PROGRESS Corticon

# Corticon Migration Guide

PROGRESS

# Notices

# Table of Contents

# 1

# Overview

This Corticon migration guide describes the qualifications and tasks to successfully advance an existing installation and deployment of Corticon to this release, 5.4.1.

**Note:** If you are applying Service Pack 5.4.1 to an existing 5.4.0 installation, you only need to download and run the Service Pack installer. See the *Corticon Studio: Installation Guide* and the Corticon Server guides for details on applying Service Pack updates.

See *What's New in Corticon* for a complete list of new and changed features.

For details, see the following topics:


- Migrating from earlier Corticon 5.x releases
- Migrating from Corticon releases earlier than 4.3
- Migrating from Corticon 4.3


# Migrating from earlier Corticon 5.x releases

If you are advancing from 5.3, no changes are required. If advancing from an older version, you need to perform the migration tasks described in the document.

If you have Corticon assets from an earlier V5 release, simply open the assets in a V5.4 Studio to initiate the updates, or use the **Upgrade** utility to upgrade whole projects and hierarchies of projects. See the *Corticon Studio: Installation Guide* for details on this utility.

Ruleflows deployed to a Corticon Server are dynamically upgraded when the upgraded Corticon Server installation opens them.

**Note:** V5.4 servers can run older assets, after migration, without issue, but it is not recommended that you attempt to use newer assets on older V5 servers.

# Migrating from Corticon releases earlier than 4.3

Corticon 3 and Corticon 4 asset formats changed substantially from release to release; therefore, if you are migrating from a Corticon release earlier than Corticon 4.3, we strongly recommend that you migrate in steps:

1. Obtain Corticon Studio 4.3 and use it to open your assets. Corticon Studio 4.3 will automatically upgrade your assets into the latest 4.X format
2. Migrate your 4.3 assets to Corticon 5 format using the Version 4 Assets Migration Wizard (part of Corticon Studio Version 5), as described in this document.

# Migrating from Corticon 4.3

Corticon Studio 5 includes a Migration Wizard to allow you to import your Corticon Version 4 assets into Corticon 5. Technically, the migration process involves converting your V4 assets into the new V5 asset format.

V5 assets have a new internal structure that is different from V4. V4 assets are Java Archives (JARs), while V5 assets are XML documents, and V4 and V5 use different file extensions.

| Corticon 4 Assets | Corticon 5 Assets |
|---|---|
| Vocabulary `(.cvj)` | Vocabulary `(.ecore)` |
| Ruleset `(.ccj)` | Ruleflow `(.erf)` of Rulesheets `(.ers)` |
| Test `(.ctj)` | Ruletest `(.ert)` |

Perhaps the most significant change is that a single V4 Ruleset (.ccj) asset is represented as multiple files in V5. The Migration Wizard will convert a single V4 Ruleset into a V5 Ruleflow (.erf) file plus one or more V5 Rulesheet (.ers) files.

In V4, multiple Rulesheets were stored inside a single Ruleset (.ccj) file. In V5, each Rulesheet is stored separately in its own (.ers) file. A V5 Ruleflow can be seen as a “container” of Rulesheets, and internally, a Ruleflow (.erf) contains internal “pointers” to all of its Rulesheet (.ers) files.

A V4 Test (.ctj) asset is referred to as a Ruletest (.ert) in V5. Similarly to V4, a V5 Ruletest asset may contain any number of Testsheet tabs.

# 2

# Tasks for advancing from Corticon 4.3 to this release

The migration tasks for advancing Corticon 4.3 assets and deployments require that you install Corticon 5.4, and then perform some tasks.

**Note: Limited RDBMS brands supported** - The Enterprise Data Connector in Corticon 5.4 is certified on specific database brands. See the Platforms page at the Progress Corticon web site or contact Technical Support.

For details, see the following topics:

- Properties that influence migration
- Using the Migration Wizard
- Import processing
- Resolving migration errors
- Post-migration asset review
- Migrating enumerated value sets to Custom Data Types
- Converting extensions

# Properties that influence migration

Corticon V5 has two properties that influence migration. You should evaluate these properties and set them appropriately for your application prior to migration.

**Table 1:**

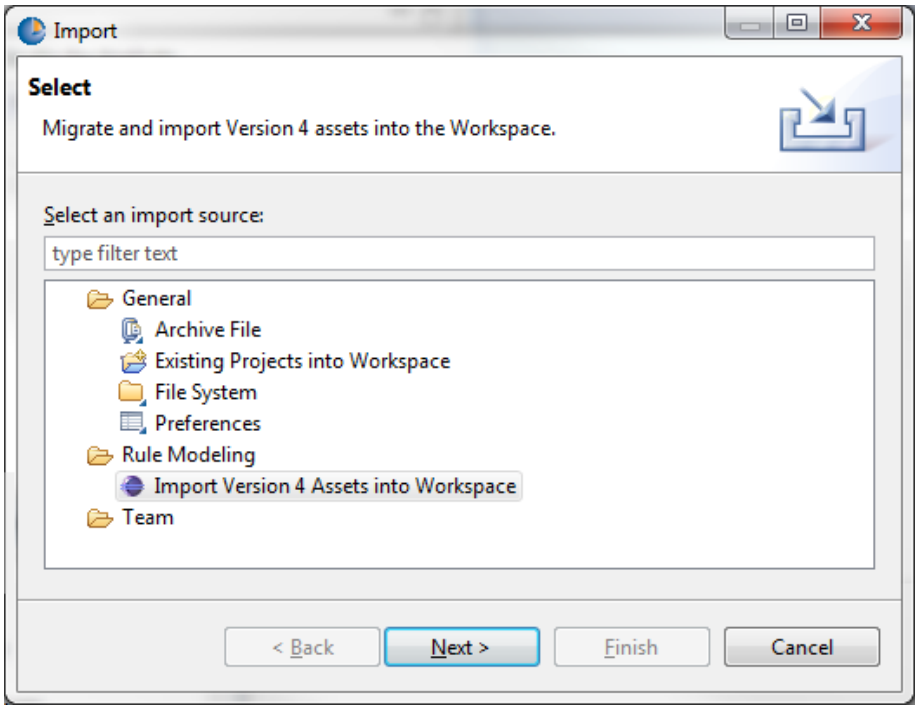
| Property | Location |
|---|---|
| `com.corticon.migration.vocabulary.mandatory.flag` | `Common properties` |
| `com.corticon.studio.migration.datesubtype.default` | `Studio properties` |

In V5, the Vocabulary Mandatory flag has an impact on engine behavior at runtime, whereas in V4 this was not the case. As a result, migration of V4 assets to V5 may result in rule execution behavior that is inconsistent with V4. Property `com.corticon.migration.vocabulary.mandatory.flag` can be used to control this behavior. See the comments on this property in *Common properties* in the *Integration and Deployment Guide*for details.

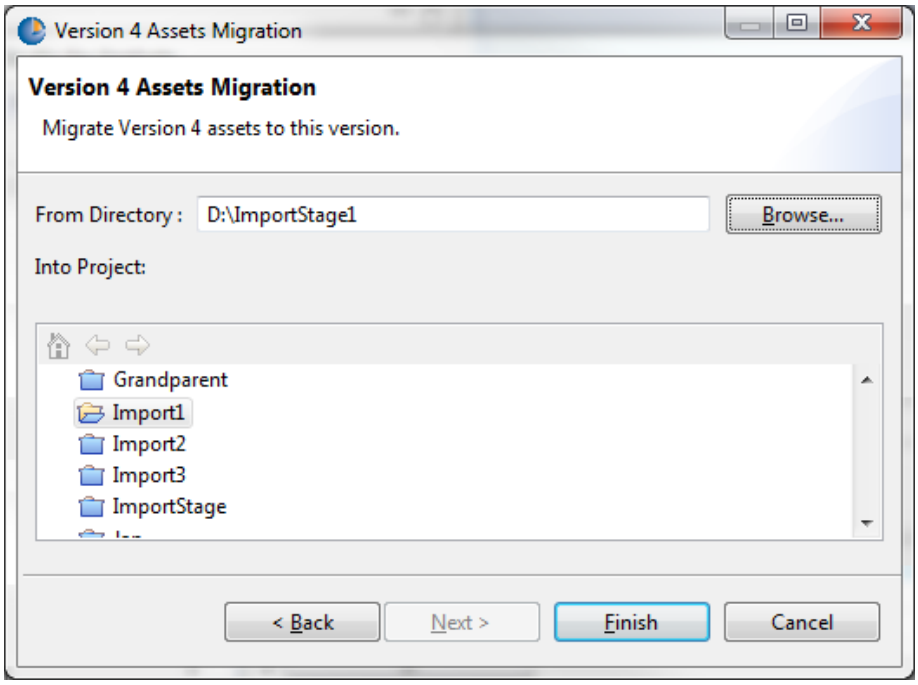
In V5, types Date, Time and DateTime are “first-class citizens” and there is no concept of Subtype. Therefore, for V4 assets that use type Date but do not explicitly specify an optional Subtype, the attribute type `Date` is ambiguous. Property `com.corticon.studio.migration.datesubtype.default` allows you to specify how these *wildcard* dates are handled (See **Date, Time and Date/Time** in this document.) See the comments on this property in *Studio properties* in the *Integration and Deployment Guide*for details.

# Using the Migration Wizard

In Corticon Studio, select **File > Import**, and then choose **Rule Modeling > Import Version 4 Assets into Workspace** to open the **Import** diailog:




Click **Next**. The Version 4 Assets Migration Wizard opens, requesting you to select a directory folder that contains Corticon 4 Assets, and to select a Project that will contain the converted Corticon 5 assets:




The **From Directory** specifies a folder which can contain any number of Corticon 4 assets. Note that if the **From Directory** folder contains subfolders, the contents of all of those subfolders will be processed recursively. This allows you to import a large number of assets into Corticon 5 in a single step.

When you click **Finish**, import processing starts.

# Import processing

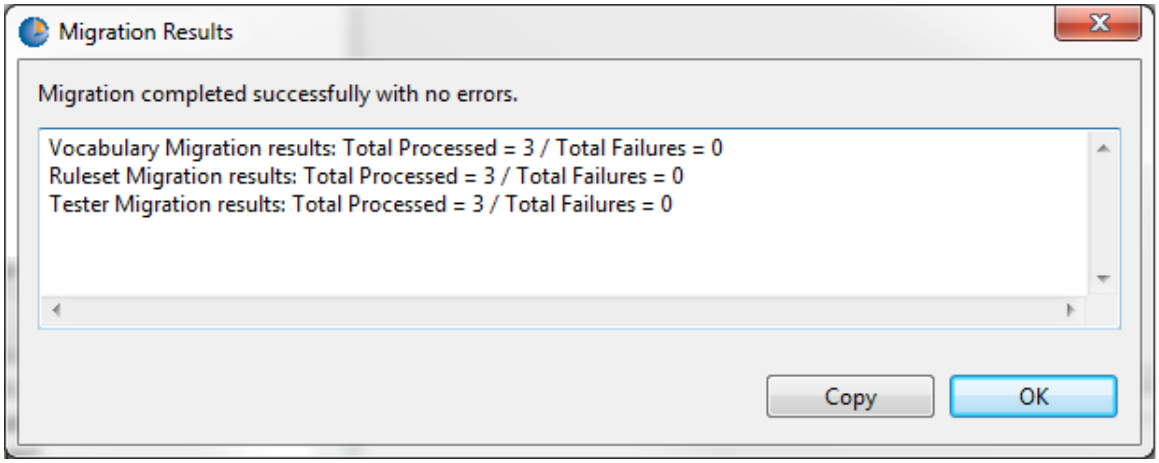
The import process executes a series of passes to process all of the Corticon 4 assets located in the **From Directory** and its subdirectories. This multi-pass process converts assets in the following sequence:

1. Vocabulary (.cvj) files are converted into Corticon 5 .ecore files.
2. Ruleset (.ccj) files are converted into Ruleflow (.erf) and Rulesheet (.ers) files.
3. Test (.ctj) files are converted into Ruletest (.ert) files.

This order of processing is necessary because of asset dependencies:

- A Ruleset cannot be converted until its Vocabulary has been successfully converted.
- A Test cannot be converted until its Ruleset has been successfully converted.

The Migration Wizard keeps track of warnings and errors that are detected during processing, and will present results to you at the end of the process, as in this example:




If any serious errors occur during the migration of an asset, none of its dependent assets are migrated. This is a particularly important with respect to the Vocabulary, because until the Vocabulary is migrated none of its related assets can be migrated.

# Resolving migration errors

Most migration errors can be traced down to problems in the input assets. Frequently, such problems can be rectified by making simple changes to the input assets in Corticon Studio 4, then re-running the Import Wizard.

For example, one classic problem is attempting to import assets which are invalid due to syntax errors in expressions or similar causes. We strongly recommend that you ensure that the V4 assets are error free before trying to import them into V5.

Another important consideration is that all related assets be imported in complete sets. For example, when importing Ruleset assets, the Import Wizard must be able to locate the Vocabulary (.cvj) associated with the Ruleset, or the migration will fail. Each Ruleset (.ccj) “knows” the relative path to its respective Vocabulary (.cvj), and if the files are not in the correct relative relationships with one another during import, the process will not be successful.

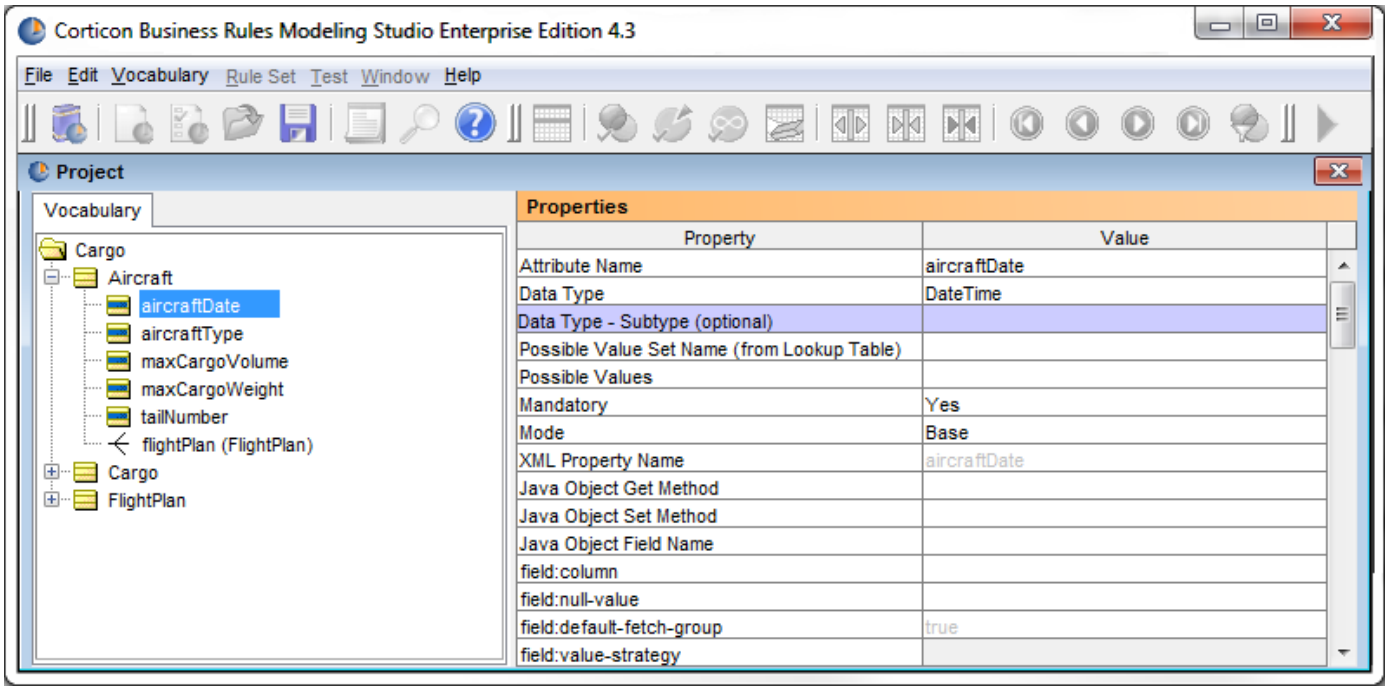
To eliminate these issues, it is a good idea to copy your V4 assets into an import “stage” folder, and then open them one last time in Corticon V4 to ensure that they are error free and in the correct relative positions.

When importing a large number of assets, it can be helpful to break down the process into smaller units of work. For example, you might consider converting just the Vocabulary (.cvj) assets first to make sure there are no problems, and convert Ruleset and Ruletest assets that share the Vocabulary. With smaller units of work, you can more rapidly resolve problems without having to wait for a long-running process to complete.

# Post-migration asset review

## Date, Time, and Date/Time

In Corticon V4, for DateTime attributes, the Vocabulary Editor allows you to specify Data Type Subtype, as shown:




This optional Subtype field allows you to explicitly state that the date is:

- **Full DateTime** - the attribute contains both date and time values
- **Date Only** - the attribute contains only a date
- **Time Only** - the attribute contains only a time

If you do not explicitly specify the Subtype, the system lets you store a full date time, a date or a time in the attribute (referred to as a *wildcard date*).

Corticon V5 doesn't support Subtypes; instead, DateTime, Date and Time are bona fide Data Types. This means you must declare whether the attribute will contain a full date/time, date-only or time-only, and there is no support for wild-card dates.

Because of this, the Import Wizard must make an arbitrary decision when wild-card dates are encountered in the input V4 Vocabulary, so it chooses the value of `CcStudio.properties com.corticon.studio.migration.datesubtype.default` (see Properties that Influence Migration in this document).

After migration, we recommend that you review all Date, Time and DateTime attributes to ensure that they are correct and appropriate for your application.

# Rulesheet Minimum Filters

In V5 minimum filters are deprecated, replaced by a more powerful approach called *limited filters*. With limited filters, you can directly control which aliases will be affected by your filters by disabling those filters in the Scope tree.

The Import Wizard will automatically convert V4 minimum filters to a functionally-equivalent arrangement of limited filters. The resultant V5 Rulesheets should show no behavior changes since V4.

# Applications with Database Connectivity (EDC)

Enterprise Data Connector (EDC) was thoroughly rewritten for V5, and released in 5.3.1.

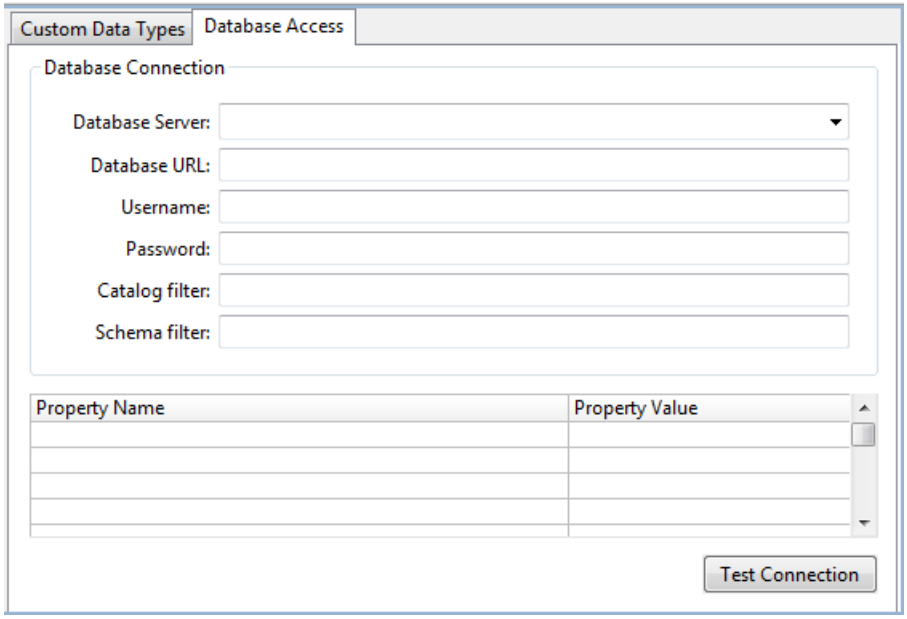
While V4 used JPOX for database access, V5 is based on Hibernate, the most widely-used Java object/relational solution.

V5 comes bundled with Progress Software JDBC drivers, which can reduce configuration headaches, because you can create a database-enabled Vocabulary without downloading third-party JDBC driver JARs.

There are significant technical differences between V4 and V5 EDC, some of which require manual intervention to convert V4 EDC connection properties and database metadata; therefore, after you import your V4 Vocabularies you have to re-enter database connection information into the V5 Database Access tab and then re-import metadata from your database.

**Configuring Database Access Tab**

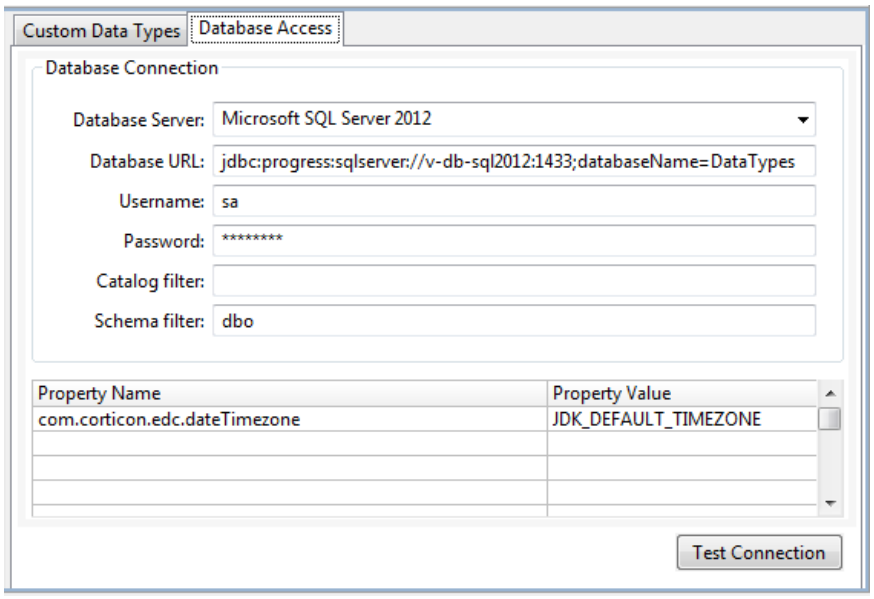
After migrating a V4 Vocabulary to V5, the **Database Access** tab is empty:




Perform the following steps:

1. In either Studio, select the menu command **Window > Preferences**, expand the **Progress Corticon** group, click on **Rule Modeling**, and then choose the User Role option **Integration and Deployment**.
2. Close any open Vocabularies and then open them to expose the **Database Access** tab.
3. Use the Database Server drop-down control to select the preferred database (for example, **Microsoft SQL Server 2012**).
4. Based on the Database Server you choose, the system will automatically display a Database URL "template" that you can use. Replace the `<server>` and `<database_name>` tags with the appropriate values for your database and its location.
5. Specify a valid RDBMS username and password. (These alphanumeric strings must not be all numerics.)
6. Optionally specify Catalog filter and/or Schema Filter to improve performance by limiting the database metadata that is imported into your Vocabulary.
7. Optionally specify database connection properties in the Properties Table.

The results will be similar to the following:




Database connection properties can be used to control Hibernate low-level behaviors, including transaction control, isolation levels, diagnostic logging and so on. Note that Hibernate property names are different from V4 JPOX property names. You can find a complete list of supported properties on the Hibernate web site (http://www.hibernate.org).

### Setting Database Time Zone

One Corticon-specific connection property might be particularly useful to you if your application stores date/time values in the database:

`com.corticon.edc.dateTimezone`

This property pertains only to DateTime data type, and allows you to declare how DateTime values are expressed in the database:

| Value | Purpose |
|---|---|
| `JDK_DEFAULT_TIMEZONE` | Declares that date/time values will be expressed in the Java Virtual Machine (JVM) time zone. Use this setting if your date/time values are expressed in “local” time. |
| `UTC` | Declares that date/time values will be expressed in GMT. This setting is typical for internet applications that are used across time zones. |

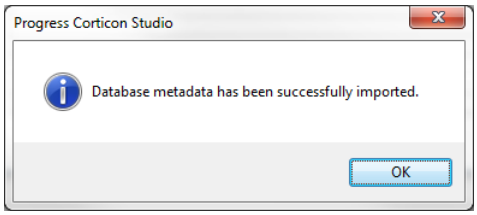
In addition, `com.corticon.edc.dateTimezone` can be set to any valid Java time zone specification. For example:

| Java Timezone | Purpose |
|---|---|
| `America/Los_Angeles` | Declares that date/time values will be expressed in America/Los Angeles time. |

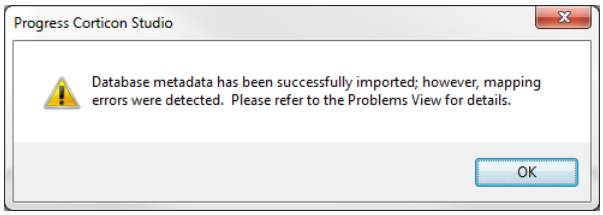
| Java Timezone | Purpose |
|---|---|
| `Europe/Paris` | Declares that date/time values will be expressed in Europe/Paris time. |
| `GMT+01:00` | Declares that date/time values will be expressed in time zone GMT plus one hour. |

### Re-Importing Database Metadata

After setting up the Database Connection tab, you can test the database connection by clicking the **Test Connection** button. Once you have a successful connection, you can re-import database metadata. Select the menu command **Vocabulary > Database Access > Import Database Metadata**. If the imported database metadata matches your Vocabulary exactly, the system will display this message:




However, if mapping issues are detected, the system will display:




Click on the **Problems** View tab to review and resolve the listed mapping warnings.

### Explicitly Setting Entity Identity

Corticon V5 EDC allows you to declare the identity of a datastore-persistent entity in two ways:

- *Application identity* where one or more of the entity's attributes belong to the primary key.
- *Datastore identity*, where a single database column that has no corresponding Vocabulary attribute serves as the primary key.

Corticon V4 has the ability to automatically infer Entity Identity from primary key metadata. Such inferred values appear in light-gray font.

Corticon V5 doesn't automatically infer Entity Identity, primarily for performance reasons. Therefore, for datastore-persistent entities that use application identity, you need to be sure that the Entity Identity field is explicitly entered. Your explicitly-specified value will appear in black font.

Note that if you migrate a V4 Vocabulary that utilized inferred Entity Identity, V5 will issue warning messages in the Problems View after you import database metadata. You'll need to rectify any such warnings by manually entering Entity Identity.

### Reviewing Tables, Columns and Join Expressions

The Migration Wizard is very effective at migrating V4 Vocabularies that use inferred values for tables, columns and join expressions; however, warnings may be issued if inferred values have been overridden with explicitly-specified values.

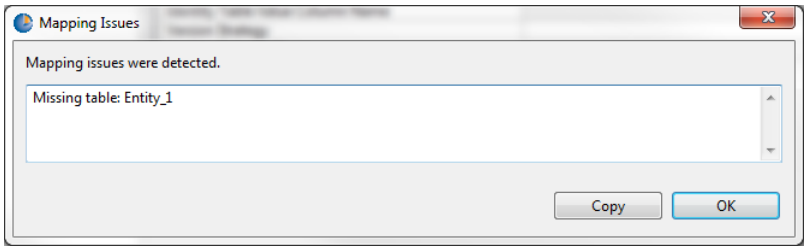
Regardless of the cause, the solution is simple: if an overridden value is flagged with a warning message, just select a valid value from the drop-down list. Note that values in the drop-down list are guaranteed to be valid as they come from freshly-imported database metadata.

Iteratively review entities, attributes and associations to ensure that any manually-entered tables, columns or join expressions are valid.

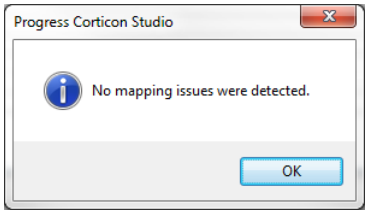
As you resolve issues, the system will remove warning messages from the Problems View until the Vocabulary is clean and free of warnings.

**Performing On-Demand Validation**

Once you have resolved all warnings, there is a good chance that your Vocabulary will be ready for use, but there is one final test you can perform. Select Vocabulary -> Database Access -> Validate Mappings and the system will perform a comprehensive validation of your Vocabulary via Hibernate. If Hibernate detects any errors, the system will display them to you. For example:




Continue to work through any errors until Hibernate states that your Vocabulary has no further issues:




When you see this message, your Vocabulary is ready for use.

**Setting Database Filters**

Rulesheet assets allow you to specify filters that trim down the information flowing into rules engine. Although filters can apply to information supplied in the input message, they are especially important for EDC-enabled applications. When filters are applied to database records, it can take a big load off of the rules engine, conserving working memory and improving performance.

In V4, the system would optionally delegate only the first filter (row 1) to the database, and only if a variety of complex conditions were met.

Corticon V5, filters are more powerful and flexible, because you can explicitly toggle one or many filter rows, thereby directly controlling which filters are processed by the database, as opposed to being processed by the rules engine.

The V4 Import Wizard never sets the database filter toggle, so it is necessary to revisit the Filter section of any EDC Rulesheets and set the database filters by hand. Since V5 allows any number of filter rows to be delegated to the database, this is a good opportunity to review all filters and determine which filters should be delegated to the database for best performance.

# Migrating enumerated value sets to Custom Data Types

Vocabularies created in Corticon BRMS 4.x might include Possible Values Sets defined for attributes. If such a Vocabulary is imported into this version of Corticon Studio, these Possible Values Sets will be converted into Custom Data Types in 5.x according to the following process.

Values in Studio 4.x **Possible Value Set Name (from Lookup Table)** fields are mapped directly to Corticon Studio custom data types. Should a v4.x attribute have a **Possible Value Set Name (from Lookup Table)** value, then this value is mapped directly to the Corticon Studio attribute's **Data Type** property value.

If the v4.x attribute's **Possible Values** property is not empty, then the upgrade process proceeds as follows:

1. If the Studio 4.x attribute's Possible Values contain only literal values (which may include `null` or `other` ) then a new enumerated custom data type will be created in the Corticon Studio Vocabulary as follows:
   a. Create a custom data type named `<AttributeName>Type`. In the event this name already exists, then the name will be made unique by adding `_1`, `_2`, etc.
   b. Set the **Base Data Type** of the Corticon Studio custom data type to the value of the v4.x attribute's **Data Type**.
   c. Set the **Enumeration** value to `Yes`
   d. Populate the **Values** list with the **Possible Values** of the v4.x attribute, *excluding* `null` and `other`.
   e. Set the **Data Type** of the Corticon Studio attribute to the newly created custom data type.
2. Else, where the Corticon Studio 4.x attribute's Possible Values does **not** contain only literal values, analyze the old Possible Values and determine the minimum possible value allowed and the maximum possible value allowed. If both minimum and maximum values are infinite and the set is unbounded on the both ends, set the Data Type of the upgraded attribute to the Data Type of the old attribute and ignore the Possible Values definition. Otherwise:
   a. Create a Custom Data Type named `<AttributeName>Type`. If this Custom Data Type name already exists, the name can be made unique by adding `_1`, `_2`, and so on.
   b. Set the Base Type of the newly created type to the Data Type of the old attribute.
   c. If the set has a lower and/or upper bound then add a Constraint Expression using the appropriate template:
      - value > *MinValue*
      - value >= *MinValue*
      - value < MaxValue
      - value <= *MaxValue*

- value in (*MinValue*..*MaxValue*)

### Upgrade Example A

Studio 4.x Integer attribute named `oddNos` with **Possible Values** = {`1,3,5,7,null,other`}. Qualifies for step 1 above.

Upon upgrade, a Corticon Studio enumerated custom data type is created named `oddNosType` of Base Type `Integer` and the `Value` entries = `{1,3,5,7}`

### Upgrade Example B

Studio 4.x Decimal attribute named `temperature` with **Possible Values** = `{<=98.6,99.0,>=100.0,null}`. Qualifies for 2.a

This set is unbounded. Therefore, no custom data type is created. The Corticon Studio attribute `temperature` receives **Data Type** = `Decimal`.

### Upgrade Example C

Studio 4.x Integer attribute named `age` with **Possible Values** = `{>0,20,30,<=120,null,other}`. Qualifies for 2.b.iii.7

Ignoring `null` and `other`, this is a constrained range with lower bound `0` (exclusive) and upper bound `120` (inclusive). Upon upgrade, a custom data type is created named `ageType` with **Base Type** = `Decimal` with the following **Constraint Expression** : `value in [0..120)`. Values `20`, `30`, `null`, and `other` are ignored.

# Converting extensions

The Corticon V5 extensions mechanism has been revamped; therefore, to use your V4 extensions in V5 you must:

- Convert your `CcExtensions.jar` to an Eclipse plug-in.
- Ensure that each of your extension classes implements certain `marker interfaces`.

See the *Corticon Studio: Extensions Guide* for complete details on implementing extensions in V5.