

Attribute Name	Description
CN_Action	Type of action: • 0 - Create – record was created • 1 - Update – record was modified • 3 - Delete – record was deleted
CN_Attribute_Language	Identifies the language for which the change was made if the attribute is multi-language, otherwise blank.
CN_Attribute_Name	Name of the attribute that was modified, or specified to be logged always or derived from an external repository.
CN_Change_ID	Sequential ID of a specific change. If more than one monitored/logged attribute was changed, all will share the same change ID.

Attribute Name	Description
CN_Changed_By	Login name of the user who initiated the change.
CN_Changed_Datetime	Date and time of when the change was made.
CN_Internal_Record_ID	Internal ID of the record that was changed. This value can be used when performing SQL queries to retrieve current data from the record.
CN_Log_ID	Unique ID for each CN_Log record.
CN_New_Value	The current or new value for the attribute (at the time of the change).
CN_Old_Value	The previous value for the attribute. For "allways log"attributes that have not changed as well as external attributes, the old and new values will be the same.
CN_PK1, CN_PK2, CN_PK3, CN_PK4, CN_PK5	Primary key for the modified record. The number of key fields defined depends on the repository. Many repositories will only have a value in the first key field.
CN_Registry_ID	ID of the CN_Registry record that was used to log the change.
CN_Repository_ID	Internal ID of the repository in which the record was changed.
CN_Repository_Name	Name of the repository in which the record was changed.

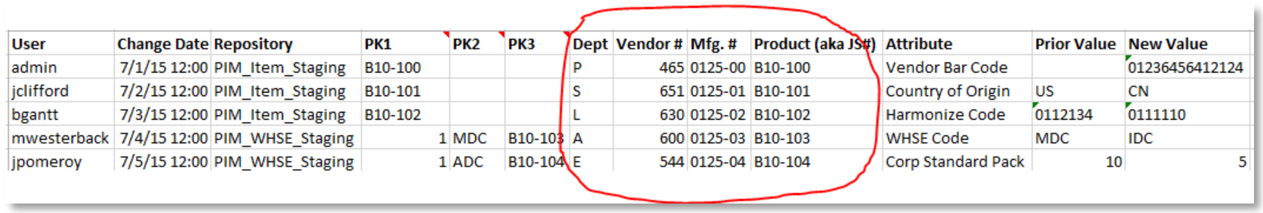
Reporting Using Scheduled Exports

Detailed reports can be generated from the CN_Log repository contents using the View (SQL) type of Scheduled Export. The SQL for the export can pull information from the Change Notification Registry and Log repositories as well as from any other repository to include contextual information:



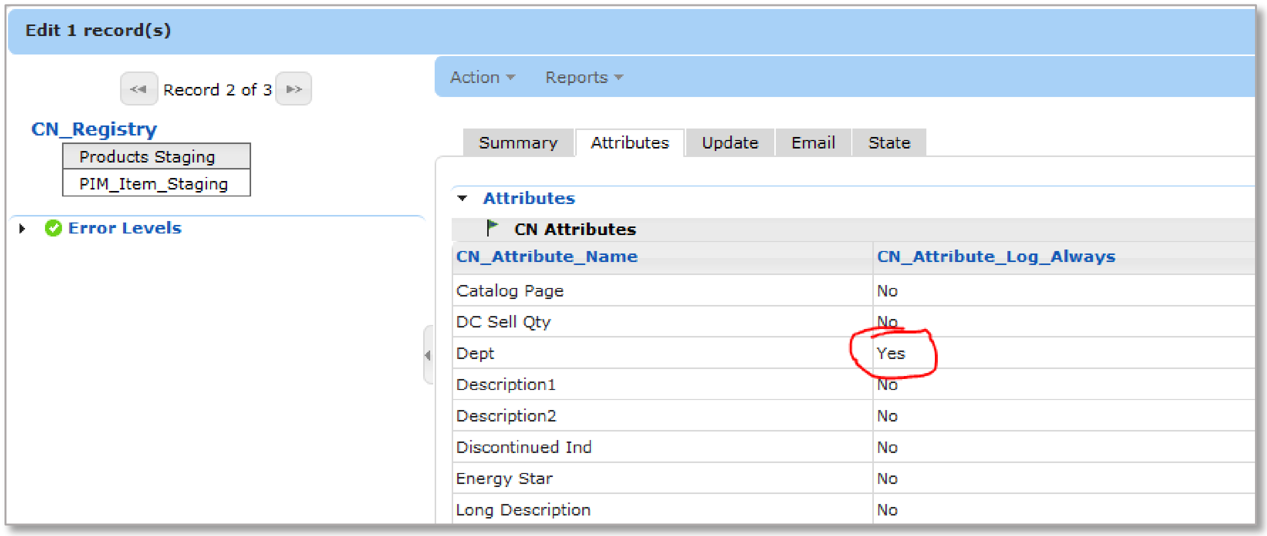
The complexity of the reporting is limited to the complexity of the SQL SELECT statement or Stored Procedure that is defined to generate that report.

For example, to generate a report that includes key business fields for each change row:



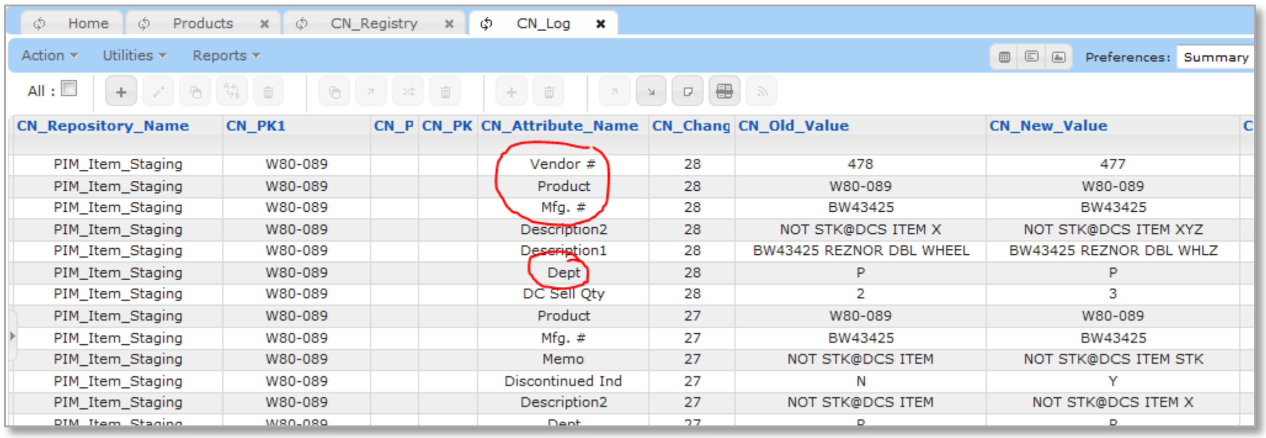
User	Change Date	Repository	PK1	PK2	PK3	Dept	Vendor #	Mfg. #	Product (aka JS#)	Attribute	Prior Value	New Value
admin	7/1/15 12:00	PIM_Item_Staging	B10-100			P	465	0125-00	B10-100	Vendor Bar Code		01236456412124
jclifford	7/2/15 12:00	PIM_Item_Staging	B10-101			S	651	0125-01	B10-101	Country of Origin	US	CN
bgantt	7/3/15 12:00	PIM_Item_Staging	B10-102			L	630	0125-02	B10-102	Harmonize Code	0112134	0111110
mwesterback	7/4/15 12:00	PIM_WHSE_Staging	1	MDC	B10-103	A	600	0125-03	B10-103	WHSE Code	MDC	IDC
jpomeroy	7/5/15 12:00	PIM_WHSE_Staging	1	ADC	B10-104	E	544	0125-04	B10-104	Corp Standard Pack	10	5

Those attributes are first defined in the Change Notification Registry as Log Always:



CN_Attribute_Name	CN_Attribute_Log_Always
Catalog Page	No
DC Sell Qty	No
Dept	Yes
Description1	No
Description2	No
Discontinued Ind	No
Energy Star	No
Long Description	No

The Change Notification log will include those attributes and their values for each logged change:

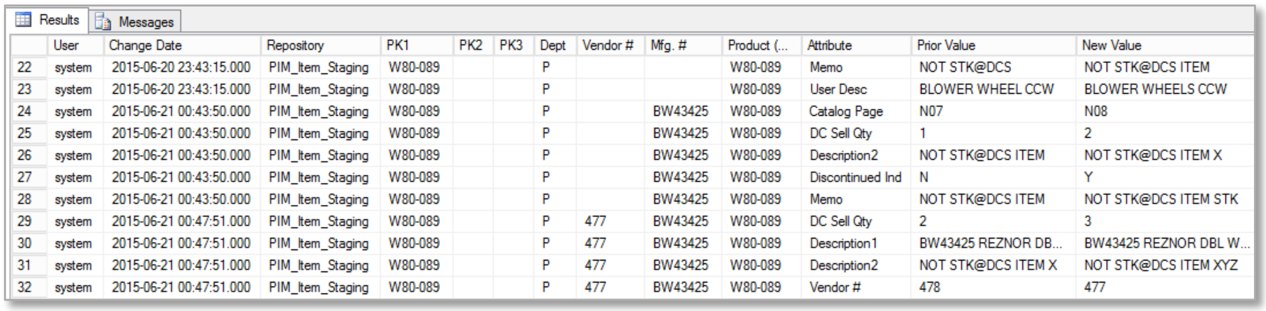


CN_Repository_Name	CN_PK1	CN_P	CN_PK	CN_Attribute_Name	CN_Chang	CN_Old_Value	CN_New_Value	C
PIM_Item_Staging	W80-089			Vendor #	28	478	477	
PIM_Item_Staging	W80-089			Product	28	W80-089	W80-089	
PIM_Item_Staging	W80-089			Mfg. #	28	BW43425	BW43425	
PIM_Item_Staging	W80-089			Description2	28	NOT STK@DCS ITEM X	NOT STK@DCS ITEM XYZ	
PIM_Item_Staging	W80-089			Description1	28	BW43425 REZNOR DBL WHEEL	BW43425 REZNOR DBL WHLZ	
PIM_Item_Staging	W80-089			Dept	28	P	P	
PIM_Item_Staging	W80-089			DC Sell Qty	28	2	3	
PIM_Item_Staging	W80-089			Product	27	W80-089	W80-089	
PIM_Item_Staging	W80-089			Mfg. #	27	BW43425	BW43425	
PIM_Item_Staging	W80-089			Memo	27	NOT STK@DCS ITEM	NOT STK@DCS ITEM STK	
PIM_Item_Staging	W80-089			Discontinued Ind	27	N	Y	
PIM_Item_Staging	W80-089			Description2	27	NOT STK@DCS ITEM	NOT STK@DCS ITEM X	
PIM_Item_Staging	W80-089			Dept	27	P	P	

Then using the PIVOT function in SQL Server, these entries can be converted to columns in the report:

```
select l.CN_Changed_By as [User], CN_Changed_Datetime as [Change Date],
CN_Repository_Name as Repository,
CN_PK1 as PK1, isnull(CN_PK2, '') as PK2, isnull(CN_PK3, '') as PK3,
isnull(c.Dept, '') as Dept, isnull(c.[Vendor #], '') as [Vendor #],
isnull(c.[Mfg. #], '') as [Mfg. #], isnull(c.Product, '') as [Product (aka JS#)],
CN_Attribute_Name as Attribute, isnull(CN_Old_Value, '') as [Prior Value],
isnull(CN_New_Value, '') as [New Value]
 from CN_Log l
left outer join (Select CN_Change_ID, Dept, [Vendor #], [Mfg. #], Product from
(SELECT CN_Change_ID, CN_Attribute_Name, CN_New_Value
    FROM (select CN_Change_ID, CN_Attribute_Name, CN_New_Value
              from CN_Log where CN_Attribute_Name in ('Dept','Vendor #','Mfg.
#','Product')) e
) AS X
PIVOT
(
MAX (CN_New_Value)
FOR CN_Attribute_Name IN (Dept, [Vendor #], [Mfg. #], Product)
) AS PivotTable) c
on c.CN_Change_ID = l.CN_Change_ID
 where not (l.CN_Attribute_Name in ('Dept','Vendor #','Mfg. #', 'Product') AND
l.CN_Old_Value = l.CN_New_Value)
 order by l.CN_Change_ID, CN_Attribute_Name
```

The above SQL query creates a pivoted table containing the data from only the change records for the attributes: Dept, Vendor #, Mfg. #, and Product. This data is joined to the main log by the Change ID. The same entries are excluded from the main entry unless they have been changed. This results in each row of the report representing a changed value:



Results | Messages

	User	Change Date	Repository	PK1	PK2	PK3	Dept	Vendor #	Mfg. #	Product (...	Attribute	Prior Value	New Value
22	system	2015-06-20 23:43:15.000	PIM_Item_Staging	W80-089			P			W80-089	Memo	NOT STK@DCS	NOT STK@DCS ITEM
23	system	2015-06-20 23:43:15.000	PIM_Item_Staging	W80-089			P			W80-089	User Desc	BLOWER WHEEL CCW	BLOWER WHEELS CCW
24	system	2015-06-21 00:43:50.000	PIM_Item_Staging	W80-089			P		BW43425	W80-089	Catalog Page	N07	N08
25	system	2015-06-21 00:43:50.000	PIM_Item_Staging	W80-089			P		BW43425	W80-089	DC Sell Qty	1	2
26	system	2015-06-21 00:43:50.000	PIM_Item_Staging	W80-089			P		BW43425	W80-089	Description2	NOT STK@DCS ITEM	NOT STK@DCS ITEM X
27	system	2015-06-21 00:43:50.000	PIM_Item_Staging	W80-089			P		BW43425	W80-089	Discontinued Ind	N	Y
28	system	2015-06-21 00:43:50.000	PIM_Item_Staging	W80-089			P		BW43425	W80-089	Memo	NOT STK@DCS ITEM	NOT STK@DCS ITEM STK
29	system	2015-06-21 00:47:51.000	PIM_Item_Staging	W80-089			P	477	BW43425	W80-089	DC Sell Qty	2	3
30	system	2015-06-21 00:47:51.000	PIM_Item_Staging	W80-089			P	477	BW43425	W80-089	Description1	BW43425 REZNOR DB...	BW43425 REZNOR DBL W...
31	system	2015-06-21 00:47:51.000	PIM_Item_Staging	W80-089			P	477	BW43425	W80-089	Description2	NOT STK@DCS ITEM X	NOT STK@DCS ITEM XYZ
32	system	2015-06-21 00:47:51.000	PIM_Item_Staging	W80-089			P	477	BW43425	W80-089	Vendor #	478	477

To produce a delta report, showing only the changes since the previous report, the SQL must include the condition comparing the Created column to the [DELTA_DATETIME] field:

```
where not (l.CN_Attribute_Name in ('Dept','Vendor #','Mfg. #', 'Product') AND
l.CN_Old_Value = l.CN_New_Value)
 and l.Created > '[DELTA_DATETIME]'
```

Transformation Extension

Overview

The Transformation Extension functionality leverages the Change Notification functionality to facilitate conditionally updating attributes in the changed record or other record(s) that are linked to the changed record, using values from the changed record or from another record that is linked to the changed record. The Transformation definitions are intended to be managed by business users and not require programming skills. Basic knowledge of SQL queries is a key requirement for accessing or updating records linked to the changed record.

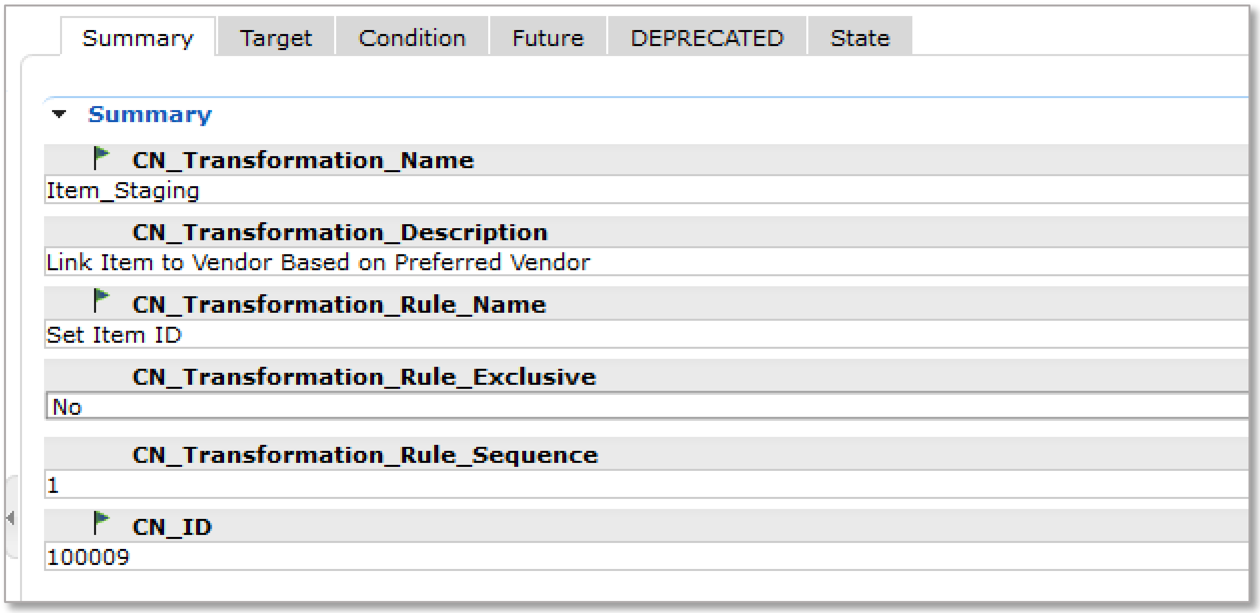
Each named Transformation is referenced by a CN_Registry entry that activates it. One or more records are defined in the CN_Transformation_Registry

CN_Transformation_Repository

The CN_Transformation repository contains the transformation rules for one or more CN_Registry record. Each record in this repository represents a single condition on a rule for a transformation. Multiple conditions may be defined for a single rule and multiple rules may be defined for a single Transformation.

Summary Tab

The Summary tab identifies the Transformation Name and Rule:



Summary

CN_Transformation_Name	Item_Staging
CN_Transformation_Description	Link Item to Vendor Based on Preferred Vendor
CN_Transformation_Rule_Name	Set Item ID
CN_Transformation_Rule_Exclusive	No
CN_Transformation_Rule_Sequence	1
CN_ID	100009

CN_Transformation_Name – name of the transformation. This must match the CN_Transformation_Name value in a CN_Registry entry. All records in the CN_Transformation_Registry repository having the same CN_Transformation_Name will be processed together whenever a repository record is changed for which a CN_Registry record applies AND that record references the Transformation by this name.

CN_Transformation_Description – optional description for the transformation. Likely to be defined only the first record for the Transformation but may have different values for each record to clarify the transformation rule or condition

CN_Transformation_Rule_Name – name of the transformation rule. Multiple CN_Transformation records may identify the same rule for the same Transformation. Multiple rules may be defined for the same Transformation.

CN_Transformation_Rule_Exclusive – indicates the transformation rule is exclusive if Yes. Each exclusive rule will only be processed if no previous transformation rule for the same transformation has its conditions met. Each rule that is not exclusive will always be applied, regardless of whether any previous rule has its conditions met. Only one exclusive transformation rule will be processed for a given Transformation and given change event. The Transformation_Rule_Sequence can be used to force the processing of exclusive rules before any non-exclusive rules in case there is a dependency. The sequence can also be used in cases where some conditions may be common to multiple rules, but some rules have additional conditions being sequenced first.

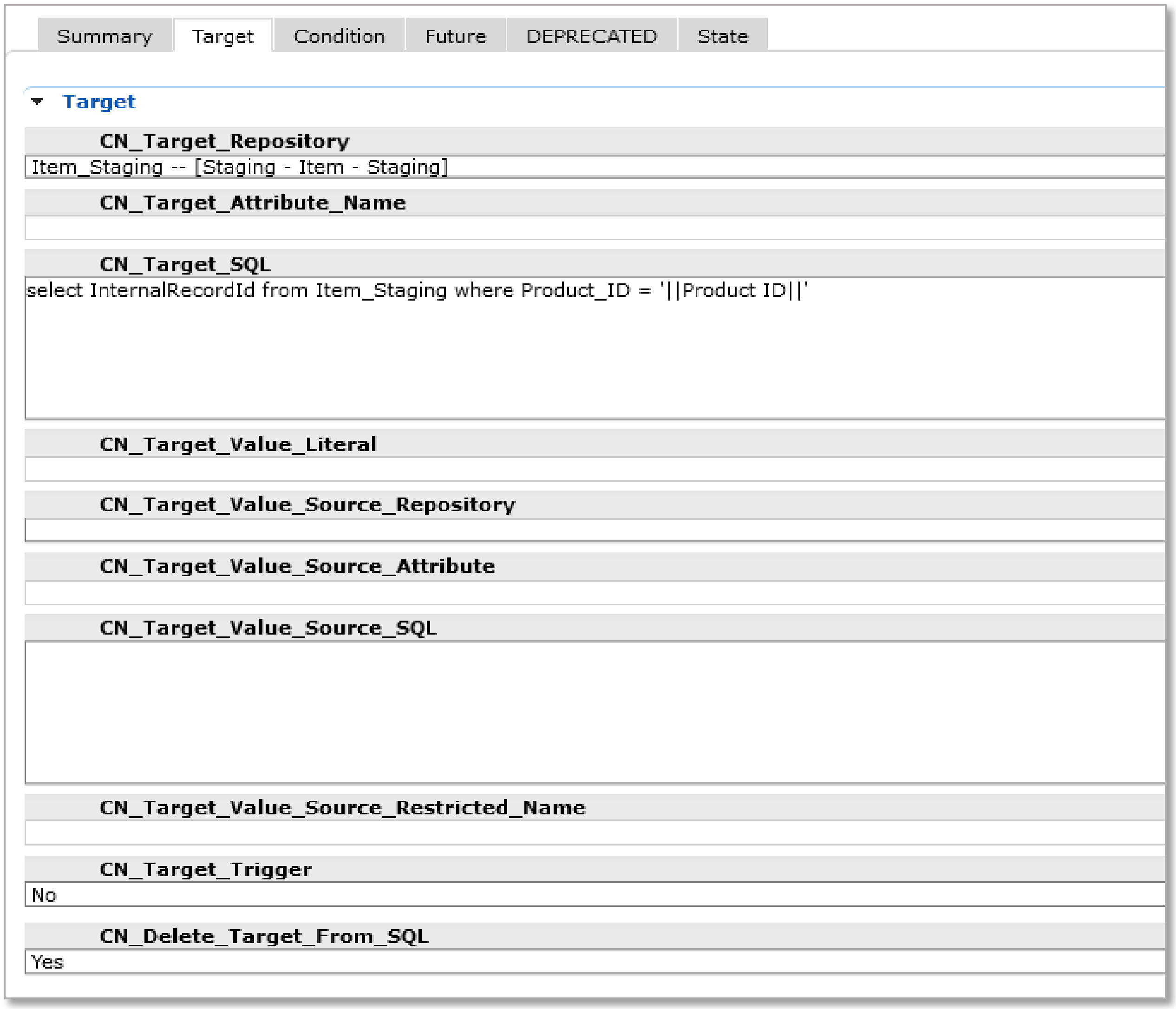
CN_Transformation_Rule_Sequence – specifies the order in which the transformation rules are processed. This is only important if the Transformation_Rule_Exclusive is Yes.

For each rule for which the conditions are met, log the details of the rule that was applied to the log file and perform the update(s) identified by the corresponding Target settings.

CN_ID – unique identifier for each CN_Transformation_Registry record

Target Tab

The Target tab identifies what attribute in what repository record(s) to update and with what value if the Rule conditions are met.



CN_Target_Repository
Item_Staging -- [Staging - Item - Staging]

CN_Target_Attribute_Name

CN_Target_SQL
select InternalRecordId from Item_Staging where Product_ID = '||Product ID||'

CN_Target_Value_Literal

CN_Target_Value_Source_Repository

CN_Target_Value_Source_Attribute

CN_Target_Value_Source_SQL

CN_Target_Value_Source_Restricted_Name

CN_Target_Trigger
No

CN_Delete_Target_From_SQL
Yes

CN_Target_Repository – identifies the target repository to be updated if the Transformation rules determine any changes should be made. This will typically be the same repository identified in the CN_Registry entry that invokes this transformation. If the target repository is not the same, there must the specific record(s) to be updated must be identified by either the CN_Target_SQL attribute.

CN_Target_Attribute_Name – name of the target attribute to be updated by the transformation.

CN_Target_SQL - SQL expression identifying the Target record(s) to be updated with the target value. The SQL must return the InternalRecordID of the existing record(s) to be updated. If it returns a null or a negative number, a new record will be created with this value. If other rules identify the same InternalRecordID, they will be included in the created record(s). The use of negative numbers allows for multiple records to be created at one time by different rules. If the query returns references to multiple records (i.e., multiple result rows), each referenced record is to be updated according to the Target_Attribute_Name and Target value.

By default, the identified target value will be updated in each repository record identified by the CN_Target_SQL. Additional attributes can be updated in those records by configuring the CN_Target_SQL to return additional columns, with their names exactly matching (including case) attribute names in the target repository.

Note: The result columns must be defined with the attribute names, not restricted attribute names. Any column that does not exactly match an attribute name will be ignored.

This means that linked repository records can be updated with a single Transformation rule AND each record can be updated with dynamic information. When using just one of the Target_Value_<type> attributes, each record identified by the CN_Target_SQL will be updated with the same value AND multiple rules would need to be defined to populate multiple attributes in each of those records.

A practical example would be to define a Transformation rule that creates missing DAMLink records when a repository record is created or updated or a DAMMaster record is created. The Rule Condition can determine whether any DAMLink records are needed for the repository record and the CN_Target_SQL can populate all fields in each of those missing records. Each of the result rows must have a unique negative number. This can be accomplished using SQL similar to the following for SQL Server:

```
select -1 * (ROW_NUMBER() OVER(ORDER BY <anyColumn> ASC)) as
InternalRecordId,
```

and similar to the following for PostgreSQL:

```
select -1 * row_number() OVER () as InternalRecordId,
```

Where <anyColumn> is any column accessible by the query, such as the one representing the primary key.

Multiple attributes in the trigger record can be updated from the same CN_Target_SQL query by having the first column be defined as: ||InternalRecordId||. This will result in those result columns being added to the item changes being collected for the trigger record.

CN_Target_Value_Literal – specifies the value for the target attribute if the transformation rule conditions are met. Must be blank if the target value is to come from an attribute in a linked record. The value “[clear]” (without quotes) must be used if the target attribute is to be cleared of its value.

CN_Target_Value_Source_Repository – name of the repository containing the target value to be used. This repository must only be specified when the source is the trigger record. Ignored if the Target_Literal_Value is not empty

CN_Target_Value_Source_Attribute – name of the attribute in the trigger record whose value is to be placed in the target.

CN_Target_Value_Source_SQL – SQL expression returning a row of data to be used as the source for the target value. A column name must match the Target_Value_Source_Restricted_Name (defined below). The same SQL may be defined in multiple conditions but each referencing a different column. This allows the data for multiple conditions to be returned and used for multiple conditions and rules with a single execution of the query. This SQL can return calculated values as well as columns from linked records. For example, if the target value is revising an End Date to be one day before a new record's start date.

CN_Target_Value_Source_Restricted_Name – restricted name of the query result column containing the target value to be used. Ignored if the Target_Literal_Value is not empty.

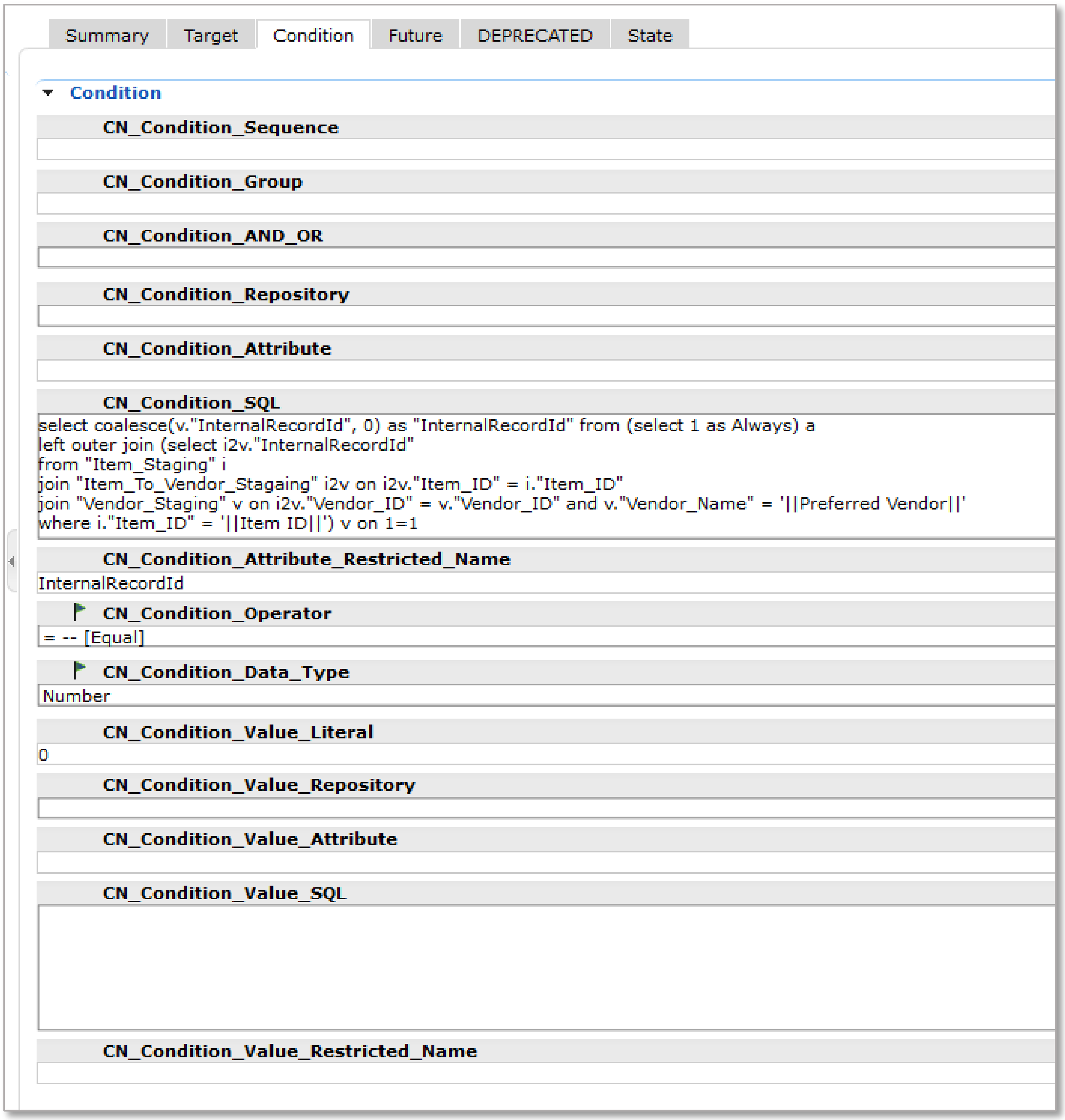
CN_Target_Trigger – causes the trigger to be fired for every record that is updated by this rule if Yes. This should be set to Yes only in cases where the updates being made to the target record(s) are subject to change notification/transformation processing.

WARNING: Care must be taken to avoid an infinite loop where an update to a record with CN_Target_Trigger=Yes results in another Change Notification Transformation that updates a record with CN_Target_Trigger=Yes such that an infinite loop results. This must be prevented by either having the CN_Target_Trigger=No or conditions on the transformation rules such that updates happen for only one cycle.

CN_Delete_Target_From_SQL – deletes the target records identified in the CN_Target_SQL if Yes.

Condition Tab

The Condition tab defines a condition for the Transformation rule that must be met in order for the target attribute to be updated with the target value.



CN_Condition_Sequence

CN_Condition_Group

CN_Condition_AND_OR

CN_Condition_Repository

CN_Condition_Attribute

CN_Condition_SQL
```
select coalesce(v."InternalRecordId", 0) as "InternalRecordId" from (select 1 as Always) a
left outer join (select i2v."InternalRecordId"
from "Item_Staging" i
join "Item_To_Vendor_Stagaing" i2v on i2v."Item_ID" = i."Item_ID"
join "Vendor_Staging" v on i2v."Vendor_ID" = v."Vendor_ID" and v."Vendor_Name" = '||Preferred Vendor||'
where i."Item_ID" = '||Item ID||') v on 1=1
```

CN_Condition_Attribute_Restricted_Name
InternalRecordId

CN_Condition_Operator
= -- [Equal]

CN_Condition_Data_Type
Number

CN_Condition_Value_Literal
0

CN_Condition_Value_Repository

CN_Condition_Value_Attribute

CN_Condition_Value_SQL

CN_Condition_Value_Restricted_Name

CN_Condition_Sequence – determines the order in which the conditions are processed when more than one condition is defined for the same rule.

CN_Condition_Group – optional name of the group of conditions in cases where combinations of AND and OR are needed. The conditions within a group are combined based on the Condition_AND_OR setting on all but the first condition in that group. The condition groups are combined by the setting of the Condition_AND_OR of the first condition of each group. Any condition that does not have a condition group assigned will be automatically part of a DEFAULT group and will be treated as though a group name was specified.

CN_Condition_AND_OR – specifies whether the condition must be met with the other conditions defined or

The conditions for a rule are handled in a consistent manner on a group by group basis, followed by the results of each group. The evaluation of the conditions in the group will have the following behavior:

1. The first condition is evaluated to produce an outcome of true or false.
2. If there are no other conditions in the group, this value is used as the outcome of the group.
3. If there are additional conditions then the first condition outcome is assigned to the current condition outcome
4. Loop for each condition
   a. If the current condition outcome is true and the next condition is set to OR, it is skipped and the value defined for the first condition is returned.
   b. If the current condition outcome is true and the next condition is set to AND, it is evaluated.
      i. If the next condition is false, the false outcome is returned for the group
      ii. If the next condition is true, the current condition outcome is set to true and return to the Loop for the next condition. All conditions for the same rule must evaluate to true in order for the rule to be applied. If this is the case, the target value associated with the last condition will be used.
   c. If the current condition outcome is false and the next condition is set to OR, return to the Loop for the next condition
   d. If the current condition outcome is false and the next condition is set to AND, the false outcome is returned for the group.
5. If all conditions have been processed, return the current condition outcome. In summary, if the conditions within a group are combined with OR, the first condition to evaluate to true will be used to identify the value for the target. If the conditions within a group are combined with AND, the last condition to be evaluated will be used to identify the value for the target (providing all conditions to evaluate to true). If multiple condition groups are defined for a rule, the same logic is applied to the outcome of each condition group.

CN_Condition_Repository – name of the repository containing the attribute containing the value to be tested for the condition. This should only be set when the value to be tested is in the trigger record.

CN_Condition_Attribute – name of the attribute containing the value to be tested for the condition. Only used if the attribute is in the Trigger record

CN_Condition_SQL - SQL expression returning the value to be applied to the left-side of the condition. The SQL should identify a record that is linked to the change event repository record (as an alternative to referencing the link relationship). The query must return one or more columns, of which only one will be referenced by this condition (in the CN_Condition_Attribute_Restricted_Name attribute). Multiple columns should be specified if different conditions need different values from the same record as the data will be cached after the first time the SQL is executed. If multiple rows are returned, the rows after the first are ignored.

CN_Condition_Attribute_Restricted_Name – restricted name of the attribute containing the value to be tested for the condition. Only used if the Condition_Repository and Condition_Link_Relationship, or Condition_Link_SQL are defined.

CN_Condition_Operator – operator to be used to compare the specified attribute value against the condition value. Supported operators are:

- = - the attribute has the same value as the Condition_Value
- <> - the attribute does not have the same value as the Condition_Value
- > - the attribute is greater (or alphabetically higher) than the Condition_Value
- >= - the attribute is greater or equal (or alphabetically higher or equal) than the Condition_Value
- < - the attribute is smaller (or alphabetically lower) than the Condition_Value
- <= the attribute is smaller or equal (or alphabetically lower or equal) than the Condition_Value
- Is One Of – the attribute matchs one of the delimited values in the Condition_Value
- Is Not One Of – the attribute doesn't match any of the delimited values in the Condition_Value
- Is Empty – the attribute doesn't have a value
- Is Not Empty – the attribute has a value
- Like – the attribute matches the pattern where '**%**' and '**_**' are wildcards
- True – always returns true. This can be used to act as a default rule for a sequence of exclusive rules. If no other rule is met, the rule with this condition will always be

CN_Condition_Data_Type – Data type for the condition. Both sides of the condition must be of the same type. The supported types are:

- Text – the values are to be compared as text strings
- Number – the values are to be compared as numbers
- Date – the values are to be compared as date (and optional time)

CN_Condition_Value_Literal – value used in the comparison with the Condition_Attribute. If the Condition_Operator is Is One of or Is Not One Of, the value must be a comma-delimited list.

CN_Condition_Value_Repository – name of the repository containing the record and attribute to be compared to the Condition_Attribute. Ignored if Condition_Value_Literal is set. The repository must be the same as the change event repository.

CN_Condition_Value_Attribute – name of the attribute to be compared to the Condition_Attribute. Only used if Condition_Value_Literal, Condition_Value_Repository and Condition_Value_Link_Relationship, or Condition_Value_SQL are not set.

CN_Condition_Value_SQL – SQL expression returning a row of data to be used as the source for the condition value. A column name must match the Condition_Value_Restricted_Attribute (defined below). The same SQL may be defined in multiple conditions but each referencing a different column. This allows the data for multiple conditions to be returned and used for multiple conditions and rules with a single execution of the query. This SQL can return calculated values as well as columns from linked records. For example, if the target value is revising an End Date to be one day before a new record's start date, the SQL will include an expression that subtracts one day from the snapshot view column that contains the record's start datae. This field is ignored if Condition_Value_Literal is set.

CN_Condition_Value_Restricted_Name – restricted name of the SQL result column to be compared to the Condition_Attribute. Ignored if Condition_Value_Literal or Condition_Value_Attribute is set

Future Tab

This tab contains some attributes that may be used in the future

CN_Target_Link_Relationship – (FUTURE – NOT YET IMPLEMENTED) optional name of the link relationship that is to be used to identify the target record. Can be blank if the Target_Repository is the same as the repository specified in the CN_Registry record invoking this transformation. If the target repository is not the same, the Target_Link_Relationship must be specified

CN_Target_Value_Source_Link_Relationship – (FUTURE – NOT YET IMPLEMENTED) name of the link relationship to be used to identify the record containing the target value as defined by the Target_Value_Repository and Target_Value_Attribute fields. Must be defined if the Target_Value_Repository is not the change event repository

CN_Condition_Link_Relationship – (FUTURE – NOT YET IMPLEMENTED) name of the link relationship if the Condition_Repository is different than the change event repository

CN_Condition_Value_Link_Relationship – (FUTURE – NOT YET IMPLEMENTED) name of the link relationship to be used to identify the record in the designated repository. Ignored if Condition_Value_Literal is set

SQL Expressions

For each of the SQL fields listed above, the SQL expression can either be one of the following:

- Full select statement, starting with “SELECT”. If the SQL is to include dynamic information from the triggered event record, values can be referenced by surrounding the attribute name with double-pipe characters. This will accommodate use cases where other Change Notification events or a trigger event on the triggered repository have modified one or more attributes that are referenced in the SQL. The SQL statement should retrieve a superset of columns such that the same SQL can be used for each target, source, or condition that access the same record. If the SQL is for a link, it must return the InternalRecordId for the record. If the SQL is for a value, it must return one more more columns which are referenced by the restricted name field.
- Stored procedure invocation, starting with “EXEC”. Dynamic information can be included as parameters to the stored procedure in the same manner as the full select statement.
- JOIN/WHERE clauses, starting with “JOIN” relies on the SELECT and FROM clauses will be generated automatically. The FROM clause will reference the snapshot view associated to the repository to which the triggered event is assigned. This view will use the alias ‘v”. To join with additional tables, the SQL should start with: “JOIN <tableOrView) v2 ON v2.<column> = v.<column>….”. If the SQL expression is to include a WHERE clause, dynamic values from the triggered record can be referenced by surrounding each attribute name with double-pipe characters.
- WHERE clause, starting with “WHERE”. The trigger repository can be referenced with the “v” alias. Dynamic values from the triggered record can be referenced by surrounding each attribute name with double-pipe characters.

Data from the changed record can be included in the statement through named references (denoted by the attribute name surrounded by double pipe-characters). These references are resolved before the SQL statement is executed. If the value needs to be treated as a character value in the query, it must be surrounded by single quotes.

For example, if the Enable data model has a Product repository linked to an Item repository by an alphanumeric ID and changes to Item records need to record the Product Group Name (from the Product record), the following SQL can be used to retrieve that information:

```
SELECT Product_Group_Name FROM _PIM_Product_Staging WHERE
Product_ID = '||Product ID||'
```

In addition to referencing any attribute in the monitored repository, the SQL may also contain the date and time stamp of when the change was made by specifying the reserved reference ||CURRENT_DATETIME||. This is resolved to the current date and time in the format yyyyMMddHHmmssSS. Alternative formats can be specified by using the Java class SimpleDateFormat's notation. For example, to only include the date in mm/dd/yyyy format, the reference must be ||CURRENT_DATETIMEMM/dd/yyyy||.

WARNING – If Enable is using the PostgreSQL database, the double-pipe characters represent the string concatenation operator. If the first character following the double-pipe characters is a letter or number, the Change Notification Transformation processing assumes its an attribute reference. If concatenating strings, the double pipe characters must be followed by a non-alphanumeric character (e.g., a space or single quote). Alternatively, the concat() function can be used.

Example Transformations

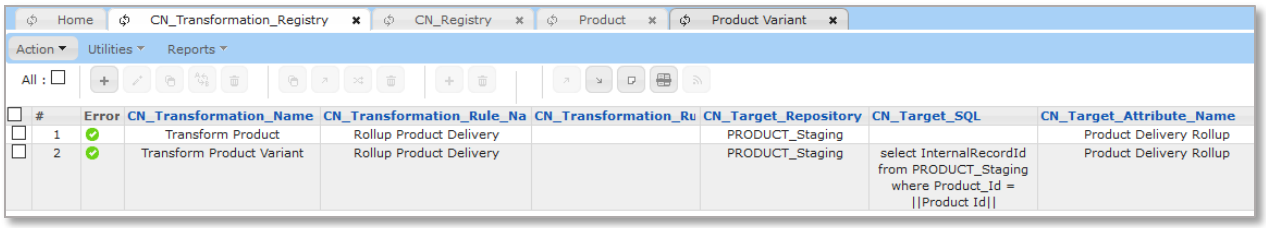
The following sections provide examples of common transformations.

Rollup Attribute

When a search is performed on a repository, Enable can only search on attributes in that repository or in linked repositories one "jump" away from the repository of interest. This means it is sometimes necessary to replicate data that is in a related repository of interest that is two or more jumps away into an adjacent repository, or even into the repository being searched. If the nature of the data is one-to-many where the repository of interest is the "one" and the related repository of interest is the "many", the different values in the "many" records need to be rolled-up into a single value and stored in a reachable repository. The rollup-attribute needs to be updated any time the data in the "many" records is changed. For thoroughness, the rollup-attribute can also be updated any time the record storing the value is changed (and the computed rollup value differs from the actual rollup value).

For this example, the data model has two repositories: PRODUCT_Staging and PRODUCT_Variant_Staging. There is a one-to-many relationship between PRODUCT_Staging (one) and PRODUCT_Variant_Staging (many). The PRODUCT_Variant_Staging has the attribute "Product Delivery", which is tied to a code set. The PRODUCT_Staging repository has the attribute "Product Delivery Rollup" which must contain a delimited list of all unique Product Delivery values from the linked PRODUCT_Staging records. In order to ensure the rollup attribute is always up to date, two change notification transformations need to be defined: one on PRODUCT_Staging and the other on PRODUCT_VARIANT_Staging.

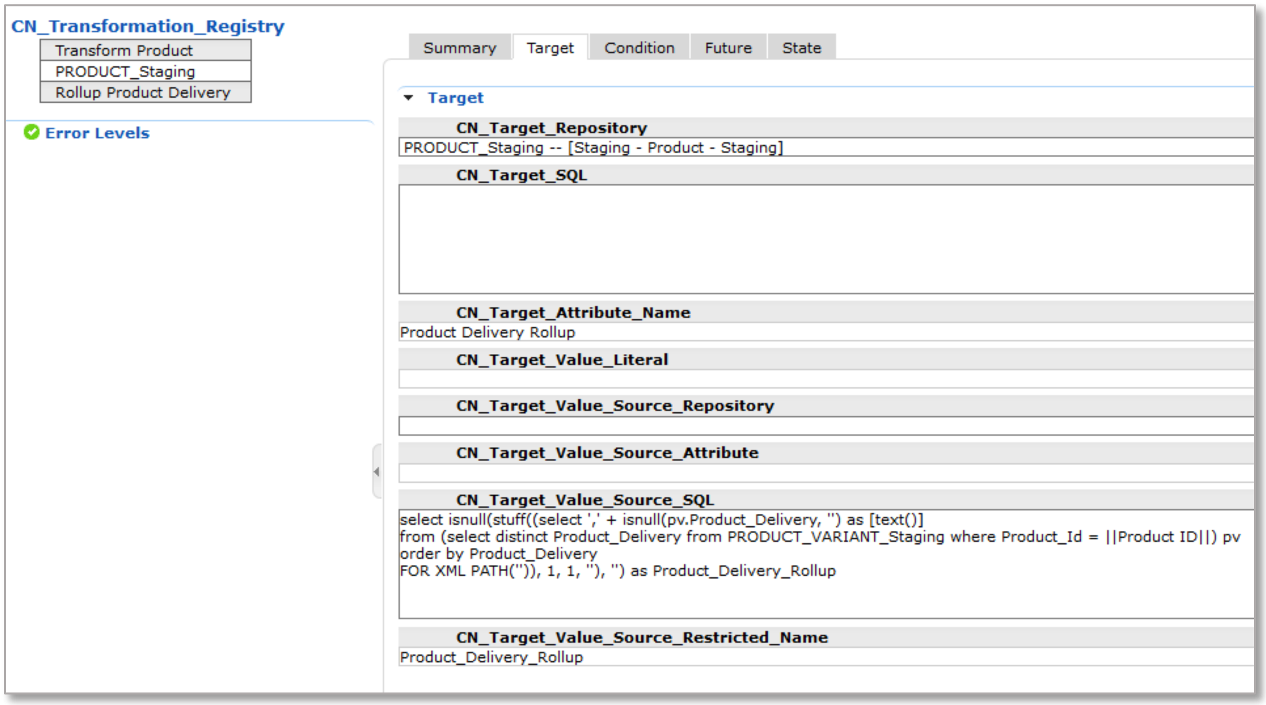
The CN_Transformation_Registry repository needs an entry for each repository to perform the transformation whenever they are invoked by their respective CN_Registry entries which are invoked any time records in the respective repositories have been created or changed:



#	Error	CN_Transformation_Name	CN_Transformation_Rule_Na	CN_Transformation_Ru	CN_Target_Repository	CN_Target_SQL	CN_Target_Attribute_Name
1	✓	Transform Product	Rollup Product Delivery		PRODUCT_Staging		Product Delivery Rollup
2	✓	Transform Product Variant	Rollup Product Delivery		PRODUCT_Staging	select InternalRecordId from PRODUCT_Staging where Product_Id = ||Product Id||	Product Delivery Rollup

The Transform Product transformation needs to collect the unique Product Delivery values from the linked PRODUCT_VARIANT_Staging records, and then compare them to the Product Delivery Rollup value currently stored in the PRODUCT_Staging record. If the values differ, the Product Delivery Rollup attribute must be updated with the new computed value.

The target for PRODUCT_Staging is defined as follows:



CN_Target_Repository
PRODUCT_Staging -- [Staging - Product - Staging]

CN_Target_SQL

CN_Target_Attribute_Name
Product Delivery Rollup

CN_Target_Value_Literal

CN_Target_Value_Source_Repository

CN_Target_Value_Source_Attribute

CN_Target_Value_Source_SQL

```
select isnull(stuff((select ',' + isnull(pv.Product_Delivery, '') as [text()]
from (select distinct Product_Delivery from PRODUCT_VARIANT_Staging where Product_Id = ||Product ID||) pv
order by Product_Delivery
FOR XML PATH('')), 1, 1, ''), '') as Product_Delivery_Rollup
```

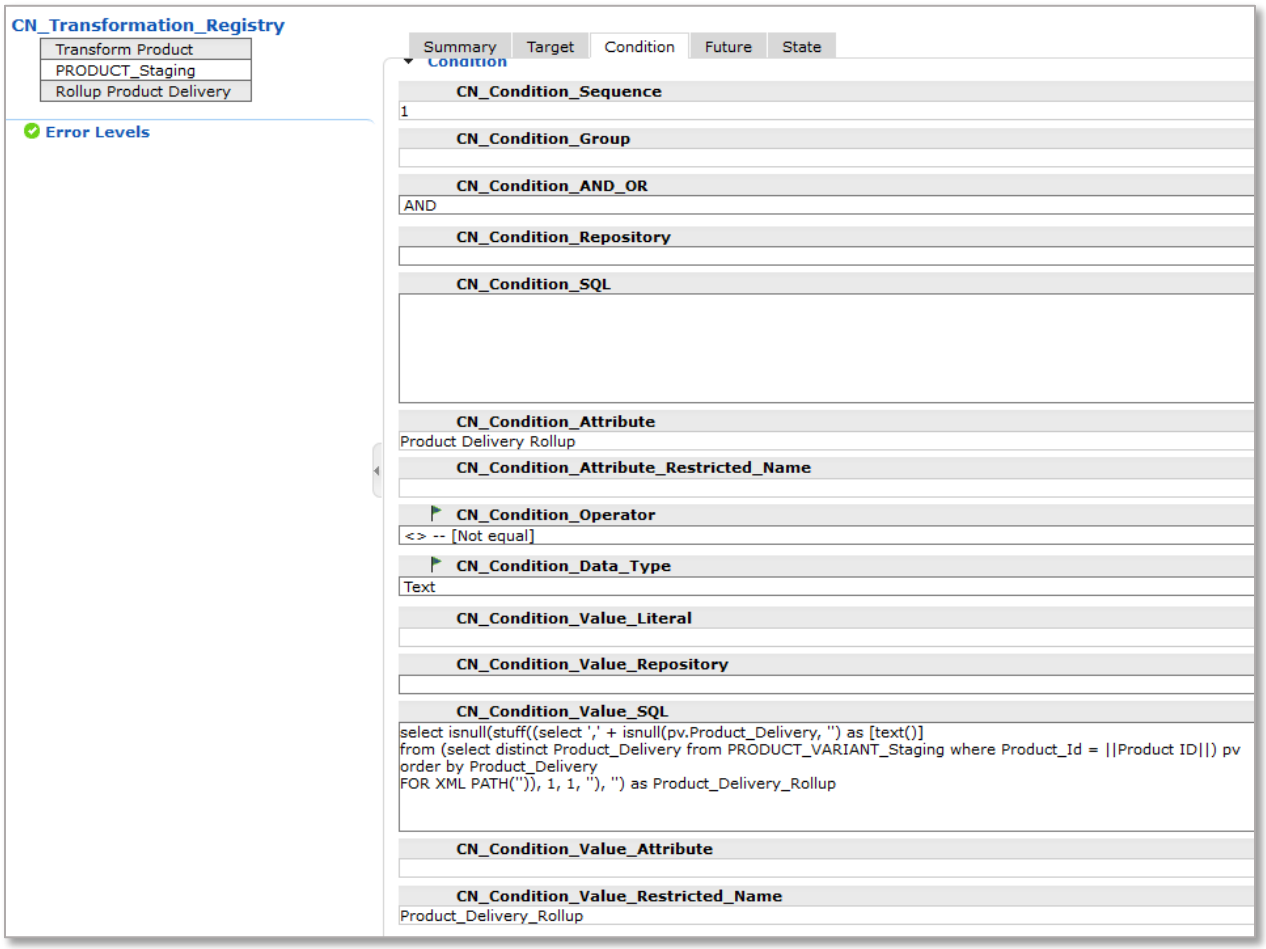
CN_Target_Value_Source_Restricted_Name
Product_Delivery_Rollup

PRODUCT_Staging is identified as the CN_Target_Repository. This is the same repository that is the trigger repository. This means the target record does not have to be identified by CN_Target_SQL or CN_Target_Link_Relationship. The source value for the target is defined by a SQL query that produces a delimited list of the unique Product Delivery attributes in the linked PRODUCT_VARIANT_Staging records using the following SQL Query:

```
select isnull(stuff((select ',' + isnull(pv.Product_Delivery, '')
as [text()]
from (select distinct Product_Delivery from
PRODUCT_VARIANT_Staging where Product_Id = ||Product ID||) pv
order by Product_Delivery
FOR XML PATH('')), 1, 1, ''), '') as Product_Delivery_Rollup
```

The SQL query ensures only the unique values are included (distinct) and they are sorted alphabetically.

Since the update is only necessary if the calculated list of values differs from the actual list of values, the condition for the transformation rule compares the two and requires that they are not equal:



CN_Condition_Sequence
1

CN_Condition_Group

CN_Condition_AND_OR
AND

CN_Condition_Repository

CN_Condition_SQL

CN_Condition_Attribute
Product Delivery Rollup

CN_Condition_Attribute_Restricted_Name

CN_Condition_Operator
<> -- [Not equal]

CN_Condition_Data_Type
Text

CN_Condition_Value_Literal

CN_Condition_Value_Repository

CN_Condition_Value_SQL
```
select isnull(stuff((select ',' + isnull(pv.Product_Delivery, '') as [text()]
from (select distinct Product_Delivery from PRODUCT_VARIANT_Staging where Product_Id = ||Product ID||) pv
order by Product_Delivery
FOR XML PATH('')), 1, 1, ''), '') as Product_Delivery_Rollup
```

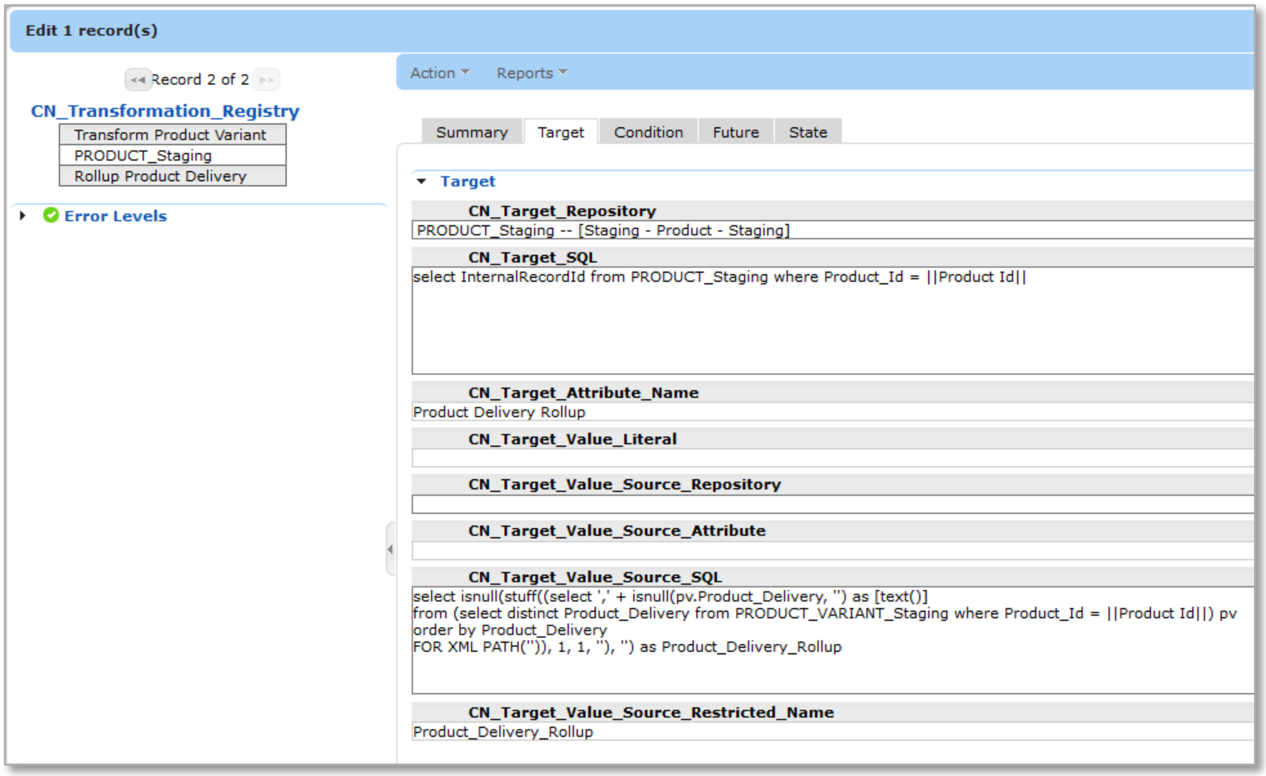
CN_Condition_Value_Attribute

CN_Condition_Value_Restricted_Name
Product_Delivery_Rollup

The attribute to be compared on the trigger record is Product Delivery Rollup. The value to be compared is the calculated rollup value defined by the same SQL that is used to populate the target attribute. Since the SQL for the target and condition are identical it is only executed once per event and the cached results used for the condition compare and the target value population.

The PRODUCT_VARIANT_Staging needs a similar Transformation to support the use-cases where new records are defined, existing records are linked to a different PRODUCT_Staging record, or Product Delivery is changed in an existing variant to a different value.

The target for PRODUCT_VARIANT_Staging is defined as follows:



CN_Transformation_Registry

Transform Product Variant
PRODUCT_Staging
Rollup Product Delivery

Target

CN_Target_Repository
PRODUCT_Staging -- [Staging - Product - Staging]

CN_Target_SQL
select InternalRecordId from PRODUCT_Staging where Product_Id = ||Product Id||

CN_Target_Attribute_Name
Product Delivery Rollup

CN_Target_Value_Literal

CN_Target_Value_Source_Repository

CN_Target_Value_Source_Attribute

CN_Target_Value_Source_SQL
```
select isnull(stuff((select ',' + isnull(pv.Product_Delivery, '') as [text()]
from (select distinct Product_Delivery from PRODUCT_VARIANT_Staging where Product_Id = ||Product Id||) pv
order by Product_Delivery
FOR XML PATH('')), 1, 1, ''), '') as Product_Delivery_Rollup
```

CN_Target_Value_Source_Restricted_Name
Product_Delivery_Rollup

PRODUCT_Staging is identified as the CN_Target_Repository. This is the not the same repository that is the trigger repository. This means the target record has to be identified by CN_Target_SQL or CN_Target_Link_Relationship. The target PRODUCT_Staging record for the triggered PRODUCT_VARIANT_Staging record is identified by the following SQL:

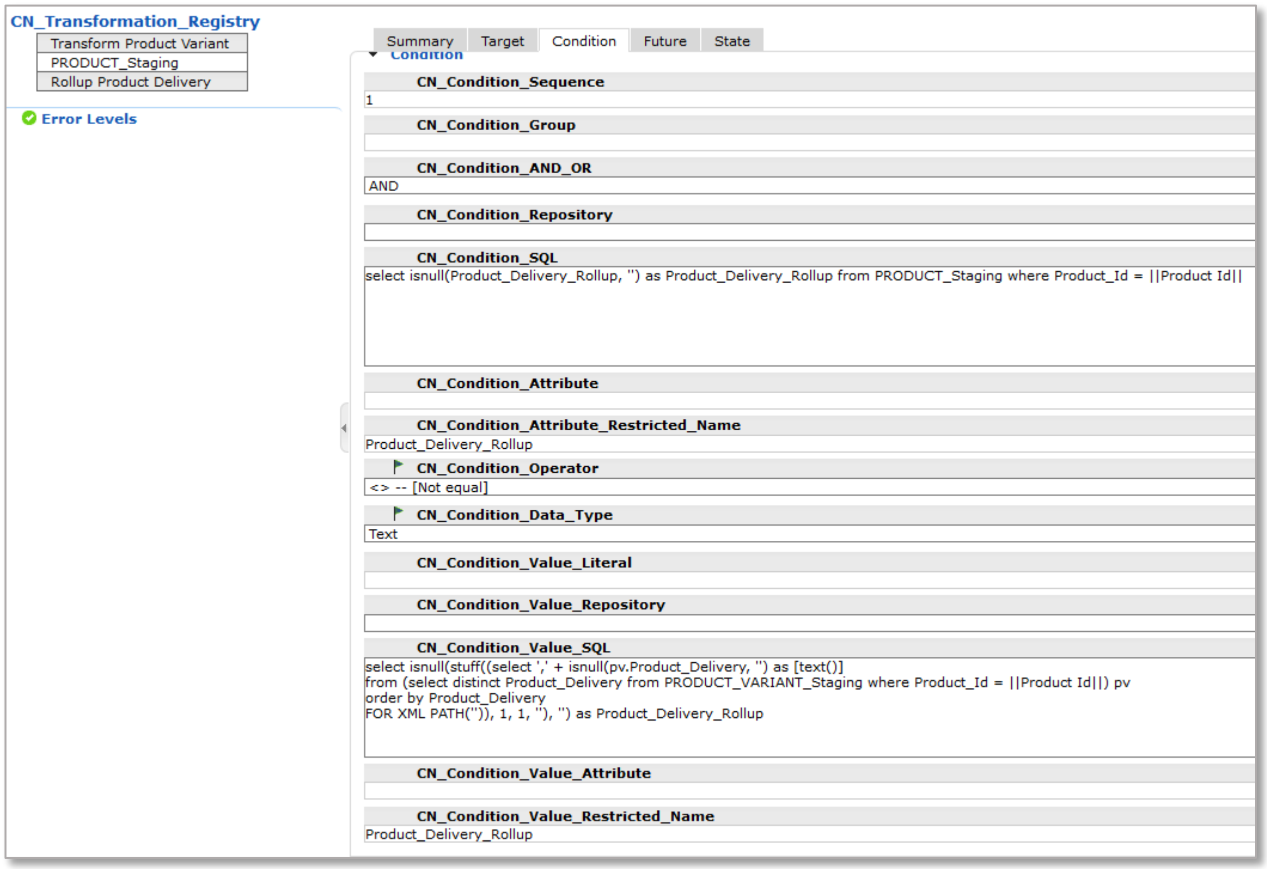
```
select InternalRecordId from PRODUCT_Staging where Product_Id =
||Product Id||
```

The source value for the target is defined by a SQL query that produces a delimited list of the unique Product Delivery attributes in the linked PRODUCT_VARIANT_Staging records using the following SQL Query:

```
select isnull(stuff((select ',' + isnull(pv.Product_Delivery, '')
as [text()]
from (select distinct Product_Delivery from
PRODUCT_VARIANT_Staging where Product_Id = ||Product Id||) pv
order by Product_Delivery
FOR XML PATH('')), 1, 1, ''), '') as Product_Delivery_Rollup
```

The SQL query ensures only the unique values are included (distinct) and they are sorted alphabetically.

Since the update is only necessary if the calculated list of values differs from the actual list of values, the condition for the transformation rule compares the two and requires that they are not equal:



CN_Condition_Sequence
1

CN_Condition_Group

CN_Condition_AND_OR
AND

CN_Condition_Repository

CN_Condition_SQL

```
select isnull(Product_Delivery_Rollup, '') as Product_Delivery_Rollup from PRODUCT_Staging where Product_Id = ||Product Id||
```

CN_Condition_Attribute

CN_Condition_Attribute_Restricted_Name
Product_Delivery_Rollup

CN_Condition_Operator
<> -- [Not equal]

CN_Condition_Data_Type
Text

CN_Condition_Value_Literal

CN_Condition_Value_Repository

CN_Condition_Value_SQL

```
select isnull(stuff((select ',' + isnull(pv.Product_Delivery, '') as [text()]
from (select distinct Product_Delivery from PRODUCT_VARIANT_Staging where Product_Id = ||Product Id||) pv
order by Product_Delivery
FOR XML PATH('')), 1, 1, ''), '') as Product_Delivery_Rollup
```

CN_Condition_Value_Attribute

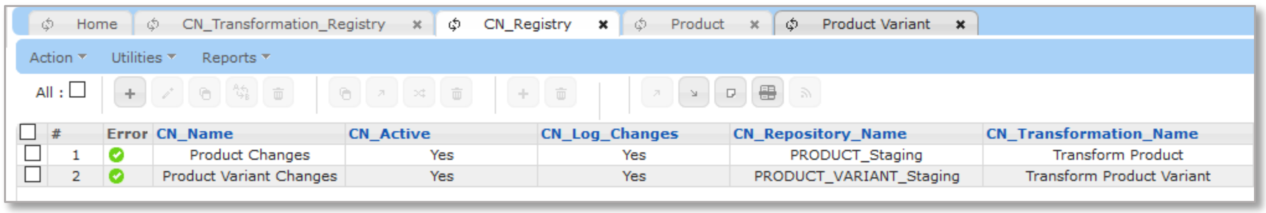
CN_Condition_Value_Restricted_Name
Product_Delivery_Rollup

The attribute to be compared on the trigger record is Product Delivery Rollup. Since the condition attribute is not in the triggered record, it must be retrieved from the PRODUCT_Staging record using the restricted name and the following SQL:

```
select isnull(Product_Delivery_Rollup, '') as
Product_Delivery_Rollup from PRODUCT_Staging where Product_Id =
||Product Id||
```

The value to be compared is the calculated rollup value defined by the same SQL that is used to populate the target attribute. Since the SQL for the target and condition are identical it is only executed once per event and the cached results used for the condition compare and the target value population.

The CN_Registry repository needs an entry for each repository to associate a triggered repository record to the appropriate Transformation:



#	Error	CN_Name	CN_Active	CN_Log_Changes	CN_Repository_Name	CN_Transformation_Name
1	✓	Product Changes	Yes	Yes	PRODUCT_Staging	Transform Product
2	✓	Product Variant Changes	Yes	Yes	PRODUCT_VARIANT_Staging	Transform Product Variant

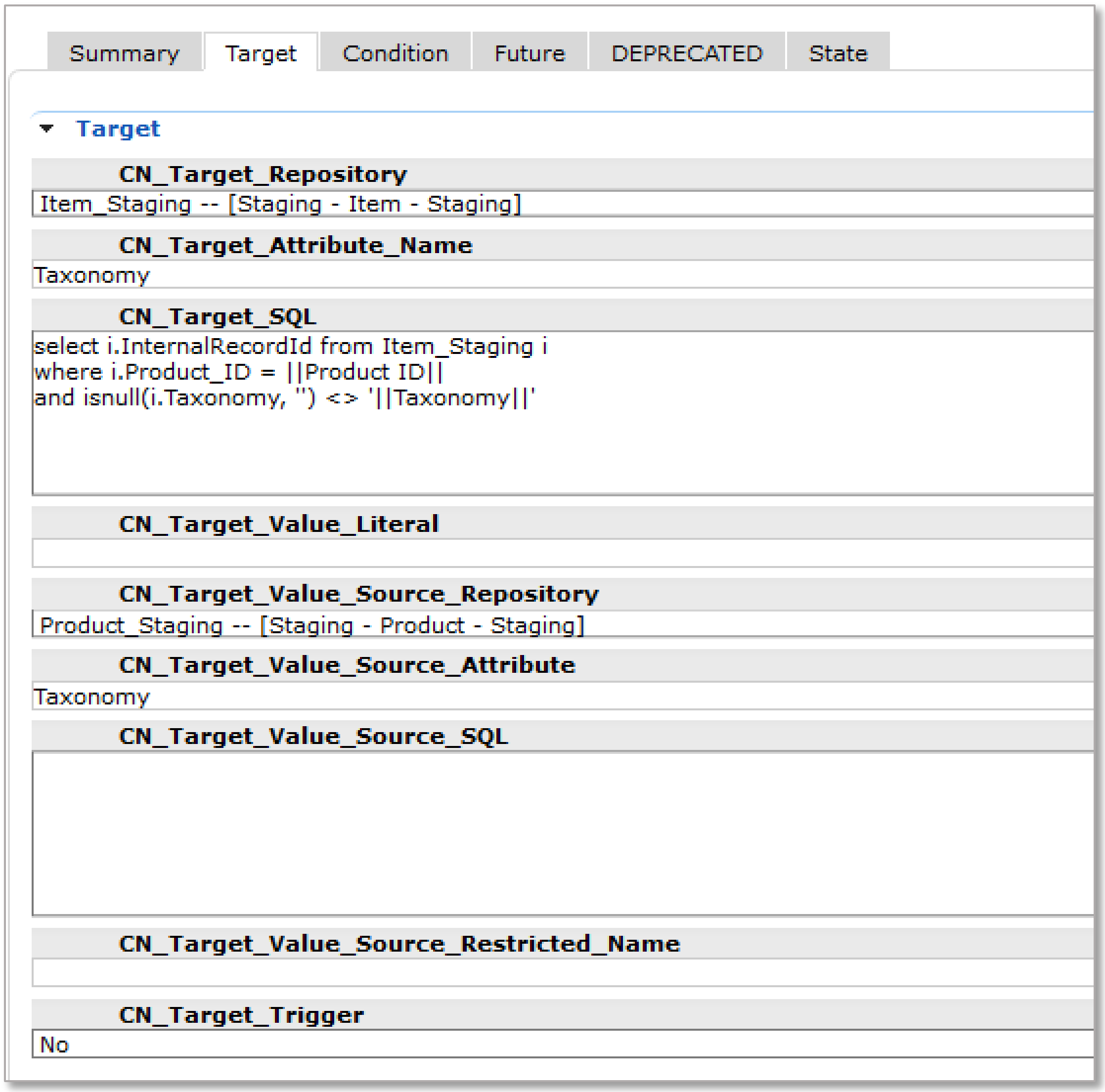
If in the future additional transformations were needed for the PRODUCT_Staging and/or PRODUCT_VARIANT_Staging repositories, more entries would be added to CN_Transformation_Registry using the appropriate value for the CN_Transformation_Name attribute.

Cascade Attribute

When a parent repository record shares an attribute with linked child (or grandchild) repository records, a change to the value in the parent repository record needs to be cascaded to all of the children/grandchildren. This may be needed to facilitate searches (to overcome the Enable limitation of searching on attributes on the current or linked repository when the values to be searched are more than one “jump” away. Another common use-case is when the parent and child (or grandchild) records must have the same Taxonomy node assignment.

In such cases, the CN_Transformation_Registry record must be assigned to the parent repository and use the CN_Target_SQL attribute to identify the record(s) to be updated with the new value in the parent record.

For example, if the Product_Staging repository has a one-to-many relationship with Item_Staging and both repositories have the Taxonomy attribute and the value of the Taxonomy in the Item_Staging records must match the value of the Taxonomy in the Product_Staging record, the CN_Transformation_Registry entry for the Product_Staging repository must identify the linked Item_Staging records for the target:



Target

CN_Target_Repository
Item_Staging -- [Staging - Item - Staging]

CN_Target_Attribute_Name
Taxonomy

CN_Target_SQL
```
select i.InternalRecordId from Item_Staging i
where i.Product_ID = ||Product ID||
and isnull(i.Taxonomy, '') <> '||Taxonomy||'
```

CN_Target_Value_Literal

CN_Target_Value_Source_Repository
Product_Staging -- [Staging - Product - Staging]

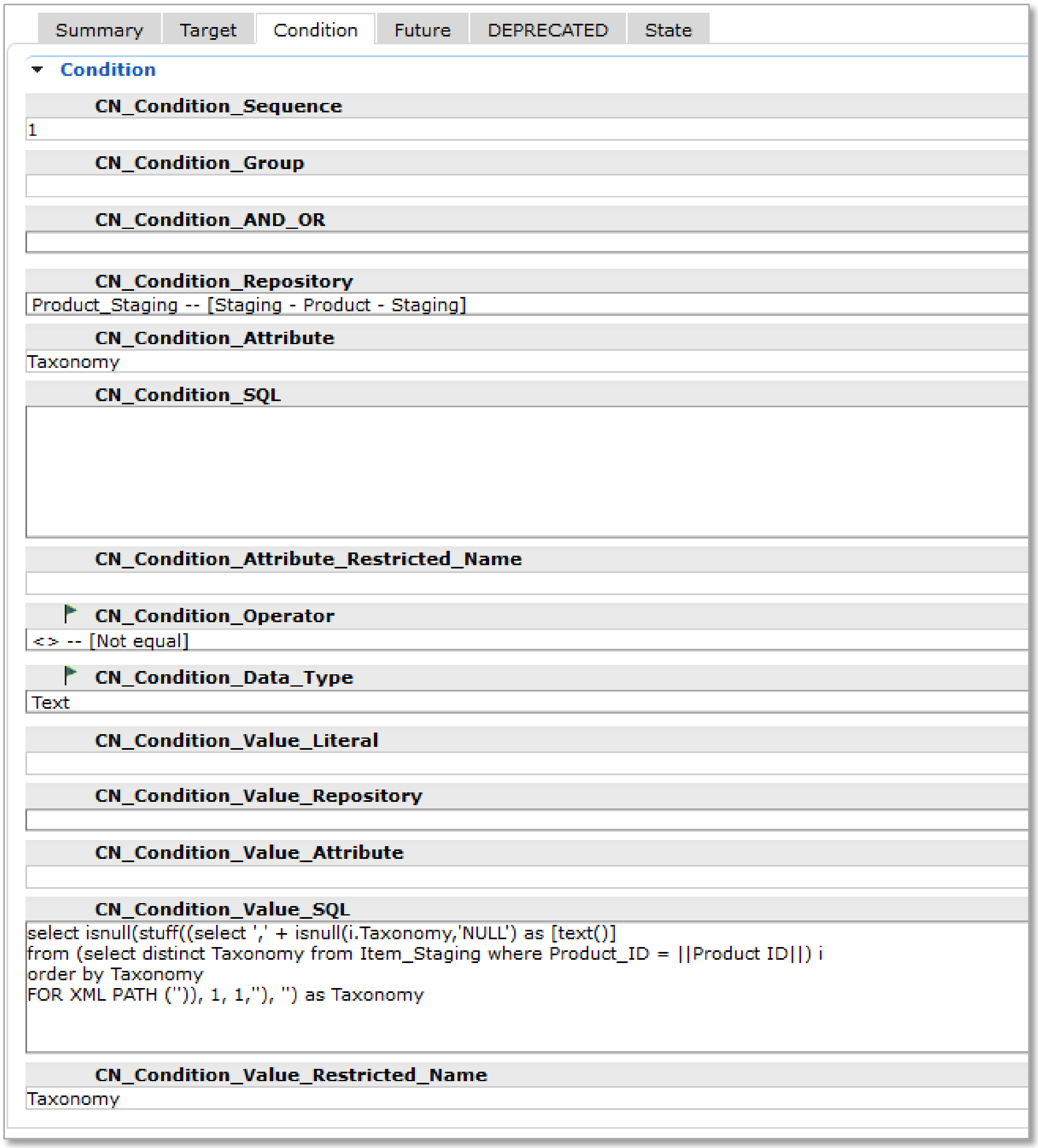
CN_Target_Value_Source_Attribute
Taxonomy

CN_Target_Value_Source_SQL

CN_Target_Value_Source_Restricted_Name

CN_Target_Trigger
No

In the above example, the SQL identifies all linked Item_Staging records for which the Taxonomy value does NOT match the Product_Staging's Taxonomy value. If an Item_Staging record has a matching value, then it is not updated. Since the Target SQL will only update the Item_Staging records that need to be updated, the Condition for the rule needs only to determine if ANY Item_Staging record needs to be updated. This can be accomplished by performing a rollup query on the linked Item_Staging records to create a delimited list of unique values. If any Item_Staging is missing the Taxonomy value or the value does not match the Product_Staging Taxonomy value, the computed condition value will not equal the Product_Staging Taxonomy value:


Summary | Target | Condition | Future | DEPRECATED | State

Condition

CN_Condition_Sequence
1

CN_Condition_Group

CN_Condition_AND_OR

CN_Condition_Repository
Product_Staging -- [Staging - Product - Staging]

CN_Condition_Attribute
Taxonomy

CN_Condition_SQL

CN_Condition_Attribute_Restricted_Name

CN_Condition_Operator
<> -- [Not equal]

CN_Condition_Data_Type
Text

CN_Condition_Value_Literal

CN_Condition_Value_Repository

CN_Condition_Value_Attribute

CN_Condition_Value_SQL
```
select isnull(stuff((select ',' + isnull(i.Taxonomy,'NULL') as [text()]
from (select distinct Taxonomy from Item_Staging where Product_ID = ||Product ID||) i
order by Taxonomy
FOR XML PATH ('')), 1, 1,''), '') as Taxonomy
```

CN_Condition_Value_Restricted_Name
Taxonomy

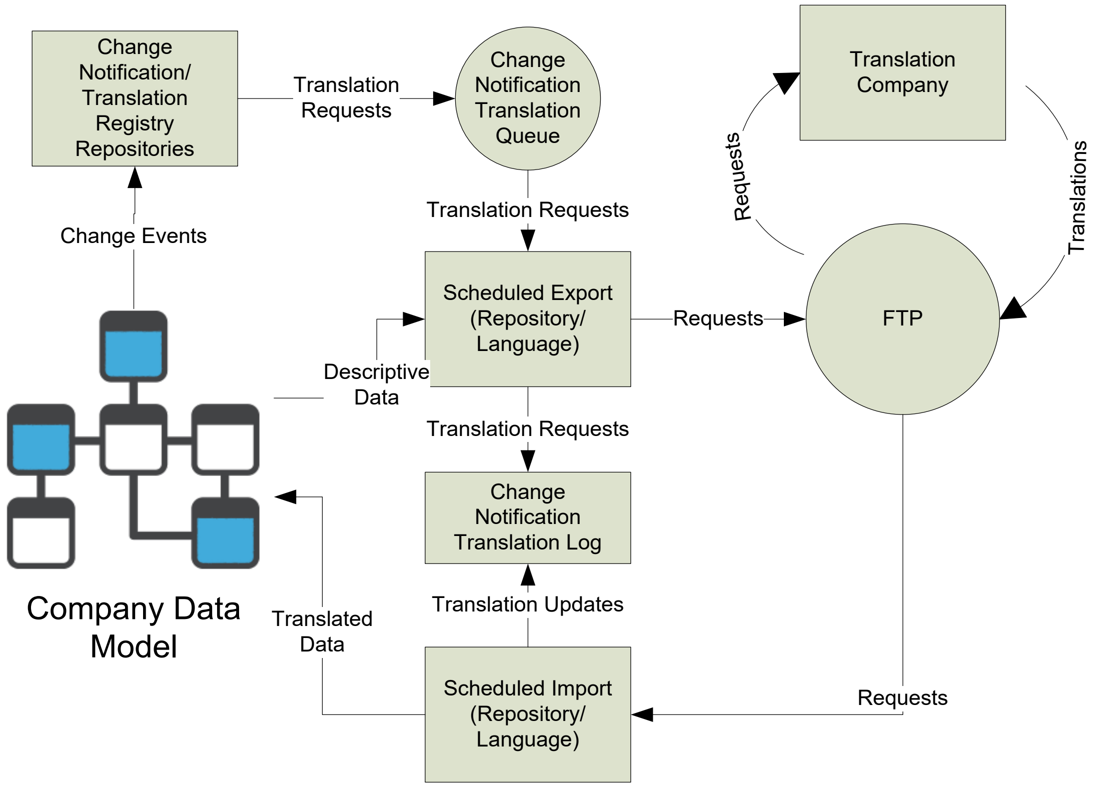
If all linked Item_Staging records have the same Taxonomy value AND that value matches the Product_Staging Taxonomy value, then the Condition for the rule is not met and no update takes place.

Language Translation Extension

Overview

The Language Translation Extension functionality leverages the Change Notification functionality to facilitate having a third-party generate language translations for changes to data within Enable.

The following diagram shows the basic flow of data for the Language Translation process:


Change Notification/ Translation Registry Repositories
Translation Requests
Change Notification Translation Queue
Translation Company
Requests
Translations
Change Events
Translation Requests
Scheduled Export (Repository/ Language)
Requests
FTP
Descriptive Data
Translation Requests
Change Notification Translation Log
Company Data Model
Translated Data
Translation Updates
Scheduled Import (Repository/ Language)
Requests

The Change Notification/Translation registry repositories identify which repositories and the attributes within the are to be translated. When a change is made to data in the Company Data Model (data repositories), a change event is generated. If this event is associated with a registered translation, a Translation Request is submitted to an internal Change Notification Translation queue.

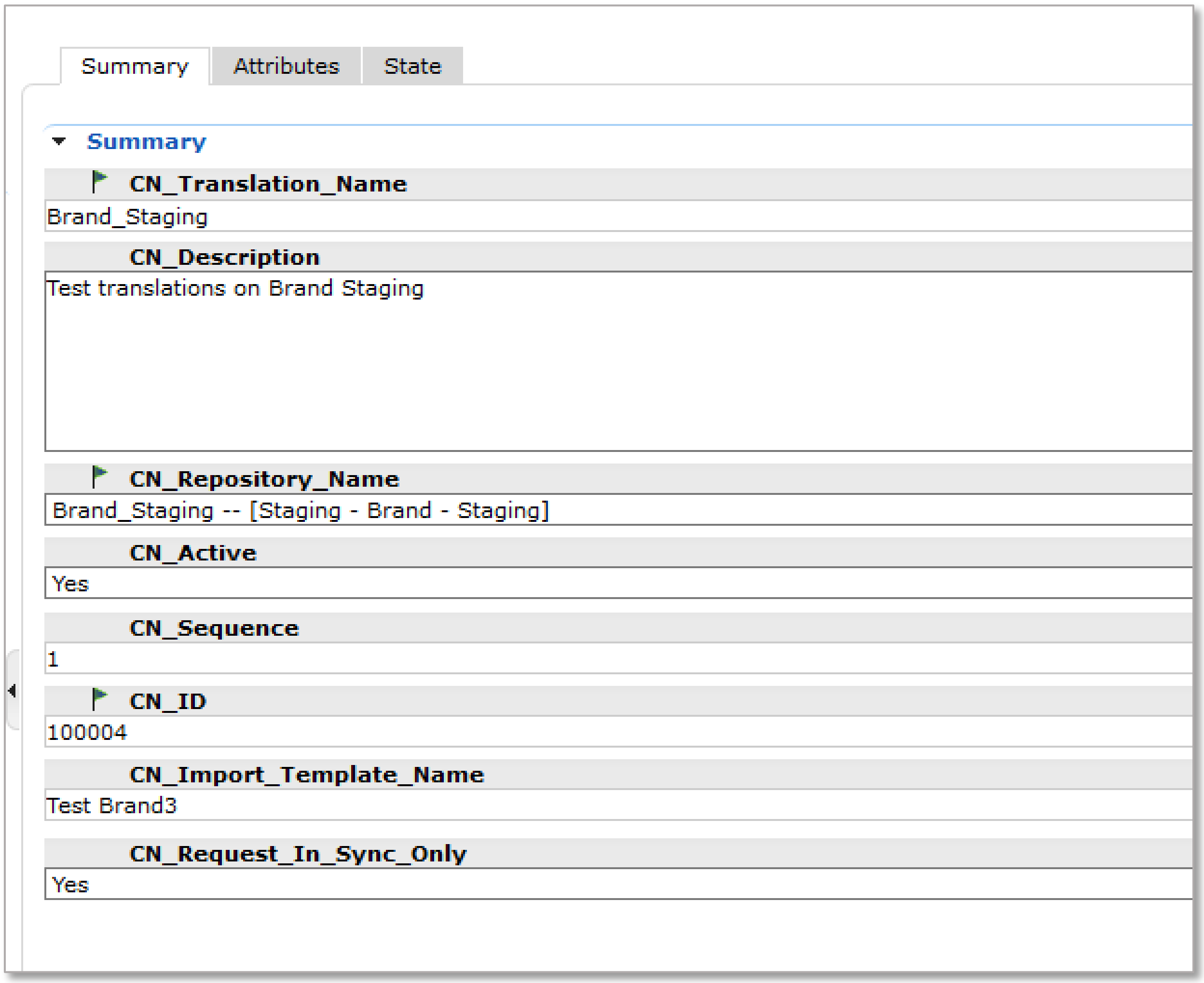
On a periodic basis, such as nightly, a Scheduled Export is launched that retrieves the most-recent translation requests from the Change Notification Translation Queue along with optional descriptive data from the Company data and generates two files. The first file is submitted to the Translation Company via FTP (this can also be Email or Network file directory). The second file updates the Change Notification Translation Log to identify which repositories, records, and attributes have been submitted for translation.

The translation company retrieves and processes each translation file, returning a translated file to the FTP server (or network file location or configured Email address). A scheduled import retrieves each update file and pre-processes it into two import files. The first file updates the designated language attributes in the Company data repository. The second file updates the Change Notification Translation Log repository to indicate translations have been received for the requests.

Change Notification Translation Registry

The Change Notification Translation Registry repository defines the details for all language translations. A separate record must be defined for each repository requiring translation, but more than one record may be defined for the same repository. The following sections provide the details for the repository attributes by tab:

Summary Tab



CN_Translation_Name	Brand_Staging
CN_Description	Test translations on Brand Staging
CN_Repository_Name	Brand_Staging -- [Staging - Brand - Staging]
CN_Active	Yes
CN_Sequence	1
CN_ID	100004
CN_Import_Template_Name	Test Brand3
CN_Request_In_Sync_Only	Yes

CN_Translation_Name - Name of the translation. Used to identify the translation request files sent to the translation company. Will usually match the repository name but may identify part number if multiple translations are needed due to a large number of multi-language attributes. This name must be selected on the Translation tab in a Change Notification Registry record.

NOTE: When a new CN_Translation_Registry record with a new CN_Translation_Name is defined, the data cache must be cleared before it becomes visible in the Translation tab of the CN_Registry detail editor.

CN_Description – Optional description of translation (user information only). It is strongly recommended that a detailed description be specified for better maintainability.

CN_Repository_Name - Name of repository containing the multi-language attributes to be translated. While the drop-down selection list shows all repositories defined in Enable, only the ones that have been registered with the Change Notification trigger will be processed.

CN_Active - Indicates the registry entry is active if Yes.

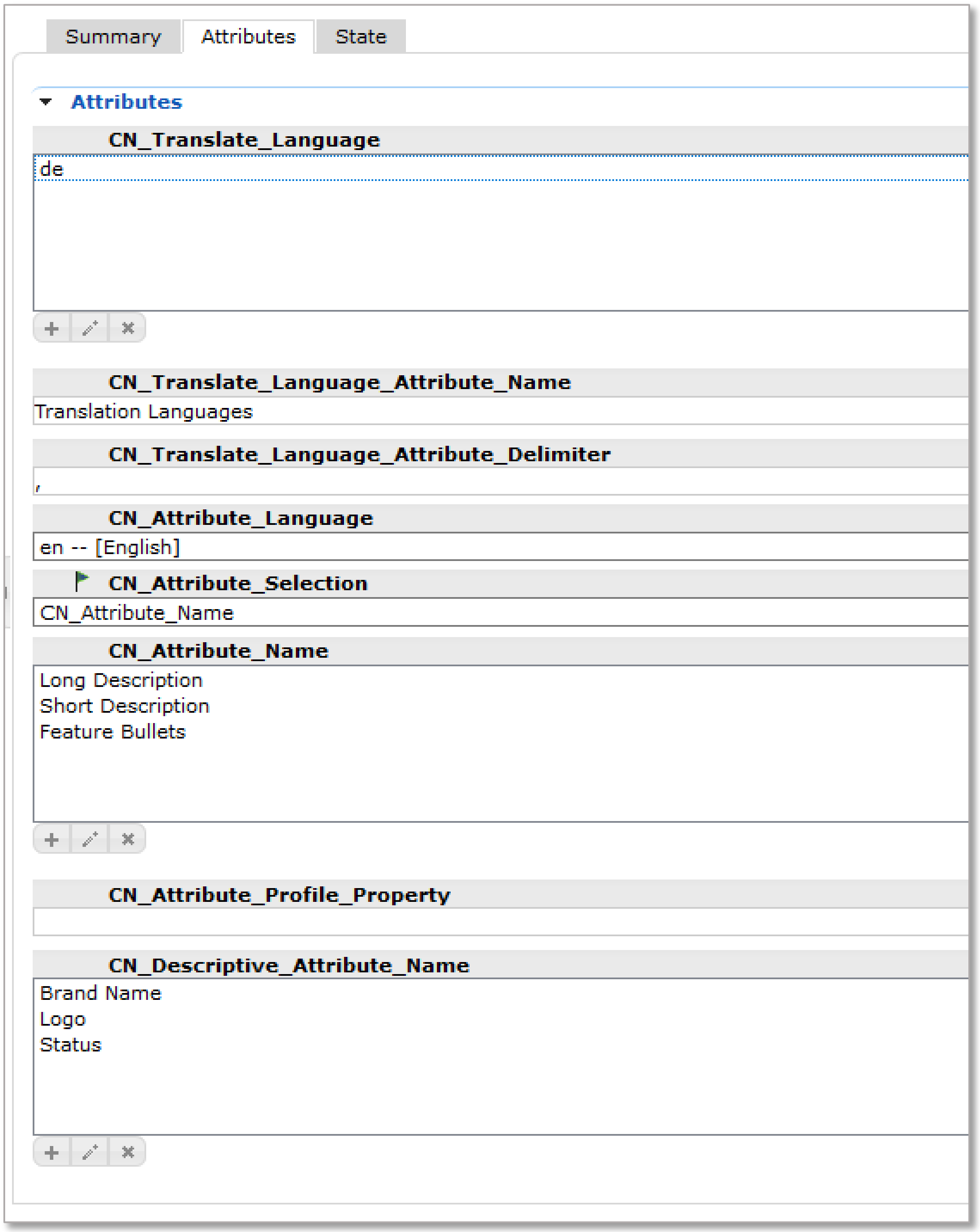
CN_Sequence - Sequence number to be used to control the order the registry entries are processed.

CN_ID - Sequential ID for each record.

CN_Import_Template_Name - Name of Import Template to be used to update the multi-language attributes in the designated repository. The drop-down selection list will show all import templates defined for the repository selected for CN_Repository_Name.

CN_Request_In_Sync_Only – Include only language translation request for records that have not been changed since the last time they were promoted if Yes. This ensures language translation requests are only transmitted on approved, promoted data.

Attributes Tab



Attributes

CN_Translate_Language
de

CN_Translate_Language_Attribute_Name
Translation Languages

CN_Translate_Language_Attribute_Delimiter
,

CN_Attribute_Language
en -- [English]

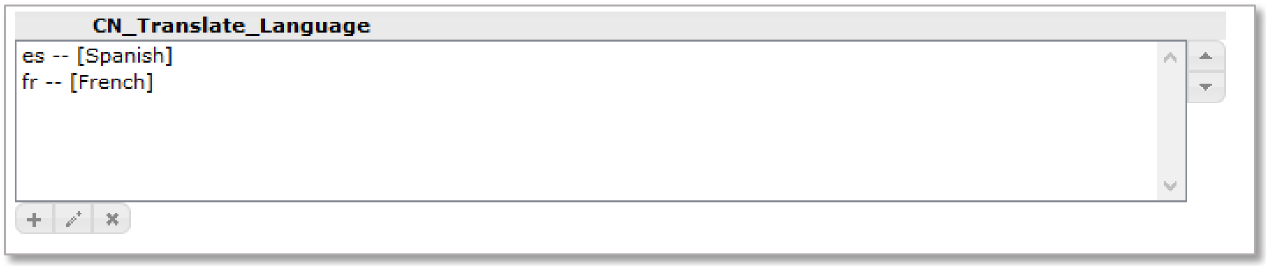
CN_Attribute_Selection
CN_Attribute_Name

CN_Attribute_Name
Long Description
Short Description
Feature Bullets

CN_Attribute_Profile_Property

CN_Descriptive_Attribute_Name
Brand Name
Logo
Status

CN_Translate_Language - Delimited list of language extensions for the target languages for this translation. For example, to have the attributes translated to Spanish and French, the es and fr language extensions must be selected:


CN_Translate_Language

es -- [Spanish]
fr -- [French]

At least one language must be selected if CN_Translate_Attribute_Name is empty.

CN_Translate_Language_Attribute_Name - Name of attribute in repository containing multi-language attributes that contains a delimited list of target language extensions. This allows each record in the repository to self-identify which language translations are needed. Required if CN_Translate_Language is empty.

CN_Translate_Language_Attribute_Delimiter - Delimiter for the language values in the attribute specified by CN_Translate_Language_Attribute_Name. The default delimiter is the comma.

CN_Attribute_Language - Language extension for the source language for this translation. If blank, the default language will be used. For example, if changes to the French version of an attribute are to be translated, this attribute must be set to French.

CN_Attribute_Selection – Specifies the source selection for the list of attributes to be translated:

- **All Multi-Language** – translate all multi-language attributes in the designated repository
- **CN_Attribute_Name** – translate only the attributes listed in the CN_Attribute_Name attribute
- **CN_Attribute_Profile_Property** – translate only the attributes with the BOOLEAN profile property identified in the CN_Attribute_Profile_Property attribute set to a value of 1 (true).

CN_Attribute_Name – Names of the multi-language attributes to be translated. The names must exactly match (including case) the names of the attributes defined in the target repository.

CN_Attribute_Profile_Property – Name of the profile property assigned to the designated repository to be used to identify the attributes to be translated, each with a property value of 1 (TRUE)

CN_Descriptive_Attribute_Name - List of descriptive attributes to be included in translation request files. These would typically be attributes that are not multi-language themselves, but help the Translation Company better understand the records being translated.

Change Notification Translation Queue

The Change Notification Translation Queue is an Enable database table with the following columns.

Column Name	Description
seqNumber	Sequential number for each record. Primary key for table.

Column Name	Description
batchId	Set to the export job number for the queued records for a repository and language. Used to process the set of records as a batch in case additional requests are queued during the update process, which avoids overlaps (sending the same request twice) and gaps.
translationName	Name of the Change Notification Translation Registry definition used to populate this queue entry. In most cases, the translationName will likely match the repositoryName. But if the repository has multiple parts or there are different Change Notification Translation Registry definitions for the same repository (perhaps different languages are translated by different companies), the translation name would uniquely identify each translation. For example, if Brand had more than 1000 multi-language attributes, they may be split between two parts having translationName of “Brand1” and “Brand2”.
repositoryName	Name of repository for the request
internalRecordId	Internal record ID for the repository record
attributeName	Name of the attribute to be translated
oldTranslatedValue	(FUTURE) Previous translated value
newValue	New value that needs to be translated
changedby	Login of the user who changed the record containing the attribute.
changedDatetime	Date and time of when the change was made.
languageExt	Language extension for the value being translated. Implies the default languate (e.g., English) if null.
translateLanguageExt	Target language extension

When a repository record (containing multi-language attributes registered for translation) is modified, each registered multi-language attribute that was changed generates a record in this queue table. If the same attribute is changed multiple times, the queue will contain multiple records – one for each change. When these records are retrieved from the queue, only the entry with the newest date and time will be sent to the translation company. For example, if the Long Description in the Brand repository is registered for translation, and the Enterworks brand description is changed to “Enterworks”, then “Enterworks Incorporated”, and finally “Enterworks Acquisition, Inc.” all in the same day, the export for Brand will only include the “Enterworks Acquisition, Inc.” value. The same changes were made over successive days (and the export is scheduled to run once per day), each change will have been sent to the translation company. If the translation company has more than a one-day turnaround, it would be responsible for either translating each value received or only translating the last value received for the same record and attribute.

The final step in the export process removes the records from the Change Notification Translation Queue table.

Change Notification Translation Log Repository

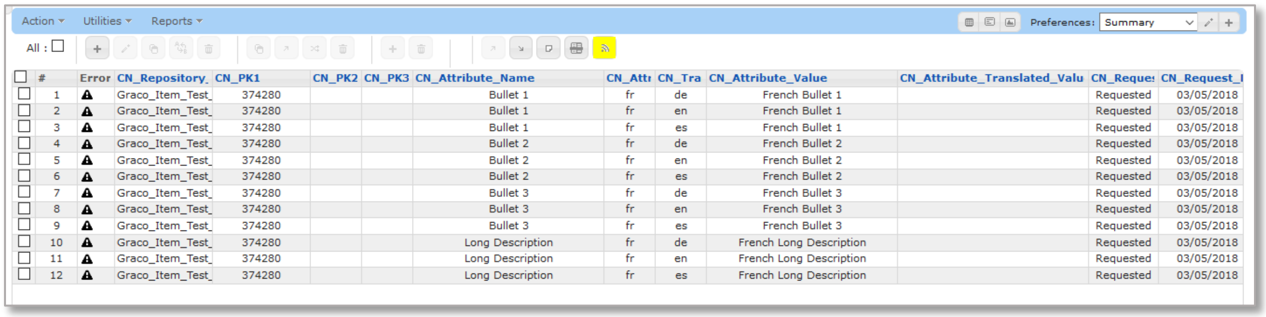
The Change Notification Translation Log repository contains the details of each language translation request and translated response. Each record in the repository represents a single language for a single attribute in a repository record. The Scheduled Exports update this repository with the details about the record, attribute, and target language, including a request status of Requested along with the date the request was sent to the translation company. This update is accomplished by the Scheduled Export invoking a dependent operation for a Scheduled Import that is set up to create or update the CN_Translation_Log records for any repository for both the requests and responses.

The Scheduled Imports also update this repository, setting the request status to Updated along with the date the update was received. The translated value is also stored in the same record. If a specific attribute value for a specific record is changed and translated frequently, the same entry in the Translation Log repository will be updated with alternating Request Status values of “Requested” and “Updated.

Attribute Name	Description
CN_Attribute_Language	Specifies the language extension for the source attribute. Part of the primary key.
CN_Attribute_Name	Name of the multi-language attribute. Part of the primary key.
CN_Attribute_Translated_Value	Translated value from the translation company
CN_Attribute_Value	Source value for the attribute to be translated
CN_ID	Unique ID for the record. While not part of the primary key, this can be used to access specific records without having to specify all four primary key attributes.
CN_Internal_Record_ID	Identifies a record in a staging repository. Part of the primary key.
CN_PK1	Repository primary key1 value.
CN_PK2	Repository primary key2 value.
CN_PK3	Repository primary key3 value.
CN_PK4	Repository primary key4 value.

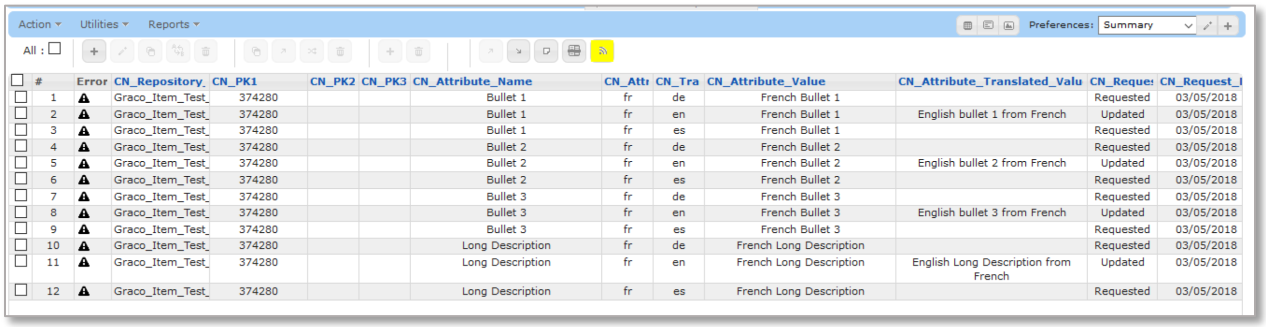
Attribute Name	Description
CN_PK5	Repository primary key5 value.
CN_Repository_Name	Name of the repository. This attribute is not part of the primary key since the internal record ID uniquely identifies each record, regardless of repository.
CN_Requested_Date	Date at which a translation request was sent or a translation update was received.
CN_Request_Status	Indicates the request status for this record: • Requested - request has been sent to the translation company • Updated – a translation update has been received from the translation company.
CN_Translate_Language	Language extension for the target language. Part of the primary key.

When attributes have been submitted to the language translation company, they will have a request status of "Requested":



#	Error	CN_Repository_	CN_PK1	CN_PK2	CN_PK3	CN_Attribute_Name	CN_Attr	CN_Tra	CN_Attribute_Value	CN_Attribute_Translated_Valu	CN_Reque	CN_Request_I
1	▲	Graco_Item_Test_	374280			Bullet 1	fr	de	French Bullet 1		Requested	03/05/2018
2	▲	Graco_Item_Test_	374280			Bullet 1	fr	en	French Bullet 1		Requested	03/05/2018
3	▲	Graco_Item_Test_	374280			Bullet 1	fr	es	French Bullet 1		Requested	03/05/2018
4	▲	Graco_Item_Test_	374280			Bullet 2	fr	de	French Bullet 2		Requested	03/05/2018
5	▲	Graco_Item_Test_	374280			Bullet 2	fr	en	French Bullet 2		Requested	03/05/2018
6	▲	Graco_Item_Test_	374280			Bullet 2	fr	es	French Bullet 2		Requested	03/05/2018
7	▲	Graco_Item_Test_	374280			Bullet 3	fr	de	French Bullet 3		Requested	03/05/2018
8	▲	Graco_Item_Test_	374280			Bullet 3	fr	en	French Bullet 3		Requested	03/05/2018
9	▲	Graco_Item_Test_	374280			Bullet 3	fr	es	French Bullet 3		Requested	03/05/2018
10	▲	Graco_Item_Test_	374280			Long Description	fr	de	French Long Description		Requested	03/05/2018
11	▲	Graco_Item_Test_	374280			Long Description	fr	en	French Long Description		Requested	03/05/2018
12	▲	Graco_Item_Test_	374280			Long Description	fr	es	French Long Description		Requested	03/05/2018

When the translation response is processed, the records are updated to show a status of "Updated" and also shows the translated values:



#	Error	CN_Repository_	CN_PK1	CN_PK2	CN_PK3	CN_Attribute_Name	CN_Attr	CN_Tra	CN_Attribute_Value	CN_Attribute_Translated_Valu	CN_Reque	CN_Request_I
1	▲	Graco_Item_Test_	374280			Bullet 1	fr	de	French Bullet 1		Requested	03/05/2018
2	▲	Graco_Item_Test_	374280			Bullet 1	fr	en	French Bullet 1	English bullet 1 from French	Updated	03/05/2018
3	▲	Graco_Item_Test_	374280			Bullet 1	fr	es	French Bullet 1		Requested	03/05/2018
4	▲	Graco_Item_Test_	374280			Bullet 2	fr	de	French Bullet 2		Requested	03/05/2018
5	▲	Graco_Item_Test_	374280			Bullet 2	fr	en	French Bullet 2	English bullet 2 from French	Updated	03/05/2018
6	▲	Graco_Item_Test_	374280			Bullet 2	fr	es	French Bullet 2		Requested	03/05/2018
7	▲	Graco_Item_Test_	374280			Bullet 3	fr	de	French Bullet 3		Requested	03/05/2018
8	▲	Graco_Item_Test_	374280			Bullet 3	fr	en	French Bullet 3	English bullet 3 from French	Updated	03/05/2018
9	▲	Graco_Item_Test_	374280			Bullet 3	fr	es	French Bullet 3		Requested	03/05/2018
10	▲	Graco_Item_Test_	374280			Long Description	fr	de	French Long Description		Requested	03/05/2018
11	▲	Graco_Item_Test_	374280			Long Description	fr	en	French Long Description	English Long Description from French	Updated	03/05/2018
12	▲	Graco_Item_Test_	374280			Long Description	fr	es	French Long Description		Requested	03/05/2018

Translation Request File

The Translation Request File contains the translation requests for one or more attributes for one or more records for a specific repository. The file name does not need to follow any specific naming convention since its contents are self-identifying but it's recommended as a best practice. Each file name should include the following information:

- Repository Name (e.g., Brand, SKU, Item, etc.). There's no need to distinguish between Staging and Production since all translations should be performed on Staging only.

- Source Language Extension – if it's possible for anything but the default language (e.g., English) to be translated, the source language for the file should be included in the file name (e.g., fr, sp, de, etc.)

- Target Language Extension – the extension of the target language (e.g., fr, sp, de, etc.)

- Unique Identifier – A unique identifier to make the resulting file unique (e.g., Export Job attribute value).

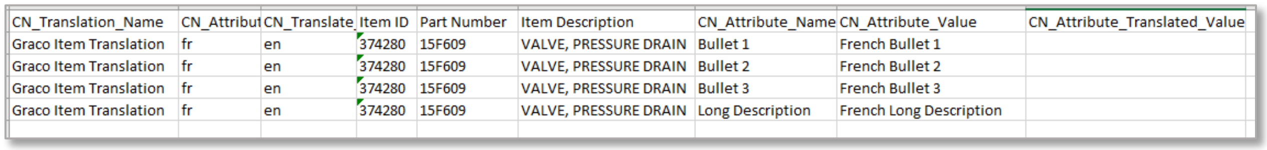
An example file format would be: Brand_en_fr_12345.xlsx – which would indicate the file is for the Brand repository and contain values in English to be translated to French.

The request file must have the following columns:

Column Name	Description
CN_Translation_Name	Name of the translation. This name is defined in the CN_Translation_Name column in the Change Notification Translation Registry. By using this name instead of the actual repository name, multiple translations can be defined for the same repository. For example, this may be necessary if there are multiple parts due to a large number of multi-language attributes (e.g., Brand1, Brand2, etc.)
CN_Attribute_Language	Extension of the source language (e.g., en, sp, de, etc.)
CN_Translate_Language	Extension of the target language (e.g., en, sp, de, etc.)
<primaryKeyAttribute>	One column for each primary key attribute, the the column names containing the attribute name (vs. the generic PK1, PK2, etc.). One column per attribute.
<descriptiveAttribute>	Optional descriptive attributes. Determined by the settings in the Translation Registry. One column per attribute.
CN_Attribute_Name	Name of the attribute to be translated

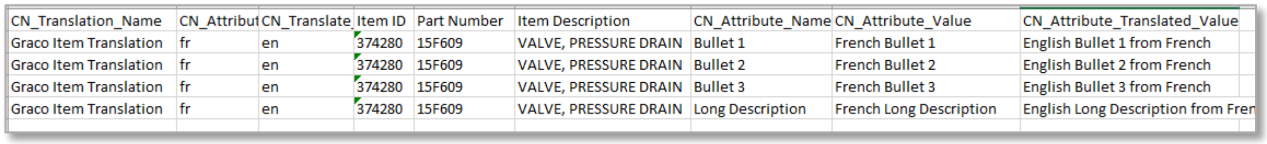
Column Name	Description
CN_Attribute_Value	Value that needs to be translated
Old Translated Value	[FUTURE] Current translated value for attribute (would be blank for new records).
CN_Attribute_Translated_Value	Translated value (populated by translation company)

The following sample file shows translation requests from French to English:



CN_Translation_Name	CN_Attribut	CN_Translate	Item ID	Part Number	Item Description	CN_Attribute_Name	CN_Attribute_Value	CN_Attribute_Translated_Value
Graco Item Translation	fr	en	374280	15F609	VALVE, PRESSURE DRAIN	Bullet 1	French Bullet 1	
Graco Item Translation	fr	en	374280	15F609	VALVE, PRESSURE DRAIN	Bullet 2	French Bullet 2	
Graco Item Translation	fr	en	374280	15F609	VALVE, PRESSURE DRAIN	Bullet 3	French Bullet 3	
Graco Item Translation	fr	en	374280	15F609	VALVE, PRESSURE DRAIN	Long Description	French Long Description	

The same file format is used for the translated files returned from the Translation company. The only difference between the two files is the request file will have an empty Translated Value column and the translated files will have it populated:



CN_Translation_Name	CN_Attribut	CN_Translate	Item ID	Part Number	Item Description	CN_Attribute_Name	CN_Attribute_Value	CN_Attribute_Translated_Value
Graco Item Translation	fr	en	374280	15F609	VALVE, PRESSURE DRAIN	Bullet 1	French Bullet 1	English Bullet 1 from French
Graco Item Translation	fr	en	374280	15F609	VALVE, PRESSURE DRAIN	Bullet 2	French Bullet 2	English Bullet 2 from French
Graco Item Translation	fr	en	374280	15F609	VALVE, PRESSURE DRAIN	Bullet 3	French Bullet 3	English Bullet 3 from French
Graco Item Translation	fr	en	374280	15F609	VALVE, PRESSURE DRAIN	Long Description	French Long Description	English Long Description from Fren

If a request file contains more than one multi-language attribute for the same repository record, the attributes must be in consecutive rows in the file. Otherwise the pivot pre-processing done for the translated files will contain more than one record for the same repository record, resulting in multiple updates, which includes multiple history records.

The Services Language Translation supports the translation of up to the number of multi-language attributes in a single repository determined by the following formula:

1,000 – (number_of_primary_key_attributes)

For example, if the primary key for a repository being translated contains 3 primary key attributes, then up to 997 multi-language attributes can be translated in a single group. If there are more multi-language attributes in a repository, additional imports and exports can be defined to accommodate them. Each part must be defined as a separate translation (e.g., Brand1, Brand2) having its own Translation Registry entry, Scheduled Export, and Import Template. If the multi-language attributes are category specific, the number of attributes that can be included in a single import may be further limited.

Translation Request Scheduled Export

A separate Scheduled Export must be defined for each repository and language to be translated. For example, if the Brand repository has multi-language attributes and the active languages are English (default), Spanish, French, German, and Chinese, there needs to be four Translation Request scheduled exports defined to translate from English to Spanish, French, German and Chinese. If the Item repository also contains multi-language attributes that need to be translated to the same languages, four additional Translation Request scheduled exports must be defined.

Each Scheduled Export should have the following settings:

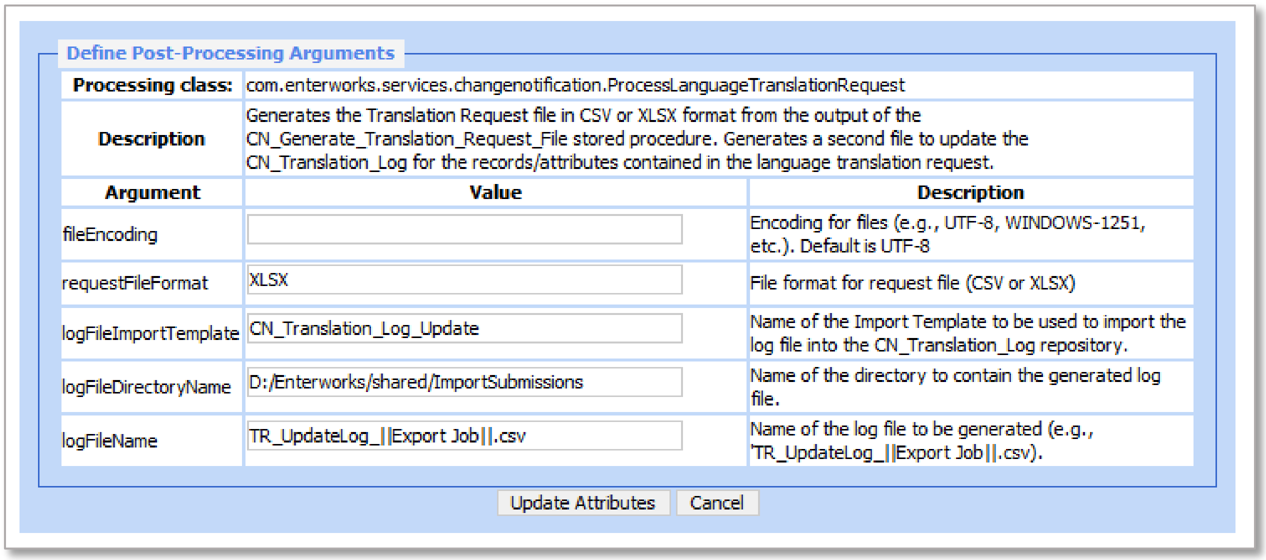
Attribute Name	Value
Export Name	Identifies the repository and language and that it is for language translation requests. It is recommended that the name for each of these exports is consistent and identifies the repository and language (e.g., Item Language Translation Request French).
Export Type	View
Postprocess File	Yes
Postprocess Class	com.enterworks.services.changenotification.ProcessLanguageTranslationRequest
Postprocess Keys Postprocess Values	See post process details below
Export Language	Set to the requested translation language
View SQL	See stored procedure details below
View Format	COMMA
Export File Name	LTR_<repo>_<language>_||Export Job||.csv – all translation request file names should follow the same pattern so the responses can be processed by the same Scheduled import without having to rename the files.
Dependent Operation Type	Import
Dependent Operation	CN_Request_Translation_Log
Dependent Operation Immediate	Yes
Dependent Keys	1. Import FileName 2. Parameter1

Attribute Name	Value
Dependent Values	1. TR_UpdateLog_||Export Job||.csv - (must match the setting in the logFileName argument to the ProcessLanguageTranslationRequest post-processing block) 2. ||Export Job|| - (must match the batchId parameter in the call to the stored procedure CN_Generate_Translation_Request_File)

The **com.enterworks.services.changenotificaion.ProcessLanguageTranslationRequest** post-processing block must be configured for each Scheduled Export. This post-processing block converts the exported file into a second file to be loaded into the Change Notification Translation Log repository. This second file contains all of the columns in the original file (described above) except for the descriptive attributes plus the request status of "Requested" and the current date for the Request Date. The post-processing block has the following configurable arguments:

Argument	Description
fileEncoding	Specifies the encoding for files. The default is UTF-8
requestFileFormat	Format for the translation request file: XLSX or CSV
logFileImportTemplate	CN_Request_Translation_Log - Name of the import template used to update the CN_Translation_Log repository. This import template must be invoked by a Scheduled Import that is launched as a Dependent Operation for this export.
logFileDirectoryName	Location of where the Translation Log import files are to be placed for processing. The Scheduled Import must have this same directory name. Example: D:/Enterworks/shared/ImportSubmissions
logFileName	TR_UpdateLog_||Export Job||.csv. Name of the Translation Log import file. This name must match the Dependent Value for the Dependent Key "Import File Name"

The following screen shot shows an example configuration for the ProcessLanguageTranslationRequest post-processing arguments:

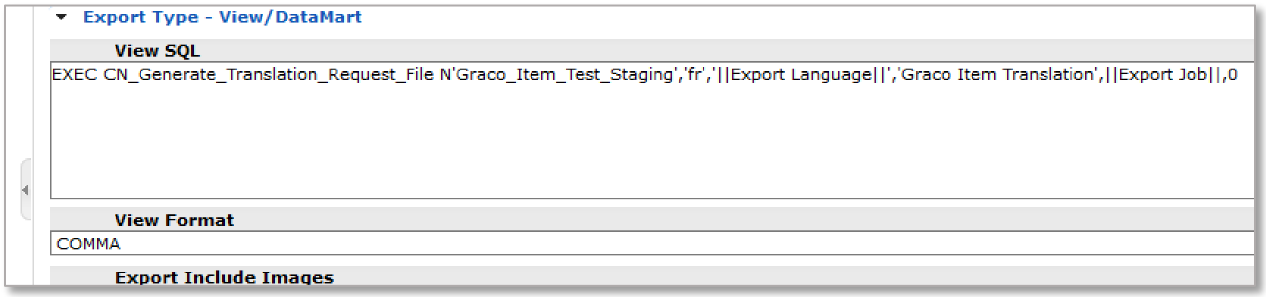


Processing class:	com.enterworks.services.changenotification.ProcessLanguageTranslationRequest	
Description	Generates the Translation Request file in CSV or XLSX format from the output of the CN_Generate_Translation_Request_File stored procedure. Generates a second file to update the CN_Translation_Log for the records/attributes contained in the language translation request.	
Argument	**Value**	**Description**
fileEncoding		Encoding for files (e.g., UTF-8, WINDOWS-1251, etc.). Default is UTF-8
requestFileFormat	XLSX	File format for request file (CSV or XLSX)
logFileImportTemplate	CN_Translation_Log_Update	Name of the Import Template to be used to import the log file into the CN_Translation_Log repository.
logFileDirectoryName	D:/Enterworks/shared/ImportSubmissions	Name of the directory to contain the generated log file.
logFileName	TR_UpdateLog_||Export Job||.csv	Name of the log file to be generated (e.g., 'TR_UpdateLog_||Export Job||.csv).

Each scheduled export invokes the SQL stored procedure CN_Generate_Translation_Request_File, passing in the following arguments:

Argument	Description
repositoryName	Name of the repository
languageExt	Extension of the source language (default if null)
translateLanguageExt	||Export Language|| - Extension of the target language
translationName	Name of the applicable translation definition in the Change Notification Translation Registry repository
batchId	||Export Job|| - A unique number
debugInd	0 - Output debug logging if 1

The following example shows the SQL for translating French to the language defined in the Export Language attribute for the language translation named: Graco Item Translation :


Export Type - View/DataMart

View SQL

```
EXEC CN_Generate_Translation_Request_File N'Graco_Item_Test_Staging','fr','||Export Language||','Graco Item Translation',||Export Job||,0
```

View Format

COMMA

Export Include Images

When called, the stored procedure performs the following steps:

1. Tags each record in the Change Notification Transformation Queue table with matching repository, source and destination language with the batchId.
2. Returns a result set of the newest translation for each attribute in each record being translated including the following data:
    a. repositoryName
    b. internalRecordId
    c. attributeName
    d. oldTranslatedValue (FUTURE)
    e. newValue
    f. changedby
    g. changedDatetime
    h. languageExt
    i. translateLanguageExt
    j. primary key columns as defined in the B_MASTER_REPOSITORY_ITEM repository (pk1 through pk5)
    k. descriptive attributes and their values from the referenced repository record

Once post-processing is complete, the original file is transmitted to the Translation Company in the manner defined in the Scheduled Export (e.g., by FTP, network file location, or e-mail). A scheduled import named "CN_Update_Translation_Log" to load the second file is launched to load the second file into the Change Notification Translation Log repository.

Translation Request Log Update Scheduled Import

The scheduled import should be named "CN_Request_Translation_Log" and must be configured to import an update file into the Change Notification Translation Log repository. It can be created from the export that is included in the Scheduled Framework deployment folder: **<deploymentFolder>\EPIM\Scheduled Imports**. It is launched by each Translation Request Scheduled Export that completes successfully. The Scheduled Import should have the following settings:

Attribute Name	Value
Import Name	CN_Request_Translation_Log
Import Type	Template
Import Directory Name	Must be the same directory specified in the Translation Request Scheduled Export post process argument logFileDirectoryName
Import File Name	<Populated by parent job>
Import Format	CSV
Move Import File	Yes
Skip File Check	No
Template Name	CN_Request_Translation_Log
Repository Name	CN_Translation_Log
Dependent Operation Type	Export
Dependent Operation	CN_Cleanup_Translation_Queue
Dependent Operation Immediate	Yes
Dependent Keys	Parameter1
Dependent Values	||Parameter1||

This import should specify the Import Template named "CN_Request_Translation_Log". It will be created by the Migrate In operation during the Services Framework deployment. The import template must have the following mapped columns:

Column Name	Description	Is Key
CN_Attribute_Language	Extension of the source language (e.g., en, sp, de, etc.)	Yes
CN_Attribute_Name	Name of the attribute to be translated (KEY)	Yes
CN_Attribute_Translated_Value	Translated value (populated by translation company)	
CN_Attribute_Value	Value that needs to be translated	
CN_Internal_Record_ID	InternalRecordId of the repository record (KEY)	Yes
CN_PK1	Repository record primary key 1	
CN_PK2	Repository record primary key 2	
CN_PK3	Repository record primary key 3	
CN_PK4	Repository record primary key 4	
CN_PK5	Repository record primary key 5	
CN_Repository_Name	Name of the repository	

Column Name	Description	Is Key
CN_Request_Date	Date of when the request was made	
CN_Request_Status	Status of the request: Requested	
CN_Translate_Language	Extension of the target language (e.g., en, sp, de, etc.)	Yes

When the referenced update file is imported, it will either create new or updating existing records in the Change Notification Translation Log repository. Once the import has completed, the Scheduled Export named “CN_Cleanup_Translation_Queue” is launched.

Translation Queue Cleanup Scheduled Export

The scheduled export named “CN_Cleanup_Translation_Queue” deletes the records in the Change Notification Translation Queue that are assigned to the batch ID passed to the export. It can be created from the export that is included in the Scheduled Framework deployment folder: **<deploymentFolder>\EPIM\Scheduled Exports**. The cleanup is handled as a separate job to facilitate troubleshooting as well as provide an opportunity to reprocess language requests if there is a failure to transmit the translation request file to the Translation Company or the update. The Scheduled Export should have the following settings:

Attribute Name	Value
Export Name	CN_Cleanup_Translation_Queue
Export Type	View
View SQL	EXEC CN_Cleanup_Translation_Request_Queue ||Parameter1||,0
View Format	COMMA
Export File Name	CN_Cleanup_Translation_Queue_||Parameter1||.csv
Target Type	File
Target Path	<drive>:/Enterworks/shared/Reports

Translation Queue Update Scheduled Import

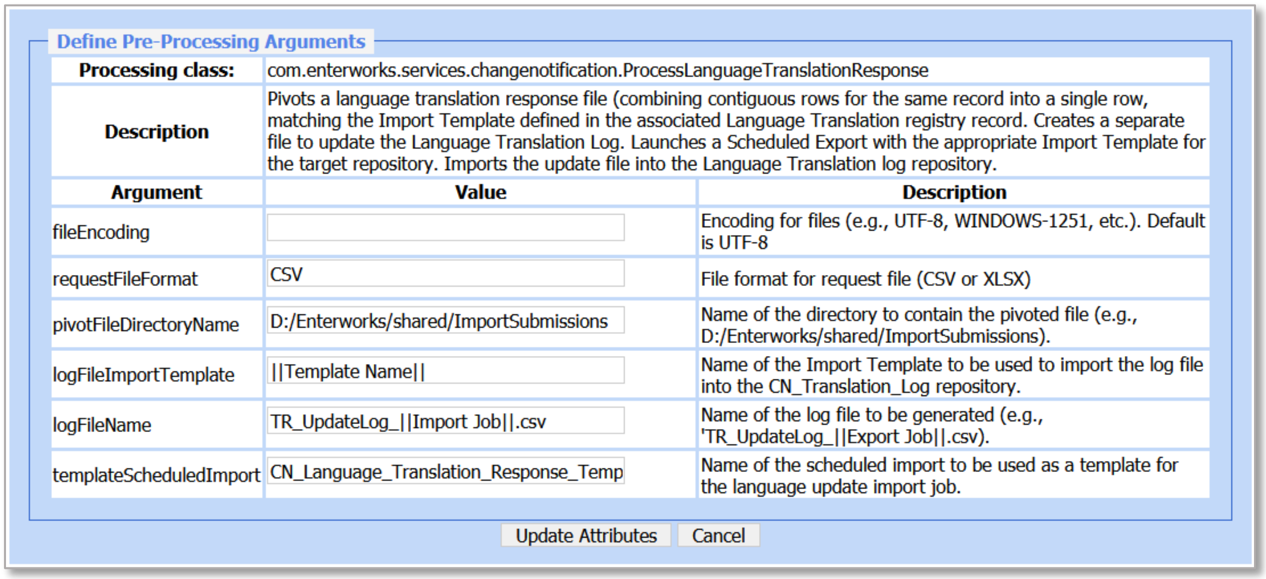
A master Scheduled Import must be defined for all repositories being translated to pre-process all inbound translation update files. The Scheduled Import will retrieve a single file from the source (e.g., from the designated FTP server directory) then invoke the PrepareTranslationUpdate pre-processing block. The pre-processing block will launch a Scheduled Import Job with the proper settings and then proceed to update the CN_Translation_Log entries to signify the response to the request has been received and processed. The Scheduled Import should have the following settings:

Attribute Name	Value
Import Name	CN_Language_Translation_Response
Import Type	Template
Preprocess File	Yes
Preprocess Class	com.enterworks.services.changenotification.ProcessLanguageTranslationResponse
Preprocess Keys Preprocess Values	See Preprocess details below
Import Directory Name	Directory in which the Language Translation Responses will be placed
Import File Name	Must match the common portion of the language translation request files generated by each Translation Request Scheduled Export. For example, if exports are defined names in the format: LTR_<language>_<batchId>.csv, the Import File Name should be: LTR_*.csv.
Import Format	CSV
Move Import File	Yes
Skip File Check	No
Template Name	CN_Translation_Request_Log
Repository Name	CN_Translation_Log
Dependent Operation Type	Import
Dependent Operation	CN_Language_Translation_Response
Dependent Operation Immediate	Yes

The Preprocess Class ProcessLanguageTranslationResponse reads the response file, identifies the CN_Translation_Registry for the file, pulls the Import Template from the entry and generates an import file for that Import template to update the translation target language with the values in the file. A separate Scheduled Import job is launched to process this file. The pre-process class also creates a separate file for updating the CN_Translation_Log. The pre-process class has the following arguments:

Argument	Description
fileEncoding	Specifies the encoding for files. The default is UTF-8
requestFileFormat	Format for the request file: XLSX or CSV
pivotFileDirectoryName	Location of where the Translation Log import files are to be placed for processing. The Scheduled Import must have this same directory name
logFileImportTemplate	||Template Name|| - Name of the import template used to update the CN_Translation_Log repository. This import template must be defined as the Template Name in this Scheduled Import
logFileName	TR_UpdateLog_||Import Job||.csv - Name of the Translation Log import file.
templateScheduledImport	CN_Language_Translation_Response_Template - Name of the template Scheduled Import to be used when creating each language translation import. The contents of this Scheduled Import are copied to the Scheduled Import Jobs and the particulars (e.g., file name, language, repository name, import template name, etc.) are updated in the record before the import job is processed.

The following example shows the configuration of the ProcessLanguageTranslationResponse pre-processing:


Define Pre-Processing Arguments

Processing class:	com.enterworks.services.changenotification.ProcessLanguageTranslationResponse
Description	Pivots a language translation response file (combining contiguous rows for the same record into a single row, matching the Import Template defined in the associated Language Translation registry record. Creates a separate file to update the Language Translation Log. Launches a Scheduled Export with the appropriate Import Template for the target repository. Imports the update file into the Language Translation log repository.

Argument	Value	Description
fileEncoding		Encoding for files (e.g., UTF-8, WINDOWS-1251, etc.). Default is UTF-8
requestFileFormat	CSV	File format for request file (CSV or XLSX)
pivotFileDirectoryName	D:/Enterworks/shared/ImportSubmissions	Name of the directory to contain the pivoted file (e.g., D:/Enterworks/shared/ImportSubmissions).
logFileImportTemplate	||Template Name||	Name of the Import Template to be used to import the log file into the CN_Translation_Log repository.
logFileName	TR_UpdateLog_||Import Job||.csv	Name of the log file to be generated (e.g., 'TR_UpdateLog_||Export Job||.csv).
templateScheduledImport	CN_Language_Translation_Response_Temp	Name of the scheduled import to be used as a template for the language update import job.

The templateScheduledImport must point to a Scheduled Import template that will be the basis of all Language Translation Response jobs.

The preprocessing reads the contents of the file, identifying the associated Translation Registry entry, which identifies the repository name and multi-language attributes. The contents of the file are pivoted so that each row contains all multi-language columns for the same record. This file will be imported into the target repository for the target language identified in the file using the Import Template defined in the Translation Registry entry. A second file is created containing the updates for the Change Notification Translation Log repository.

The first file has the following format:

Column Name	Description
<primaryKeyAttributes>	Up to 5 primary key columns (depending upon the repository), with the column name being the actual primary key attribute names.
<multiLanguageAttributes>	One column for every attribute listed in the referenced Translation Registry record.

This second file contains:

Column Name	Description
CN_Repository_Name	Name of the repository
CN_PK1	First primary key
CN_PK2	Second primary key (empty if repository has only one primary key)
CN_PK3	Third primary key (empty if repository has fewer than three primary keys)
CN_PK4	Fourth primary key (empty if repository has fewer than four primary keys)
CN_PK5	Fifth primary key (empty if repository has fewer than five primary keys)
CN_Attribute_Name	Name of the multi-language attribute
CN_Attribute_Language	Language extension of the source value for the attribute (empty if the default language)
CN_Translate_Language	Language extension of the target language
CN_Request_Status	Set to “Updated”

Column Name	Description
CN_Request_Date	Set to the current date.
CN_Translated_Value	Translation for the value.

Once pre-processing is complete, a Scheduled Import Job record is created and a job launched to process the target repository update file. This scheduled import job is used as the basis for the new job, but all of the appropriate attributes are updated to reflect the generated file name, target repository name, target language, and import template name. If the job is successfully launched, this import job proceeds to import the second file to update the Request Status in the corresponding records in the Change Notification Translation Log repository.

Translation Queue Update Response Template

This Scheduled Import is the basis for all Scheduled Imports launched by the Translation Queue Update Scheduled Import to update the multi-language values from the Language Translation Response. This Scheduled Import should have some attributes pre-defined but others will be auto-populated by the ProcessLanguageTranslationResponse pre-process based on the content of the response file and the referenced Language Translation Registry record (the bolded values are ones overwritten by the Pre-processing):

Attribute Name	Value
Import Name	CN_Language_Translation_Response_Template – the launched job will have the name: "**TRR_<repositoryName>_<translationLanguage>**"
Import Type	Template
Preprocess File	**<set to No>**
Preprocess Class	**<cleared>**
Preprocess Keys	**<cleared>**
Preprocess Values	**<cleared>**
Import Directory Name	**<populated from value of pivotFileDirectoryName argument in ProcessLanguageTranslationResponse pre-processer>**
Import File Name	**<populated from the value of the response file with _PIVOT suffix added before the extension>**
Import Format	CSV or XLSX
Move Import File	Yes
Skip File Check	No
Source Type	**<set to File>**
Template Name	**<set to the template defined in the CN_Translation_Registry record>**
Saved Set	**<set to the same as Import File Name>**
Repository Name	**<set to the repository name defined in the CN_Translation_Registry record>**

Translation Language Update Import Templates

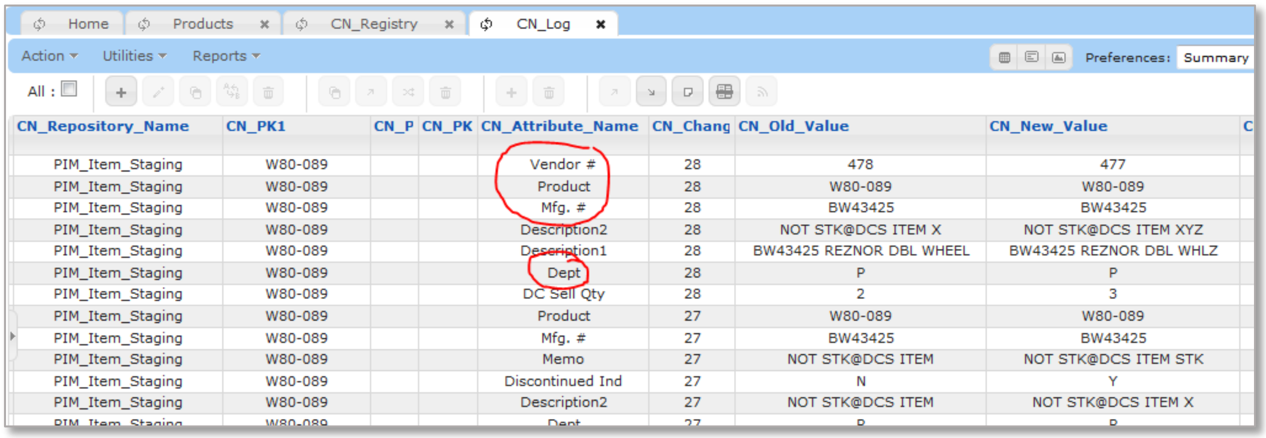
An Import Template must be defined to update the multi-language attributes in each repository. Only one import template must be defined regardless of the number of target languages. Each scheduled import for the different target languages will reference the same Import Template for the same repository. Each import template must contain only the primary key attributes and the multi-language attributes. Multiple import templates must be defined for the same target repository if there is more than one Translation Registry entry for that repository, such as in cases where large numbers of multi-language attributes are broken up into multiple parts.

For example, if the repository Product_Staging has a primary key of Product ID and the multi-language attributes: Product Name and Product Description, the import template would have the following:

Column Name	Description	Is Key
Product ID	Primary key for Product_Staging	Yes
Product Name	Name of Product	No
Product Description	Description of Product	No

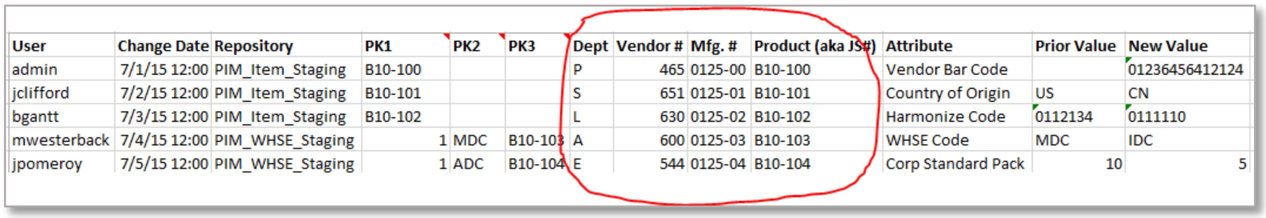
SQL-Based Reports

When the Change Notification functionality is used to log changes to the CN_Log repository, the data can be exported based on search criteria (e.g., repository, date range, etc.) without needing to use SQL. However, the resulting report will show referential attributes (i.e., attributes and their values that provide context such as key business fields, but aren't necessarily part of the change) as separate rows in the export since they are stored that way in the CN_Log repository:



CN_Repository_Name	CN_PK1	CN_P	CN_PK	CN_Attribute_Name	CN_Chang	CN_Old_Value	CN_New_Value	C
PIM_Item_Staging	W80-089			Vendor #	28	478	477	
PIM_Item_Staging	W80-089			Product	28	W80-089	W80-089	
PIM_Item_Staging	W80-089			Mfg. #	28	BW43425	BW43425	
PIM_Item_Staging	W80-089			Description2	28	NOT STK@DCS ITEM X	NOT STK@DCS ITEM XYZ	
PIM_Item_Staging	W80-089			Description1	28	BW43425 REZNOR DBL WHEEL	BW43425 REZNOR DBL WHLZ	
PIM_Item_Staging	W80-089			Dept	28	P	P	
PIM_Item_Staging	W80-089			DC Sell Qty	28	2	3	
PIM_Item_Staging	W80-089			Product	27	W80-089	W80-089	
PIM_Item_Staging	W80-089			Mfg. #	27	BW43425	BW43425	
PIM_Item_Staging	W80-089			Memo	27	NOT STK@DCS ITEM	NOT STK@DCS ITEM STK	
PIM_Item_Staging	W80-089			Discontinued Ind	27	N	Y	
PIM_Item_Staging	W80-089			Description2	27	NOT STK@DCS ITEM	NOT STK@DCS ITEM X	
PIM_Item_Staging	W80-089			Dept	27	P	P	

A more useful format for the report is one that includes the referential fields for each change row:



User	Change Date	Repository	PK1	PK2	PK3	Dept	Vendor #	Mfg. #	Product (aka JS#)	Attribute	Prior Value	New Value
admin	7/1/15 12:00	PIM_Item_Staging	B10-100			P	465	0125-00	B10-100	Vendor Bar Code		01236456412124
jclifford	7/2/15 12:00	PIM_Item_Staging	B10-101			S	651	0125-01	B10-101	Country of Origin	US	CN
bgantt	7/3/15 12:00	PIM_Item_Staging	B10-102			L	630	0125-02	B10-102	Harmonize Code	0112134	0111110
mwesterback	7/4/15 12:00	PIM_WHSE_Staging	1	MDC	B10-103	A	600	0125-03	B10-103	WHSE Code	MDC	IDC
jpomeroy	7/5/15 12:00	PIM_WHSE_Staging	1	ADC	B10-104	E	544	0125-04	B10-104	Corp Standard Pack	10	5

To produce the above report requires creating a View (SQL) scheduled export with a PIVOT like the following:

```
select l.CN_Changed_By as [User], CN_Changed_Datetime as [Change Date],
CN_Repository_Name as Repository,
CN_PK1 as PK1, isnull(CN_PK2, '') as PK2, isnull(CN_PK3, '') as PK3,
isnull(c.Dept, '') as Dept, isnull(c.[Vendor #], '') as [Vendor #],
isnull(c.[Mfg. #], '') as [Mfg. #], isnull(c.Product, '') as [Product (aka JS#)],
CN_Attribute_Name as Attribute, isnull(CN_Old_Value, '') as [Prior Value],
isnull(CN_New_Value, '') as [New Value]
 from CN_Log l
left outer join (Select CN_Change_ID, Dept, [Vendor #], [Mfg. #], Product from
(SELECT CN_Change_ID, CN_Attribute_Name, CN_New_Value
    FROM (select CN_Change_ID, CN_Attribute_Name, CN_New_Value
              from CN_Log where CN_Attribute_Name in ('Dept','Vendor #','Mfg.
#','Product')) e
) AS X
PIVOT
(
MAX (CN_New_Value)
FOR CN_Attribute_Name IN (Dept, [Vendor #], [Mfg. #], Product)
) AS PivotTable) c
on c.CN_Change_ID = l.CN_Change_ID
 where not (l.CN_Attribute_Name in ('Dept','Vendor #','Mfg. #', 'Product') AND
l.CN_Old_Value = l.CN_New_Value)
 order by l.CN_Change_ID, CN_Attribute_Name
```

To produce a delta report, showing only the changes since the previous report, the SQL must include the condition comparing the Created column to the [DELTA_DATETIME] field:

```
where not (l.CN_Attribute_Name in ('Dept','Vendor #','Mfg. #', 'Product') AND
l.CN_Old_Value = l.CN_New_Value)
 and l.Created > '[DELTA_DATETIME]'
```

There may also be use cases (such as the one described in the first section) where a report needs to extract the first old value and the last new value for each attribute from the CN_Log entries that have been created since the last time the report ran. Generating such a report increases the complexity of the required SQL query.

Reports Using the CN_Reports Repository and Scheduled Export

The SQL-based reports can be very complex, which limits who can create and maintain such reports to those with advanced knowledge/experience with SQL. The CN_Reports repository in conjunction with the ProcessLaunchChangeNotificationReport Scheduled Export Post-Processor, provides the means for users with no SQL knowledge to create and maintain reports from the CN_Log repository. The CN_Reports identifies what is to be reported and to whom. The Scheduled Export determines when and how.

The key features of the CN_Reports processing are:

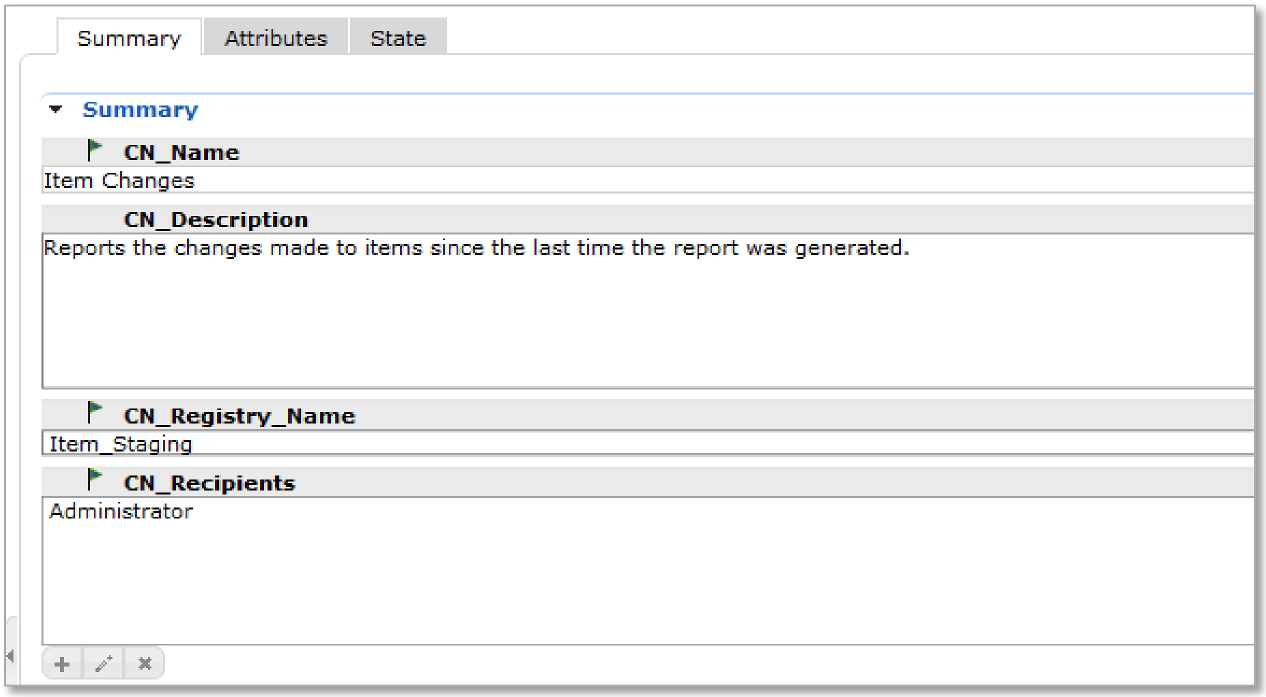
- Reports are linked to a specific CN_Registry record, meaning only the CN_Log entries that were saved from the CN_Registry record will be included in the report.
- Reports are defined in terms of list of selected attributes for which the old and new value are to be displayed, the new logged value (e.g., descriptive attributes), and the values from the current record are displayed.
- Recipients for the reports are defined in terms of Enable groups.
- Reports can be transmitted in any fashion supported by the Scheduled Exports processing.
- Repositories with ownership (where individual users are assigned to records so that they "own" them and are the only ones who can change specific attributes). Separate files are generated for each owner recipient. If the Report Scheduled Export is configured to send the export via Email, each owner will receive only their version of the report.
- Reports can be configured to be full or delta

CN_Reports Repository

The CN_Reports repository contains the reports definitions. Each report links to a CN_Registry record. Each Report Scheduled Export links to a CN_Report record.

Summary Tab

The Summary tab identifies the name of the report, what CN_Registry record it is associated to, and the intended recipients of the report.



Summary

CN_Name
Item Changes

CN_Description
Reports the changes made to items since the last time the report was generated.

CN_Registry_Name
Item_Staging

CN_Recipients
Administrator

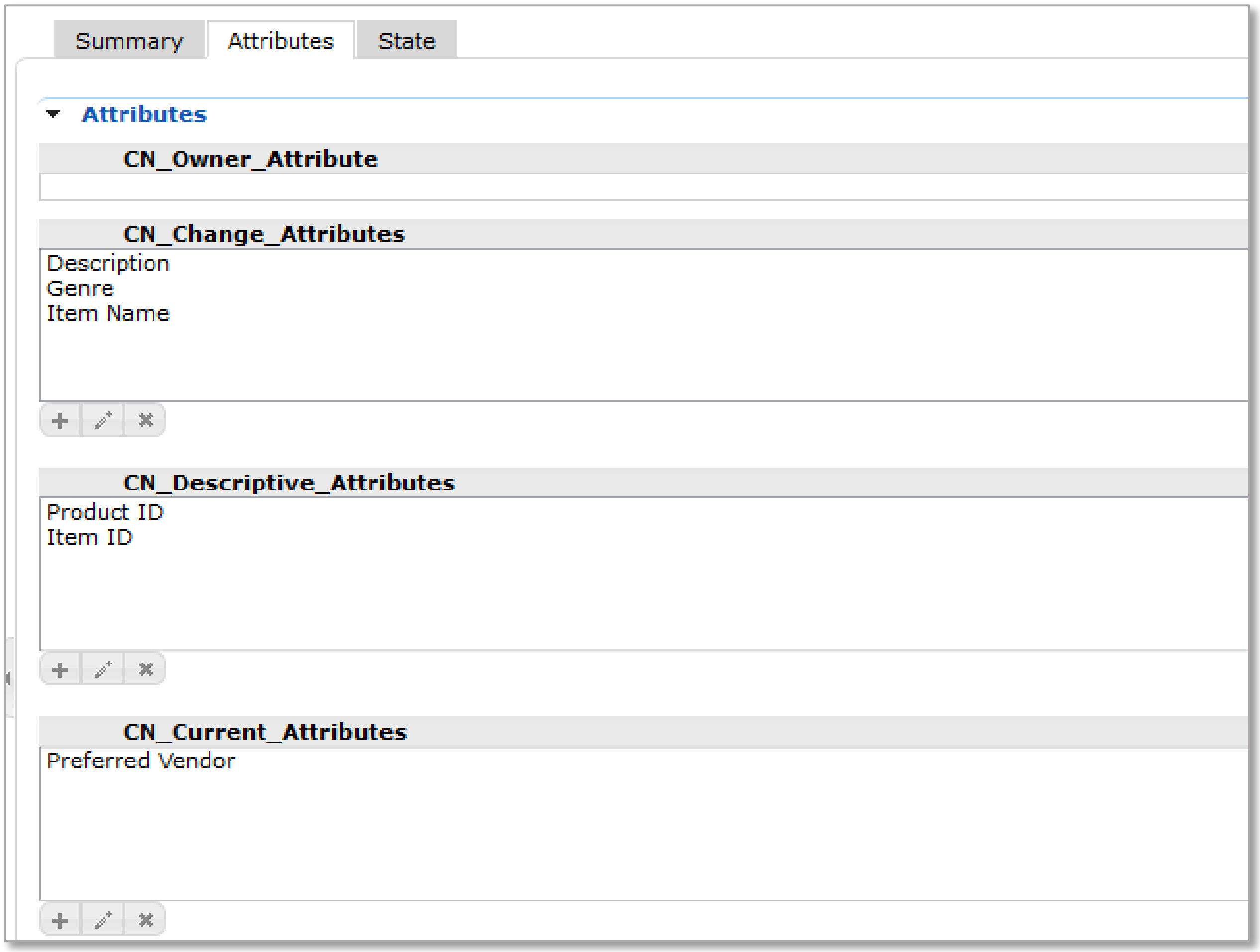
CN_Name – name of the report. Each report must be uniquely named.

CN_Description – optional description for the report

CN_Registry_Name – identifies which CN_Log records are to be included in the report by way of which CN_Registry entry was responsible for the generation of those records.

CN_Recipients – identifies the Enable groups that will receive the report if the Scheduled Export that launches the report has its Target Type set to Email.

Attributes Tab



CN_Owner_Attribute

CN_Change_Attributes
Description
Genre
Item Name

CN_Descriptive_Attributes
Product ID
Item ID

CN_Current_Attributes
Preferred Vendor

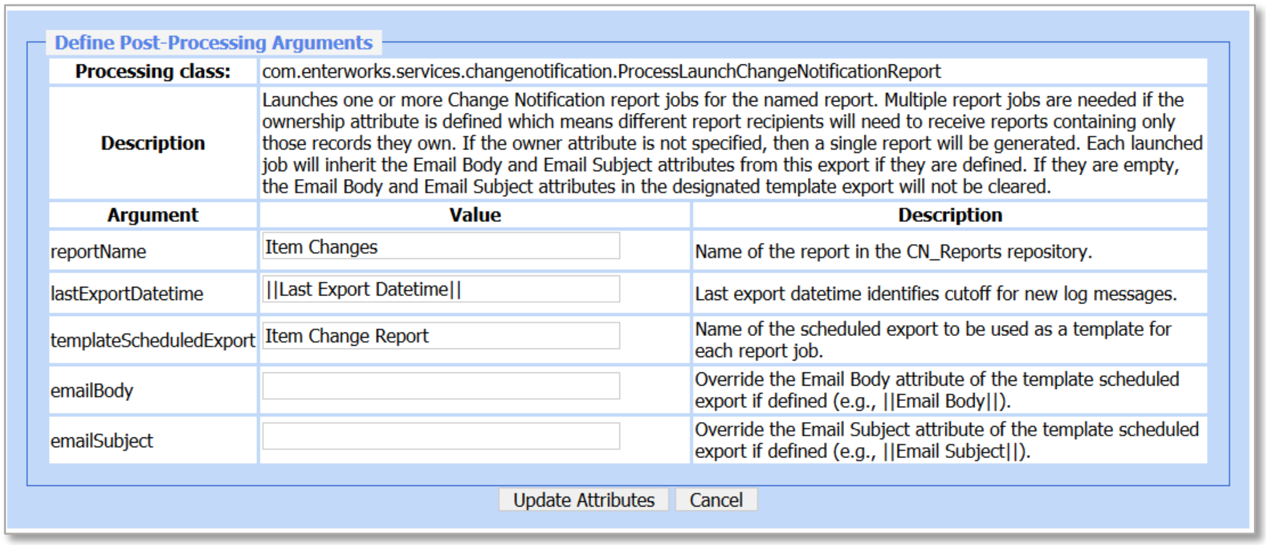
CN_Owner_Attribute – Name of the attribute that is associated with a code set that contains ownership by group. Each value in this attribute determines how many different report files are generated and who sees them. If set, then a separate report will be generated for each owner and contain only those records assigned to that owner. Only attributes with ownership Code Set, Hierarchy, or Taxonomy can be selected. The processing only supports group, not user ownership. The owner must match a group in the CN_Recipients attribute. The selected attribute must be in the snapshot table (i.e., set to Relational).
CN_Change_Attributes – Comma-delimited list of attributes to show the old and new values for. If **-All-** is selected, then all change attributes will be included in the report. A separate row is shown for each attribute with it's old and new values as separate columns.
CN_Descriptive_Attributes – Comma-delimited list of the attributes that were logged at the same time as the attributes of interest were changed but to be shown as separate columns in the report. Attributes included in this list will be ignored in the CN_Change_Attributes list. Only the new value is displayed. The attributes included in this list will most often be the attributes flagged as Log Always in the associated CN_Registry record.
CN_Current_Attributes – Comma-delimited list of attributes to be retrieved from the current records to be included in the report as separate columns. If it's necessary to report the values at the time the change was made, the attributes must be assigned to CN_Descriptive_Attributes. Any attributes listed here will be ignored in the CN_Descriptive_Attributes and CN_Change_Attributes lists.

Report Scheduled Export

Scheduled Exports are used to launch and transmit the CN_Reports. Two Scheduled Exports must be defined. The first is a Custom export that invokes a Post-Processing module to launch one or more Scheduled Export Jobs based off the information in the designated CN_Report record along with the configuration settings of the Post-Processing module. The second Scheduled Export is the template for the actual jobs that are launched by the first Scheduled Export. The number of jobs launched is dependent upon the settings in the CN_Reports record.

Launch Report Scheduled Export

The Launch Report Scheduled Export is a Scheduled Export of type Custom that does not export any CN_Log data (it does produce an export containing details about the launch processing). Instead the Scheduled Export invokes the ProcessLaunchChangeNotificationReport post-processing module that launches the Scheduled Export jobs to actually generate one or more reports. This module has the following configurable settings:



Define Post-Processing Arguments

Processing class:	com.enterworks.services.changenotification.ProcessLaunchChangeNotificationReport
Description	Launches one or more Change Notification report jobs for the named report. Multiple report jobs are needed if the ownership attribute is defined which means different report recipients will need to receive reports containing only those records they own. If the owner attribute is not specified, then a single report will be generated. Each launched job will inherit the Email Body and Email Subject attributes from this export if they are defined. If they are empty, the Email Body and Email Subject attributes in the designated template export will not be cleared.

Argument	Value	Description
reportName	Item Changes	Name of the report in the CN_Reports repository.
lastExportDatetime	||Last Export Datetime||	Last export datetime identifies cutoff for new log messages.
templateScheduledExport	Item Change Report	Name of the scheduled export to be used as a template for each report job.
emailBody		Override the Email Body attribute of the template scheduled export if defined (e.g., ||Email Body||).
emailSubject		Override the Email Subject attribute of the template scheduled export if defined (e.g., ||Email Subject||).

reportName – Identifies the CN_Reports record to process. Must match the CN_Name in a CN_Reports record

lastExportDatetime – specifies the value to be placed in the Last Export Datetime attribute of each launched Scheduled Export job. Must be blank if the report is not a delta. If the report is a delta, this should be set to ||Last Export Datetime|| to pass in the date and time of the last time the launch Scheduled Export ran (the Scheduled Export must be configured as a Delta export in order for the delta export processing to be enabled.

templateScheduledExport – name of the template Scheduled Export that will be copied for each launched Scheduled Export. This Scheduled Export must be of type View and be configured to have the desired target for the generated report. Some fields will be updated for each launched job

emailBody – optional override for the Email Body attribute in each launched Scheduled Export. To pass in the Email Body from this Scheduled Export, the parameter must be set to ||Email Body||.

emailSubject – optional override for the Email Subject attribute in each launched Scheduled Export. To pass in the Email Subject from this Scheduled Export, the parameter must be set to ||Email Subject||.

This Scheduled Export's output is a report file, detailing the processing of launching the actual Report Scheduled Exports that generate the reports.

Report Scheduled Export

The Report Scheduled Export generates the actual report. It must be defined as type View (SQL-based). When launched, the following attributes are updated by the Launch Report Scheduled Export:

Export Name – name of the export – this is set to the name of the Report Scheduled Export template plus the report group name. If no ownership is identified, then the name "CN_All_Recipients" will be used. For example, the Launch Report Scheduled Export with a reportName of "Item Change Report" with no ownership specified, the launched Report Scheduled Export will be "Item Change Report_CN_All_Recipients"

Parameter1 – set to the name of the report. This can be referenced by any attribute in the Scheduled Export by specifying ||Parameter1|| in any attribute with a text field.

Parameter2 – set to the name of the report group

View SQL – set to the dynamically-generated SQL to create the report. The SQL that is generated is based on the settings in the CN_Reports record along with the designated Group/User information (for Owner-based reports)

Last Export Datetime – set with the value specified in the post processing argument lastExportDatetime

Target Email – set to the email address list obtained form the report group users

Email Body – set with the value specified in the post processing argument emailBody

Email Subject – set with the value specified in the post processing argument emailSubject

Change Report Processing

Whenever a change report scheduled export is launched, the following steps will be performed:

1. Retrieve the details for each named report.
2. For each report, determine whether there is an owner attribute specified. If there is, perform the following sub-steps:
    a. Retrieve the list of unique owner groups from the code set associated with the owner attribute. This determines how many separate report jobs are going to be run.
    b. For each owner group, cross-references the users assigned to the group with the users assigned to each recipient group defined for the report and generate a list of e-mail recipients.
    c. One report will be generated for each unique owner group
3. If there isn't an owner attribute specified, perform the following sub-steps:
    a. Retrieve the list of e-mail addresses solely from the users assigned to the designated recipient groups.

   b. Only one report will be generated

4. At this point, the process knows how many reports need to be generated and who the e-mail recipients would be.
5. Retrieve the details from the scheduled export to be used as the basis for each report job to be launched.
6. For each report identified in the previous steps, update the base export properties with the specifics, such as the identified e-mail targets and the specific report (including name and owner if specified) and create a new Scheduled Export Jobs record to launch each report.

Each new Scheduled Export Job's post-processing generates the report according to the settings in the report definition and the specified owner group (if defined). The Scheduled Export process will then send the file as defined in the original job (Directory, Email, or FTP).