

# DTS Import Wizard User Guide

## Version 4.2

### *Introduction*

The Import Wizard DTS Editor plug-in, the standalone ImportWizard application, and the DTSImport batch file enable the importing of information into DTS Namespaces and DTS Subsets. For Namespace import, data can be imported into either new or existing (populated) Namespaces; Thesaurus and Ontology Extension Namespaces are supported. For Subset import, expression creation (from specified Concepts) and building are supported. Import data comes from either delimited text files (such as those resulting from TermWorks mapping sessions), Excel files (.xls or .xlsx) or specially formatted XML files (Namespace import only, files can be created by the TQL Editor or written independently).

### *Installation*

Extract the files in ImportWizard-4.2.zip into your *DTSInstall* directory. Be sure the Use folder names box is checked. This will place all the Import Wizard files into the appropriate folders:

<u>Folder</u>	<u>Files</u>
<i>DTSInstall</i> \bin\importwizard	DTSImport.bat ImportWizard.bat thesaurus-schema-v4.xsd extension-schema-v4.xsd subset-schema-v4.xsd
<i>DTSInstall</i> \lib\modules	importwizard.jar apelwizard.jar pluginutils.jar xmldigester.jar poi-3.6-20091214.jar poi-ooxml-3.6-20091214.jar poi-ooxml-schemas-3.6-20091214.jar xmlbeans-2.3.0.jar
<i>DTSInstall</i> \docs	importwizarduserguide.pdf
<i>DTSInstall</i> \docs\help	importwizarduserguide.htm

### *Importing Namespace Data into DTS*

Importing external data into DTS Namespaces typically consists of the following four steps:

1. **Data Modeling:** External data must first be translated from a current (if any) data representation into a DTS Data Model representation. This involves defining (conceptually) the basic elements (Concepts and Terms) of the data, their attributes and their relationships using the elements of the DTS Model: DTSPropertyTypes, AssociationTypes, Qualifiers, etc. **Appendix A** to this User Guide provides a brief tutorial on the creation of DTS Data Models.
2. **Data Preparation:** Next, the data to be imported must be prepared in the correct format. When loading via an XML file, this entails creating an XML file according to the DTS XML Schemas provided in the distribution kit. For more details see **XML Schema Files** in the **Namespace Importing from XML Files** section. (DTS XML files can also be created from existing DTS

Namespaces; see the **DTS TQL Editor User Guide** for further information.) The required structure of text or Excel import files is described in **Text/Excel File Structure** in the **Namespace Importing from Text and Excel Files** section below.

3. **Namespace Preparation:** Once the Data Model has been designed and the import file created, the DTS Namespace itself must be prepared. First, if not already present, the Namespace into which the data will be imported must be created. (Even this step can be eliminated if running a batch XML import and the “/N” option is used. See the **Using DTSImport** section of this Guide.) Since DTS XML files contain complete information on the DTS Data Model no further preparation is necessary for XML imports. For text or Excel imports, however, each DTS Attribute Type defined in the Data Model, e.g., Synonym Type, Property Type, Association Type, or Qualifier Type, must be created within the Namespace. Only if the Type name itself will be present in a tab position, i.e., an explicit Attribute Type, does the DTS Attribute Type not (necessarily) need to be created. Note that (a) only data associated with Attribute Types from (owned by) the *import Namespace* can be imported, and (b) Qualifier Types must always be pre-created, i.e., they cannot be explicit.

**Data Import:** Run the import using the Import Wizard plug-in (described in the **Using the DTS Import Wizard Plug-In** section of this Guide), the `ImportWizard` application (described in the **Using the DTS ImportWizard Application** section), or the `DTSImport` batch file (described in the **Using DTSImport** section). The Import Wizard automatically designates Root Concepts when importing from XML files. Otherwise, Roots can be set using the DTS Editor. See **Setting Namespace Roots** below.


## ***Importing Subset Data into DTS***

Importing data into DTS Subsets consists of creating a Subset’s Expression from Concept descriptors, e.g. Concept names, in the import file. Optional modifier fields can be used to specify that Children, Descendants, etc. of the base Concept are to be included in the Expression. The following two steps are used:

1. **Data Preparation:** Subset member information, i.e., list of member Concepts, is prepared in text or Excel import files as described in **Text/Excel File Structure** in the **Subset Importing from Text and Excel Files** section below. Concept modifiers can be included.
2. **Data Import:** Run the import using the Import Wizard plug-in (described in the **Using the DTS Import Wizard Plug-In** section of this Guide), the `ImportWizard` application (described in the **Using the DTS ImportWizard Application** section), or the `DTSImport` batch file (described in the **Using DTSImport** section). These sections describe the options available for Subset import.

## ***Running Imports***

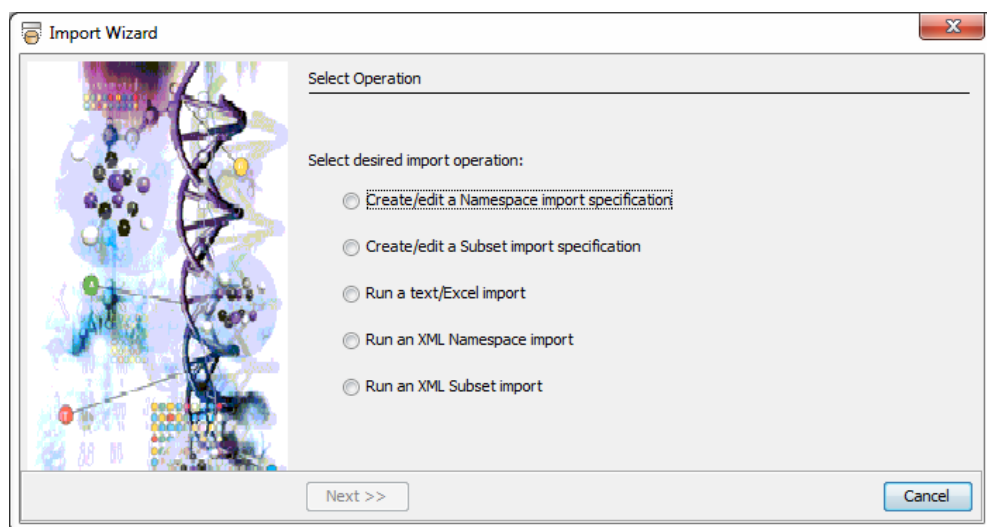
Import capabilities are accessed by:

- Creating an import specification and/or running an import using the DTS Import Wizard option in the DTS Editor. The `Import Wizard` can be found in the Tools menu and on the Icon Bar: . See **Using the DTS Import Wizard Plug-In** below for details.
- Creating an import specification and/or running an import using the `DTS ImportWizard` standalone application. This option is recommended for production imports of medium to large data sets as it avoids DTS Editor overhead associated with maintaining the GUI presentation. Imports run significantly faster using this option. See **Using the DTS ImportWizard Application** section for details.
- Running an import from a batch file that calls the `DTSImport` class (see, for example `DTSImport.bat` in the *DTSInstall\bin\importwizard* folder). Like the `ImportWizard` application, this option avoids `DTSEditor` overhead increasing import performance. See the **Using DTSImport** section for further details.

## Using the DTS Import Wizard Plug-In

After selecting the Import Wizard DTS Editor menu option (or clicking on the Import Wizard icon), you will be lead through a set of wizard pages. Click **Back** to go back to the previous page, **Next** to go to the following page. You may click on **Cancel** at any time to exit the wizard. Some options may not be available until certain selections are made in the page. The following paragraph, and the succeeding sections, describes the operation of the individual Import Wizard pages.

**Select Operation:** Five options are available: Create/edit a Namespace import specification, Create/edit a Subset import specification, Run a text/Excel import, Run an XML Namespace import, and Run an XML Subset import. Select the first option to create or edit a Namespace import specification. (Import specifications are only required for text and Excel imports.) If this option is selected, an import using the specification can subsequently be run without restarting the wizard. The second option performs similar functions for a Subset import. The third option will display the wizard pages to immediately perform a Namespace or Subset import from a text or Excel file. When running this option, an appropriate specification file must have already been created. The fourth and fifth options will initiate a Namespace (or Subset) import using an XML import file. Click **Next** to start the selected operation. See the **Namespace Importing from Text and Excel Files**, **Namespace Importing from XML Files**, **Subset Importing from Text and Excel Files** and **Subset Importing from XML Files** sections below for further instructions on these options.



## *Namespace Importing from Text and Excel Files*

### **Text/Excel File Structure**

Data to be imported into a Namespace from a delimited text or Excel import file must be structured so that each line/row describes data attribute values associated with a single Concept (or Term). This Concept/Term is called the line's Concept Key (or Term Key). Each tab position/column in the line then corresponds to either the *value(s)* or *name* of a DTS Attribute on that Concept/Term Key. ("Tab position" is used here for simplicity. In fact, any single character delimiter can be used to separate text fields.) The *name* (or other unique attribute) of the Concept/Term Key must be in one of the tab positions. Other tab positions are typically associated with the *value(s)* of a single DTS Attribute Type for all lines.

For example, if a tab position/column represents a text definition for a Concept, then that string might be assigned to the value of a DTS Property Type called `Definition`. Similarly, if a tab position/column is the name of a parent Concept for the line's Concept Key, then that name would be the value of a `Child of Concept Association`. Each imported *value* tab position/column must have a correspondence to a DTS Attribute Type; either an internal Type such as Namespace Name, Key Name, Key Code, Key ID, Primitive, or Defining Concept, or a user-defined Synonym Type, Property Type, Association Type, Defining Role Type or Qualifier Type. The tab position/column string associated with certain Attribute Types can also be a delimited list of values. In this case, each of the *values* in the list is treated as an independent *value* of the Attribute. See the description of the `Multiple` parameter in the **Parameter Dialog Boxes** section below for more information.

Rather than having a tab position/column always correspond to the same Attribute Type, it is possible to have tab position/column *values* be associated with the DTS Attribute Type *named* in another tab position/column (in the same line). See the **Parameter Dialog Boxes** section below for more information on specifying these explicit Attribute Types.

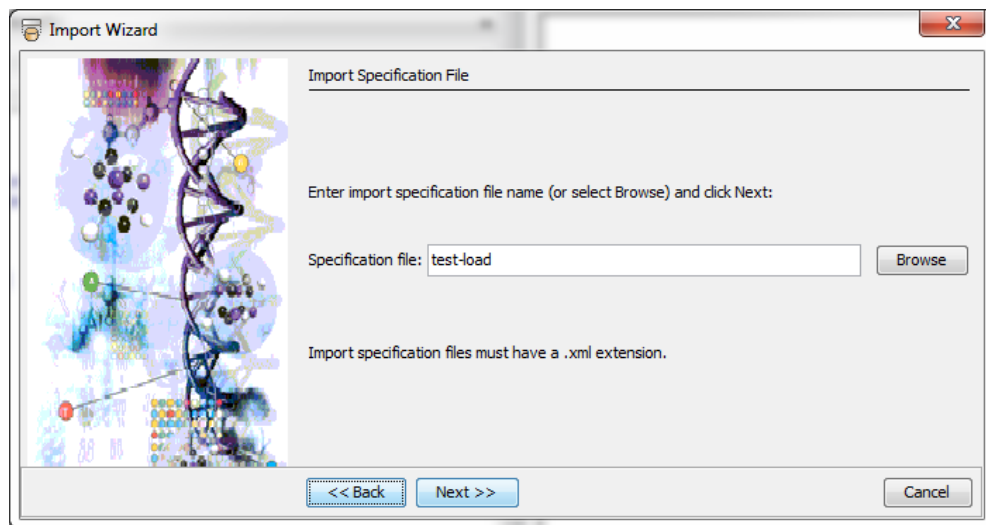
For implicit Attribute Types (where a tab position/column always refers to the same Type), the Attribute Type must have been previously created in DTS (see **Namespace Preparation** in the **Introduction** section). For explicit Attribute Types (where the name of the Type is itself in a tab position/column) the Attribute Type may or may not be previously created.

Multiple instances (*values*) of the same Attribute Type on a Concept/Term Key can be accommodated by using a list of *values* in the tab position/column, using additional tab positions/columns, or even using additional lines (as long as a unique identifier for the Concept/Term Key is the same in all lines). An example of a properly formatted file is a text export from a TermWorks Excel spreadsheet, or the Excel file itself. Finally, note that not all tab positions/columns must be imported, so only necessary positions need have valid Attribute correspondences.

## Text/Excel File Import Pages

This section describes the wizard pages used in text/Excel file Namespace imports.

**Import Specification File:** Namespace import from a text or Excel file is driven by an Import Specification. Import Specification files are XML files and can theoretically be created in any standard editor. It is recommended, however, that Specification files be created and edited (or at least validated) through the Import Wizard to avoid program errors. Enter the path and name of the desired specification file in the text field, or click the **Browse** button to search the file hierarchy. The wizard will supply a default extension of “.xml”. Click **Next** to continue.

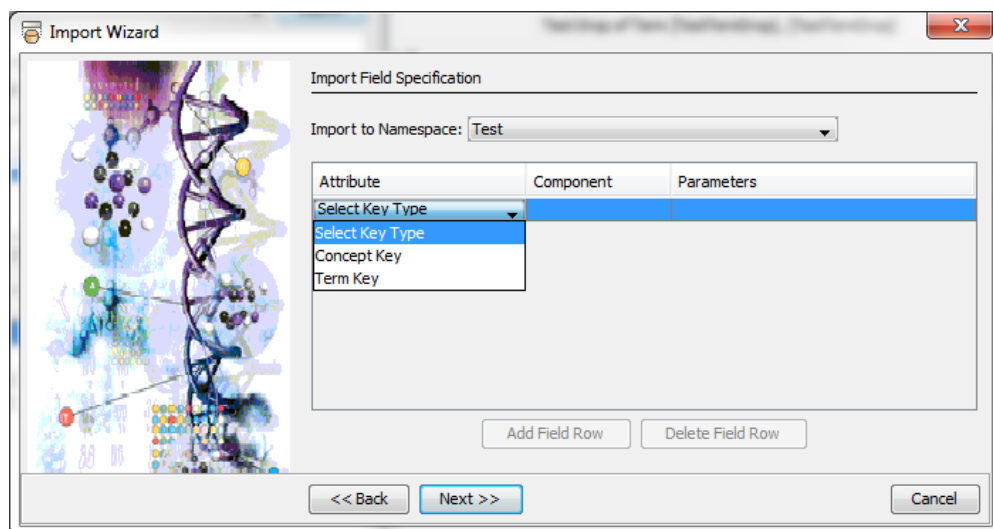


**Import Field Specification:** A Namespace import is defined by specifying (i) the (Local) Namespace to which data will be added, and (ii) the parameters of the DTS Attributes (Types, values, etc.) to be created and/or edited in the Namespace. These two types of data constitute the Import Specification which will be saved in the Import Specification File.

The Namespace is selected from the Combo Box at the top of the **Import Field Specification** page (see screen shot below), while the Attribute parameters (collectively the Field Specification) are selected and/or modified in the table at the center of the page. Usually, the values for the parameters come from fields (columns) in the import file. During import, each row of the import file is processed according to the Field Specification defined on this page. With one exception (see the **Blank** parameter below) each import row is processed independently.

Because of the complexity in supplying all of the parameters for a given Attribute class, e.g. Property or Association, each Attribute specification consists of one to four consecutive table rows. The rows associated with one Attribute class are called a row group. The cell in the first column of the first row of any group gives the DTS Attribute class associated with the group. The available Attribute classes are the Concept Key, Term Key, Action, Concept/Term Name, Concept/Term Code, Concept/Term Id, Concept/Term Status, Synonym, Property, and Association Attribute classes. For Ontylog Extension Namespaces, the Primitive, Defining Concept, and Defining Role Attribute classes are also available. Non-defining Roles may not be imported. The cells in the second column of the row group give the **Components** of the Attribute, and the cells in the third column give the **Parameters** used to specify the value of the component (see screen shot below). The details of the component structure for each Attribute are given later in this section.

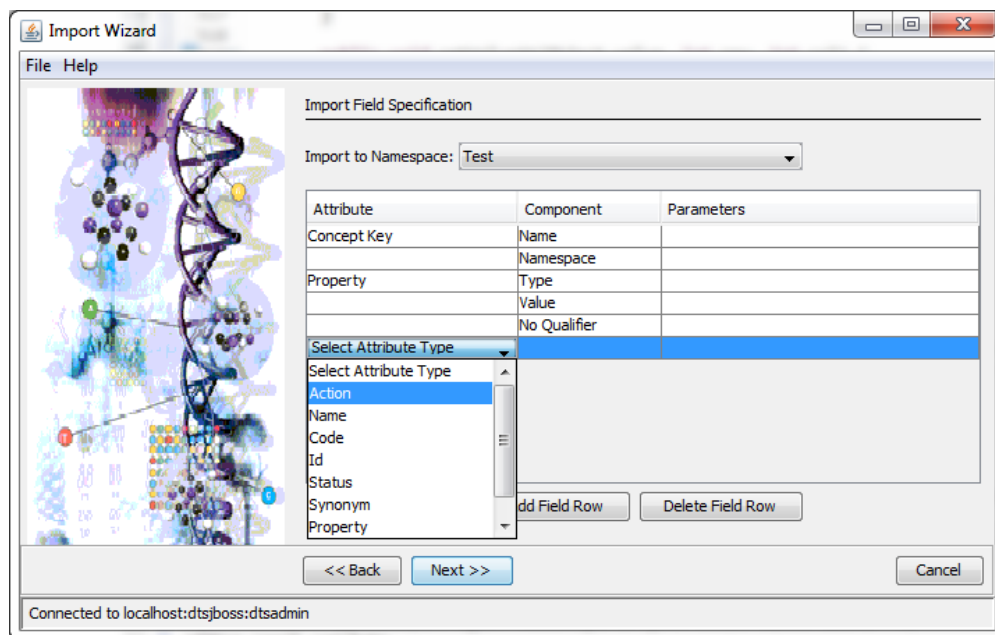
The first Attribute in any specification must be the Concept Key or Term Key Attribute. (This requirement is enforced by the table logic – it is impossible to delete or move this Attribute.) A single specification can only be used to import Concepts and their Attributes, or Terms and their Attributes. Both are not supported in the same specification. The Concept/Term Key Attribute is a special Attribute that designates the target object for the other Attributes. This is the Concept or Term to which the other Attributes (Code, Id, Properties, Synonyms, and Associations) in the specification will be attached. Double click on the Select Key Type cell of the table (row 1, column 1) to choose either Concept Key or Term Key (see screen shot below).



Additional Attribute row groups can be added to the specification by selecting an existing row group (click once on any cell in the group, the rows in the group will be highlighted) and pressing the Add Field Row button. This will add a single new row below the selected row group. To delete an Attribute from the specification, select the Attribute's row group as described above and press the Delete Field Row button. With the exception of the initial Concept/Term Key Attribute, the order of the other Attributes in the specification is immaterial.

To create a new Attribute specification, double click the Select Attribute cell in the first column of the new row, and select the desired Attribute class from the drop-down. The Attribute's components will then appear in the second column with additional rows added as needed for the group. The Attribute associated with an existing row group can be modified by again double clicking the Attribute class name cell and selecting a different Attribute class. The previous Attribute row group is deleted and a new, empty, row group created.

The screen shot below shows an example Namespace Import Field Specification panel with the Concept Key and Property Attribute classes and the Attribute class drop-down.



The steps to complete the specification for each Attribute class are described below:

#### Concept/Term Key Attribute Class

The Concept Key or Term Key Attribute class has two components, a Name and a Namespace. The Name component represents the name of the concept/term and the Namespace component represents its Namespace. Note that this Namespace can be, and often is, different from the specification's Namespace. To specify the parameters that will define the import values for either of these components, double click in the Parameters (third column) cell for the component. This will bring up an associated parameters dialog box. See the **Parameters Dialog Boxes** section below for a description of this dialog. After the dialog has been completed, an encoded string representation of the selected parameters is placed in the component's Parameter cell.

#### Action Attribute Class

The Action Attribute class is used when the import file specifies standard transaction (CRUD - Create, Replace, Update and Delete) operations. The operation type is decoded by translations as specified in the single Value component. Double click in the component's Parameters cell to define the operation's field location and names. See the **Parameters Dialog Boxes** section below for a description of this dialog and the **Using Actions** section for details on Action processing. Note: Only one instance of the Action attribute is permitted in any specification.

#### Name Attribute Class

This row group is used to rename the Key Concept or Key Term, and to supply a Concept/Term Name when new Concepts or Terms are to be added to the knowledge base and the Encoded parameter is used in the Key Concept/Key Term Attribute (see the discussion of the Add and Encoded parameter in the **Parameters Dialog Boxes** section below). Double click in the Parameters cell to specify the Value component's parameters.

#### Code and Id Attribute Classes

These optional single row component Attribute classes are only used when new Concepts or Terms are to be added to the knowledge base and the values for the object's Code and/or Id are to be explicitly given. The Add parameter in the associated Key Concept or Key Term Attribute class



MUST be present to use these Attributes. If the run-time value of the `Code` or `Id` Attribute is not the empty string, then this value is used on Concept/Term creation. If the run-time value is the empty string, the final value is determined by the default generator. Double click in the `Parameters` cell to specify the `Value` component's parameters. For further information on these Attribute classes, see the discussion of the parameter in the **Parameters Dialog Boxes** section below.

#### Status Attribute Class

This optional, single row component can be used to set or change the Status of a Concept or Term. Status values of `ACTIVE`, `INACTIVE` or `DELETED` are supported using translations that are specified in the `Value` component. Double click in the `Parameters` cell to specify the `Value` parameters. See the **Parameters Dialog Boxes** section below for further details. Only one instance of the Status attribute is permitted in any specification.

#### Property Attribute Class

This row group represents a Property to be added to the key Concept or Term. A Property Attribute Class has three components, a `Type`, `Value` and a `Qualifier`. The `Type` component is the DTS Property Type of the Property that will be created. This Type can be either fixed (the same for all input lines) or explicit (taken from a field in the data file) depending on the setting of the `Parameters`. (See the discussion in the **Parameters Dialog Boxes** section below.) The `Value` component represents the value to be assigned to the resulting Property. The `Qualifier` component cell is actually another drop down Combo Box. Double click on this cell (shown initially as `No Qualifier`) to select the desired Property Qualifier Type (if any) for the Property. Then, as above, double click in the respective `Parameters` cell to specify a component's parameters.

#### Synonym Type Attribute Class

This row group represents a Synonym to be added to the key Concept. The Synonym Type Attribute Class has three components, its `Type`, its `Term`, and its `Preferred` state. The `Preferred` component permits specification of the preferred state from a column in the input file. The preferred state can be updated on an existing Synonym. Double click in the respective `Parameters` cell to specify a component's parameters.

#### Association Type Attribute Class

This row group represents an Association, or an Inverse Association, to be added to the key Concept or Term. The Association Attribute class has four components, a `Type`, a `Name`, a `Namespace`, and a `Qualifier`. As with the other classes, the `Type` component is the (Inverse) Association Attribute Type. The `Name` component represents the name of the "to" Concept/Term (or "from" Concept/Term in the case of Inverse Associations) of the Association and the `Namespace` component represents its Namespace. The `Qualifier` component represents the Association's Qualifier as described in the Property Attribute Class section. Double click in the respective `Parameters` cell to specify a component's parameters.

#### Primitive/Defined Attribute Class

This row group (only available when the import Namespace is an Ontylog Extension Namespace) represents the value of the Primitive attribute on the key Concept. The group has only one component, its `Value`. Double click in the `Parameters` cell to specify the Attribute's parameters. The import values "true" or "primitive" (case-insensitive) declare the key Concept to be Primitive. Any other value sets the Concept to Defined. Double click in the `Parameters` cell to specify the `Value` component's parameters.

#### Defining Concept Attribute Class

This row group (only available when the import Namespace is an Ontylog Extension Namespace) represents a Defining Concept for the key Concept. The group has two components, the Defining Concept's Name and Namespace (the Namespace must be either the import Namespace or the import Namespace's linked Namespace). Double click in the respective Parameters cell to specify a component's parameters.

#### Defining Role Attribute Class

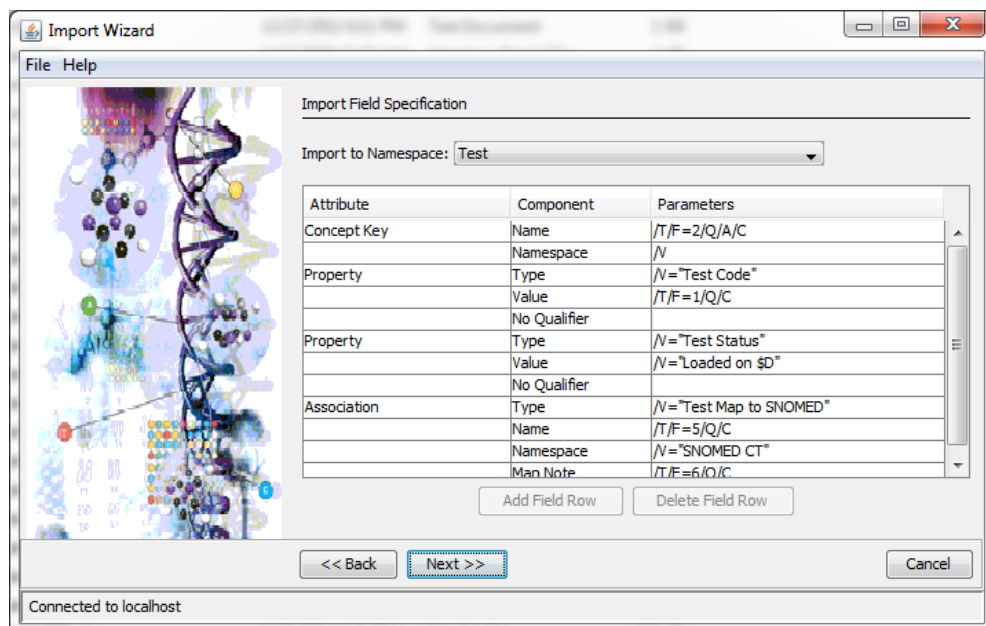
This row group (only available when the import Namespace is an Ontylog Extension Namespace) represents a Defining Role for the key Concept. The group has five components, the Defining Role's Type, its target's Name and Namespace, its Modifier and, optionally, its Group. The Role Type may be drawn from either the import Namespace's base or linked Namespaces. Parameters for the first three components may be set by double clicking in the respective Parameters cell. The Modifier component should generally be kept as "some", but "all" can be selected by double clicking in the Modifier Component cell. Defining Roles cannot be added to existing Groups; only new Groups can be created. To associate a Defining Role with a (new) Group, double click in the Group Component cell and select Group. Then double click in the Parameters cell to set the Group value parameters. The value for a Group can be any String; the value is only used for identifying the other Defining Roles in the Group. The following points should be considered in using Defining Role Groups:

- All Defining Roles in the Group must be in the same import line.
- The context for a Group value string is the current line; i.e., the same Group value can be used in different lines.
- There must be at least two Defining Roles in a Group.
- An empty Group value ("") is permitted and the Defining Role is interpreted as No Group.

Note that, for convenience of specification, if any Namespace Attribute component value is the empty (blank) string, the name of the import Namespace is used.

Press Next when the specification is completed.

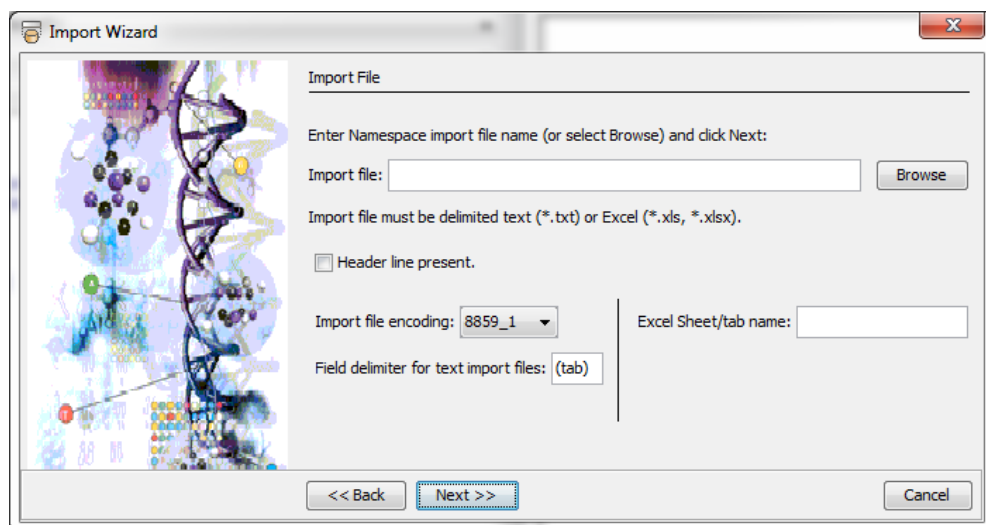
The screen shot below shows a completed Namespace Import Specification.



Specification Complete: Click on the **Finish** button to simply save the specification and exit. Click on the **Next** button to save the specification then go on to run an import.

Import File: The import file is the file from which import values will be taken. Enter the file name or click the **Browse** button to search the file hierarchy (see below). The file must be a delimited text (.txt) file or Excel (.xls or .xlsx) file. If the import file has a header row, check the **Header line present** box (see screen shot below). This will cause the importer to ignore the first line of the import file.

Based on the type of import file, either the left or right additional options will be enabled. For a text file, these are the file encoding and field delimiter. First select the file's character encoding. Most files will be encoded in ASCII Latin 1 ("8859\_1"), although UTF8 encoding is also supported in the combo box..Next, enter the (single character) field delimiter. The default field delimiter is the tab character, shown on the page as (tab). If an Excel file is selected, optionally enter the name of the Sheet (or Tab) to be used. If this field is left blank, the Excel default (first) sheet is used.



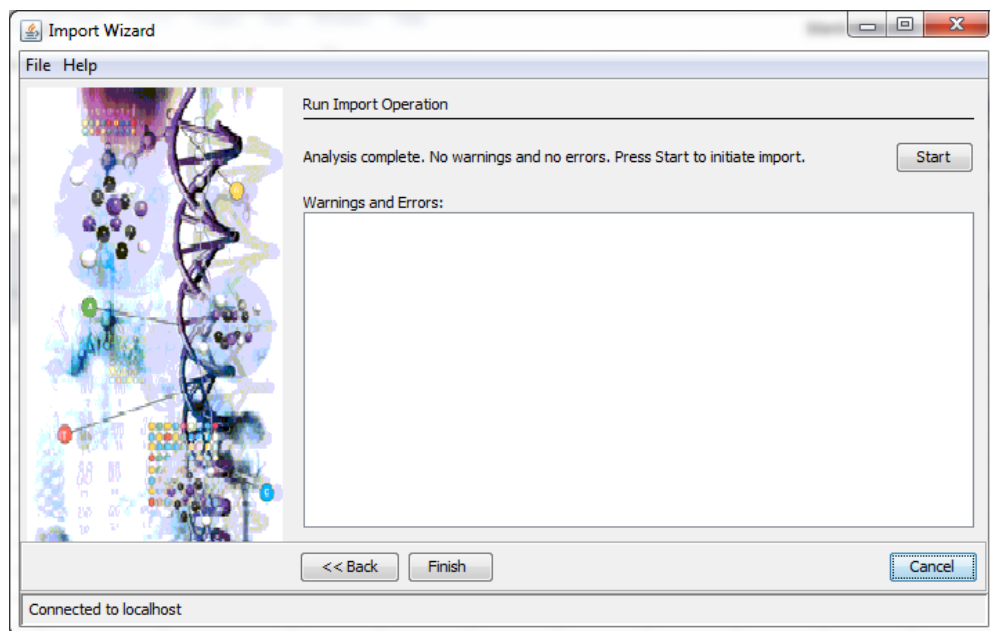
Confirm Import Operation: Press **Next** to start the import operation.

Run Import Operation: Importing is performed in two steps, a data validation step followed by the actual import step. During the first step, the specification file is read and checked for errors. Any errors are displayed in the page's text box (see screen shot below). Any specification error is "fatal" in the sense that it will prohibit subsequent checking of the import file. A specification error can result, for example, if a selected DTS Attribute Type is deleted from the Namespace between the time the specification was created and then run, or if the specification was improperly edited outside of the wizard.

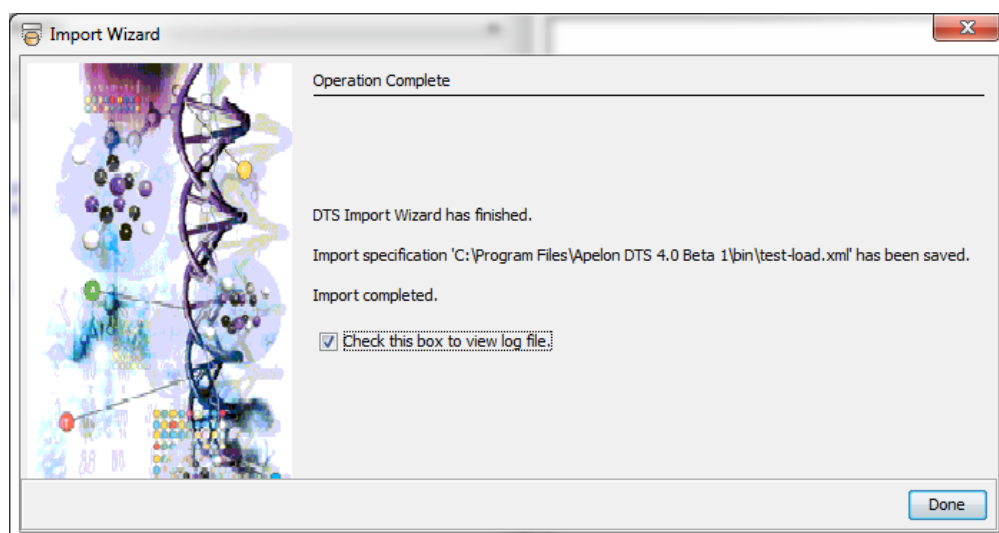
Once the specification file has been successfully processed, the import file is read and validated. All import processing is performed, EXCEPT actually creating Namespace objects. Any errors identified, for example a null or unknown Namespace name or null Property value, are reported in the text area. These errors are not fatal; an error will only prevent processing of the associated Attribute. Processing of other Attributes on the input row will be performed. Note, however, that an error in Concept/Term Key processing will prevent processing of the whole input row.

Based on the errors found, press **Cancel** to terminate the wizard or press **Start** to initiate the actual import step (see screen shot below). During importing, the importer writes messages to a log file in the execution directory. This file contains a record of all import actions including errors detected and Attributes created. By default, the importer uses a "rolling daily" log protocol: a new log file is created every day (at midnight) and all log entries for all imports run during the day are written to this file. The format of the log file name is "DTSImport-2012-12-25.log". The log file name can be overridden when using the `DTSImport` class from a batch file. See the **Using DTSImport** section for details.

During either import step, press the **Stop** button (renamed from **Start**) to terminate processing. No further import processing will be performed.



Operation Complete: If an import was run, the daily import log file can be viewed by selecting the check box and clicking **Done**. Note that this file may contain entries from previous imports. The entries from the current import will be at the bottom of the file.



## Parameter Dialog Boxes

Double clicking in a `Parameters` table cell opens a Parameter Dialog box. There are different Parameter Dialog box formats for each Attribute class and component combination. Every Parameter Dialog box, however, contains at least two of three standard sections, each with a set of parameters appropriate for the associated Attribute class/component. The paragraphs below describe all the possible parameters in each section. Refer to Figure 1 below for representative examples of Parameter Dialog boxes.

The first section is the `Source` section. All Parameter Dialog boxes contain this section. The `Source` section is used to specify the source for the string value of the Attribute component. The value is specified as coming either from a field in an import file row or an explicit literal (see the example boxes in Figure 1). Note: for the `Action` Attribute class, the explicit literal option is not available.

If the value is to be taken from the input file, select the `Field` radio button and use the “spinner” to select the (one-based) input field number. A number of additional parameters (see below) can be associated with this selection. The specific set of parameters displayed in a given dialog is dependent on the dialog’s underlying component type.

- |            |   |
|------------|---|
| CRUD:      | This area is only available for the <code>Action</code> Attribute class. For each of the four CRUD actions, enter the (case-insensitive) name or phrase used in the import file to denote the action. E.g., the name for the <code>Delete</code> action could be “remove”. The default names are “CREATE”, “REPLACE”, “UPDATE” and “DELETE”.  |
| Status:    | This area is only displayed for the <code>Status</code> Attribute class. For each of the three Status values, enter the (case-insensitive) name or phrase used in the import file to denote the status. E.g., the name for the <code>Deleted</code> status could be “del”. The default names are “ACTIVE”, “INACTIVE”, and “DELETED”.   |
| Preferred: | This line is only displayed for the <code>Preferred</code> component of the <code>Synonym</code> Attribute class. Enter the (case-insensitive) name or phrase used in the import file to denote a preferred state. If there is no import value or the value does not match the entered phrase, the preferred state is not set or changed. The preferred state from this operation is ORed with the state from the <code>Mark as Preferred</code> <code>Synonym</code> checkbox from the <code>Synonym Value</code> component (see discussion below) before the state is committed to the Knowledgebase. |
| Blank:     | If selected, when the designated import field is empty (null string) the value will be taken from the same field number in the previous input row. This operation is applied iteratively until a non-null value is found. Among other uses, this facilitates loading of certain TermWorks data files. This parameter is the sole exception to the rule that each line in the import file is processed independently.  |
| Required:  | If selected, a value is required. A null value will result in an error. If not selected, null values will be ignored, i.e., the associated Attribute component will not be created. This parameter is not available on the <code>Concept Key</code> or <code>Term Key</code> Attributes. The <code>Required</code> parameter is tested AFTER the <code>Blank</code> operation (if selected) has been performed.   |
| Add:       | If selected, when the named <code>Concept Key</code> , <code>Term Key</code> or <code>Attribute Type</code> does not exist in the Namespace, it is added. This parameter is only present for the <code>Concept</code>   |

Key Value, Term Key Value and Type components. For the Concept Key Value and Term Key Value components, Concepts or Terms will only be added when the Concept/Term Namespace is the import Namespace. Existence is determined by testing the supplied Attribute Value as a Concept or Term Name, or as the encoded Attribute if Encoded is also selected (see below). If the Concept/Term does not exist using Encoded, then a Name Attribute class must also be present and the value of this Attribute is used as the new Concept/Term Name. Since Term Names are not necessarily unique, an Add operation is also executed if Add is selected, and the Name exists. This enables new Terms to be added to the database that have duplicate Names, but unique associated Code and/or Id values. If the Code and/or Id Attribute classes are present in the specification, the Concept/Term Code and/or Id are taken from the attribute import values, if any. See the description of the Action Attribute class for related functionality.

- Delete: This parameter is only present on Concept Key Value and Term Key Value components. If selected, all Concepts/Term keys designated in the import file are deleted. No other Attribute groups are permitted in the specification if Delete is selected in the Concept Key Value or Term Key Value component. See the description of the Action Attribute class for related functionality.
- Encoded: This parameter is available for the Value (Name) component of a Concept- or Term-valued Attribute class, e.g. Term Key, Concept Key, Synonym, Defining Concept, Defining Role, or Association. If selected, the value from the import field is interpreted not as a Concept or Term Name, but as the value of an associated attribute. This attribute can be the element's Code, Id or DTS Property. The DTS Knowledgebase is searched for a Concept having this attribute. The attribute, e.g., Code, Id or Property Type name, is given by the selected entry in the Encoded combo box. If the specified attribute value is found, the component Value is the associated Concept or Term. If there is no such attribute value, and the Add parameter is not present, an error is thrown. If there is no such value and Add is present, a new Concept/Term is created as described in the Add section above. If there are multiple elements having the Property value, an error is reported. This parameter typically enables the import field to be an alternate encoding for a Concept, e.g., the Code in Source value of an ICD-9-CM Concept. The Encoded parameter also enables disambiguation of similarly-named Terms when specifying Term Keys or Synonym targets. Note that the correct Namespace for the Attribute's target Concept must still be specified in the Attribute class' Namespace component line. For a Concept or Term Key Attribute, Encoded is disabled if the Add option is selected.
- Multiple: This parameter is available for the Value (Name) component of Defining Concept, Defining Role, Synonym, Property and Association Attribute classes. If selected, the value from the import field is interpreted as a list of values, separated by the designated (single-character) delimiter. (This delimiter cannot, obviously, be the same as the field delimiter of the import file.) Each (sub) value in the list is acted on, using the other parameters of the component specification, as an independent value of the associated Attribute class. This so-called "micro-syntax" allows multiple instances of Attributes to be created from one import field.

If the data value is to be explicit, i.e., a literal or constant value, select the Use ... Value radio button. This parameter is not present for the Concept Key Value and Term Key Value components. For Type and Namespace Name components, select the desired Type or Name from the combo box. The box holds the existing Types for the Attribute class, or local Namespace Names, found in the knowledgebase. Selection of the “(Import Namespace)” Namespace value in the Namespace Name combo box will save an empty (blank) string as the parameter value which will be replaced at import by the name of the import Namespace. For other components, enter the literal value in the text box. Literals can be helpful as Property and Qualifier values. Any string is acceptable, except it may not contain double quotes. The literal string value may also contain certain special codes. These codes are replaced during the import with system values as described in Table 1. Thus, the literal “updated @D” would be converted to “updated 12/25/2004”.

For Property, Synonym, Defining Concept, Defining Role, Association and Qualifier Name/Value Parameters, three “radio” buttons are displayed that further specify the operation to be performed on the Attribute. If the Set attribute instance box is checked (the default), the Attribute (a Property, Synonym, Defining Concept, Defining Role, Association or Qualifier) is *set*, i.e., a new instance of the Attribute is always created on each selected key Concept or Term. When Update attribute is selected, however, a new Attribute instance is only created if there *is not* an existing Attribute of the designated Attribute Type on the Concept/Term. If there *is* any occurrence of the Attribute Type, all such occurrences will be updated to (replaced by) a single instance of the Attribute Type with the specified name/value. If Delete attribute instance is selected, all instances of the Attribute Type having the specified name/value will be *deleted*. Note that the Update attribute option is not available on Defining Concept Attributes.

Finally, if the Parameter Dialog Box is a Synonym Value component, a Mark as Preferred Synonym checkbox is displayed at the bottom of the section. Check this box to designate that the synonym is a preferred synonym for the associated key Concept. The preferred state from this operation is ORed with the state from the Preferred component (if any, see above) before the state is committed to the Knowledgebase.

The second section is Filters. All Parameter Dialog boxes contain this section. The parameters in the Filters section provide data formatting and conversion operations that are applied to the value resulting from the first section. If Multiple has been selected in the associated Source parameter specification, the filters are applied each individual (sub) value in the list. The following filters are applied *in the order below*:

- |          |  |
|----------|--|
| Trim:    | If selected, spaces are removed from the beginning and end of the value.   |
| Quote:   | If selected, enclosing double quotation marks are removed from the value. Paired double quotation marks (as may be added by the Excel export process) are also replaced by single occurrences.   |
| Control: | If selected, control characters are removed from the value.  |
| Regex:   | If selected, the value is searched for matches to the first, Regular Expression, field. Matched characters are replaced by the second RegEx string, which can be empty or can contain references to the match. For example, to remove all punctuation in a dollar amount such as “\$12, 492.88” use a RegEx of [ \$, . ] and a blank replace |



string. Or to add a decimal point before the last two digits of an ICD-9-CM code, use a RegEx of `\d{2}$` and a replace string of `.$0`.

- Upper: If selected, all alphabetic characters in the value are converted to uppercase.
- Lower: If selected, all alphabetic characters in the value are converted to lowercase.
- Word: If selected, all alphabetic characters in the value are converted to lowercase, then the first letter in each word is converted to uppercase.
- Sentence: If selected, all alphabetic characters in the value are converted to lowercase, then the first letter in the first word is converted to uppercase.

The last section is `Validators`. This section is present in most `Value` components and in `Concept/Term Key Name` components. These parameters test that the value (after processing by the filters) meets certain format requirements. If the value does not meet these requirements, an error is thrown; no attempt to “convert” the value to the format is attempted.

- Integer: If selected, the value must be in the form of a signed integer and must be between the given Minimum and Maximum limits (inclusive). Either limit, or both, may be blank.
- Number: If selected, the value must be in the form of a signed decimal number and must be between the given Minimum and Maximum limits (inclusive). Either limit, or both, may be blank.
- RegEx: If selected, the (entire) value must match the given Regular Expression.

The `Filters` and `Validators` sections are only enabled when the attribute value is to be taken from the input file, i.e., the `Field` radio button is selected.

The following buttons are available at the bottom of the `Parameter Dialog` boxes:

- Save Validates the parameters, and if valid, closes the dialog and puts an encoded representation of the parameters in the selected `Parameters` table cell. The underlined character in each parameter description is the abbreviation returned by the dialog box to the table cell. See the screen shot at the end of the `Import Field Specification` page description for examples of encoded parameter strings.
- Clear All Clears/resets all the parameters in the dialog.
- Test Opens a `Test Value` dialog box. This dialog prompts for a sample value, performs the specified `Filters` and `Validators`, and then reports the results, i.e., the filtered value and the result of validation.
- Cancel Closes the dialog box without saving any changes.

## Using Actions

The `Action` Attribute class is used when an import file includes CRUD transaction designators. *Every line* in the import file must include a CRUD operation designator in the cell/field location specified in the `Action Value` parameters (see Figure 1). The associated operation cell/field values must also be as given in the `Value` parameters.

Each `Action` operation defines specialized `Concept/Term Key` processing on the import line. After this key processing, processing of other import attributes proceeds normally except as noted below. The following steps are performed for each operation:

- CREATE:** The `Concept/Term Key` is created. It is not necessary for the `Add` parameter on the `Concept/Term Key` attribute to be selected. An error is thrown if the `Concept/Term Key` already exists.
- REPLACE:** If the `Concept/Term Key` does not exist, an error is thrown. Otherwise, the existing `Concept/Term Key` is deleted and a new `Concept` or `Term` is then created.
- UPDATE:** If the `Concept/Term Key` does not exist, an error is thrown.
- DELETE:** If the `Concept/Term Key` exists, it is deleted. It is not necessary for the `Delete` parameter on the `Concept/Term Key` attribute to be selected. Further processing of the import line is not performed, i.e., any existing cells/fields are ignored.

## Setting Namespace Roots

The last task after importing data into a DTS Namespace is to designate, if desired, the root Concepts of the Namespace. For the DTS Browser and DTS Editor, root Concepts are the top of the default “Parent Of” Association display tree. A namespace can have any number, including zero, roots.

Namespace Roots can be set in either the DTS Editor (the Namespace Editor and Namespace Details panels) or with the DTS API. See the **DTS Editor User Guide** and the DTS API Javadoc for further information.

**Table 1 –Special Codes for use in Literal Values**

<i>Code</i>	<i>Replaced With</i>
@D	The import date as mm/dd/yyyy.

**Concept Key Value Parameters**

Source

Use value from import file Field:

☐ If Blank, use previous concept/term value.

☐ Add Concept/Term if not present (must be in import Namespace).

☐ Delete Concept/Term.

☐ Value is encoded via:

Filters

☐ Trim leading and trailing whitespace. ☐ Set case to UPPER.

☐ Remove leading, trailing, and doubled Quote marks. ☐ Set case to lower.

☐ Remove Control characters (except new line). ☐ Set case to Word.

☐ Regex:  with  ☐ Set case to Sentence.

Validators

☐ Validate to an Integer Min value:  Max value:

☐ Validate to a Number Min value:  Max value:

☐ Validate with RegEx RegEx:

**Action Value Parameters**

Source

☒ Use value from import file Field:

☐ Require a value (empty is error).

Create:

Replace:

Update:

Delete:

☐ Use literal Value:

Filters

☐ Trim leading and trailing whitespace. ☐ Set case to UPPER.

☐ Remove leading, trailing, and doubled Quote marks. ☐ Set case to lower.

☐ Remove Control characters (except new line). ☐ Set case to Word.

☐ Regex:  with  ☐ Set case to Sentence.

**Synonym Value Parameters**

Source

☒ Use value from import file Field:

☐ If Blank, use previous concept/term value.

☐ Require a value (empty is error).

☐ Value is encoded via:

☐ Multiple values with delimiter:

☐ Use literal Value:

☒ Set attribute instance. ☐ Update attribute. ☐ Delete attribute instance.

☐ Mark as Preferred Synonym.

Filters

☐ Trim leading and trailing whitespace. ☐ Set case to UPPER.

☐ Remove leading, trailing, and doubled Quote marks. ☐ Set case to lower.

☐ Remove Control characters (except new line). ☐ Set case to Word.

☐ Regex:  with  ☐ Set case to Sentence.

Validators

☐ Validate to an Integer Min value:  Max value:

☐ Validate to a Number Min value:  Max value:

☐ Validate with RegEx RegEx:

**Association Value Parameters**

Source

☒ Use value from import file Field:

☐ If Blank, use previous concept/term value.

☐ Require a value (empty is error).

☐ Value is encoded via:

☐ Multiple values with delimiter:

☐ Use literal Value:

☒ Set attribute instance. ☐ Update attribute. ☐ Delete attribute instance.

Filters

☐ Trim leading and trailing whitespace. ☐ Set case to UPPER.

☐ Remove leading, trailing, and doubled Quote marks. ☐ Set case to lower.

☐ Remove Control characters (except new line). ☐ Set case to Word.

☐ Regex:  with  ☐ Set case to Sentence.

Validators

☐ Validate to an Integer Min value:  Max value:

☐ Validate to a Number Min value:  Max value:

☐ Validate with RegEx RegEx:

**Synonym Type Parameters**

Source

☒ Use Type name value from import file Field:

☐ Add attribute Type if not present (must be in import Namespace).

☐ Use literal Value:

Filters

☐ Trim leading and trailing whitespace. ☐ Set case to UPPER.

☐ Remove leading, trailing, and doubled Quote marks. ☐ Set case to lower.

☐ Remove Control characters (except new line). ☐ Set case to Word.

☐ Regex:  with  ☐ Set case to Sentence.

**Namespace Name Parameters**

Source

☐ Use value from import file Field:

☐ If Blank, use previous concept/term value.

☐ Use existing Namespace Value:

Filters

☐ Trim leading and trailing whitespace. ☐ Set case to UPPER.

☐ Remove leading, trailing, and doubled Quote marks. ☐ Set case to lower.

☐ Remove Control characters (except new line). ☐ Set case to Word.

☐ Regex:  with  ☐ Set case to Sentence.

**Figure 1 –Parameter Dialog Boxes**

## *Namespace Importing from XML Files*

### **XML Schema Files**

The Import Wizard distribution kit includes two XML Schema (.xsd) files that can be used to create and validate XML namespace import files. The two file formats are briefly described below:

thesaurus-schema-v4.xsd	This schema is used to load data into a Thesaurus Namespace. Attribute data (Synonyms, Properties, and Concept Associations) associated with Concepts and Terms are structured within their respective primary objects. Attributes which are not connected to a Concept or Term in the Namespace (including Namespace Properties, Namespace Version Properties, and “non-local” Properties and Concept Associations) are structured in separate elements.
extension-schema-v4.xsd	This schema is used to load data into an Ontylog Extension Namespace. Attribute data (Synonyms, Properties, Concept Associations, Defining Concepts and Defining Roles) associated with Concepts and Terms are structured within their respective primary objects. Attributes which are not connected to a Concept or Term in the Namespace (including Namespace Properties, Namespace Version Properties, and “non-local” Properties and Concept Associations) are structured in separate elements.

### **XML File Import Pages**

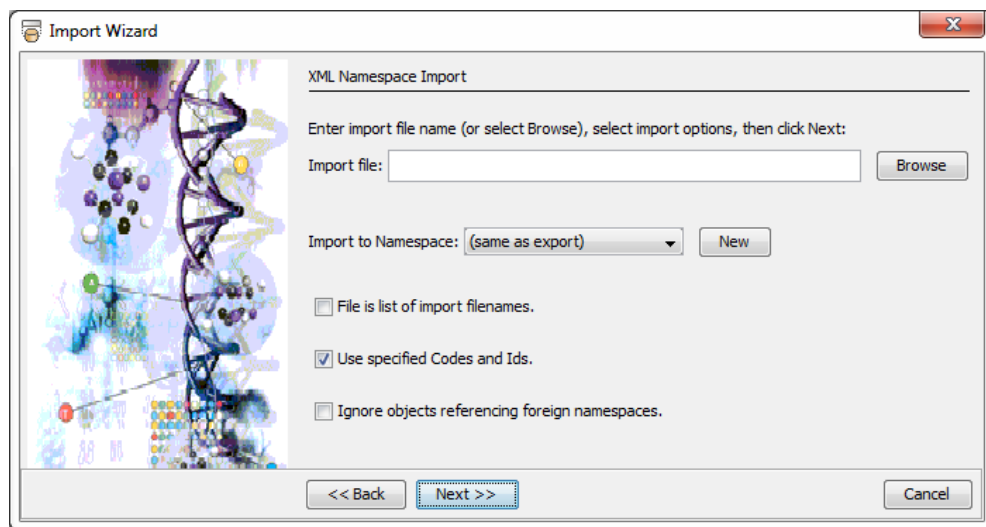
This section describes the wizard pages used in XML namespace file imports.

**XML Namespace Import:** XML data files can also be used to load terminology Objects and Attributes into DTS Namespaces. These files are typically created using the TQL Editor plug-in module but can also be created independently by following the XML Schema specifications (\*.xsd files) in the distribution kit. See the **TQL Editor User Guide** for further information on creation of export data files, and **XML Schema Files** above for information on these files.

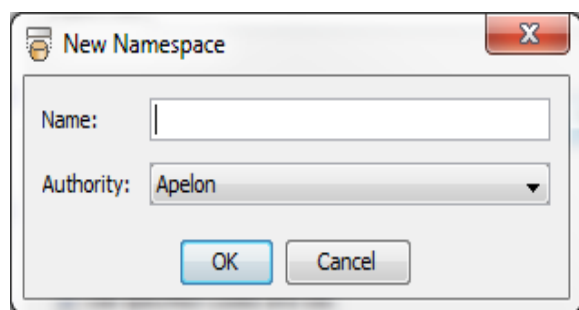
The XML data files contain all the definitional information necessary to load DTS Namespace Objects and Attributes. Importing of Namespace Properties and Namespace Version Properties are supported for all local, writable Namespaces. Note that since only the ‘Working’ Namespace Version is writable in a local Namespace, saved Version Properties (from any named Version) are always written into the Working Version of the target Namespace. For Thesaurus Namespaces, all Property Type, Synonym Type, Association Type and Qualifier Type definitions from the source Namespace are loaded as well as all Concepts, Terms, Properties, Synonyms, and Associations. For Ontylog Extension Namespaces, the Primitive Attribute, Defining Concepts and Defining Roles are also included. Since XML imports create all necessary Attribute Types, they are normally made to newly-created (empty) Namespaces.

The XML Namespace Import page (see screen shot below) collects all the additional (operational) information needed to execute an import. The import file name is entered into the top text field (or press the Browse button to search the file hierarchy). Next, select the target Namespace from the Import to

Namespace Combo Box. All DTS objects associated with the source Namespace of the data file will be created in the selected target Namespace.



To have the wizard create a new target Namespace, press the **New** button. . (This button is only enabled if the user has the `NAMESPACE_ADMIN` permission.) This opens the **New Namespace** dialog (see left). Enter the name of the Namespace to be created in the text field and select the new Namespace's Authority from the Combo Box. Click **OK** to accept the values or **Cancel** to quit. The new name will be loaded into the **Import to Namespace** Combo Box and the selected Authority will be shown to the right of the **New** button. Note that any subsequent Namespace selection from the Combo Box, or new Namespace addition, will remove the previous temporary entry.



There are three additional XML load options specified by the checkboxes at the bottom of the page. Select **File is list of import filenames** to tell the wizard that the import file is text file containing a list of XML file names. During import, these files will be processed as a group rather than independently. This option enables two (or more) interconnected Namespaces to be loaded. If the imports were done separately, inter-Namespace relationships (DTS Associations) could not be resolved.

Select **Use specified Codes and Ids** to create DTS Type and Concept/Term Objects with the same Codes and Ids as were used in the original source Namespace. This box is true by default and should be used to ensure that an exact duplicate of the source Namespace is created. This box **must be checked** if Synonyms are to be created to multiple Terms having the same Name. If this is not checked, Codes and Ids are created automatically by the current DTS Code and Id generator, and Synonyms to similarly-named Terms will not be created. Note that the setting of this box **does not apply** to the creation of a new Namespace. Namespaces created by the Import Wizard always have their Id set by the internal Id generator.

Select **Ignore objects referencing foreign namespaces** to *not* create a DTS Attribute that attaches to or refers to a Namespace other than the target (source) Namespace. Such objects could include “local” Properties on a subscription Namespace, or Associations that cross Namespace boundaries. If this option is not selected, all source Attributes are created and connected to their referents (which are

assumed to exist exactly as given in the source data, i.e., having the same Namespace and Concept/Term name). Note that this option DOES NOT apply to Defining Concepts or Defining Roles in Ontylog Extension Namespaces.

**Confirm Import Operation:** Click on the **Next** button to run the import.

**Run Import Operation:** Like text imports, XML imports are performed in two steps. During the first step, the import file is parsed and checked for errors. Any errors are displayed in the page's text box. (See the screen shot in the **Run Import Operation** page in the **Importing from Text and Excel Files** section.) An XML structural error is "fatal" in the sense that it will terminate the import process at that point. Fatal errors also include mismatches between the saved Namespace and the target Namespace, e.g., one is a Thesaurus Namespace and the other an Ontylog Extension Namespace.

Attribute referents are validated and these errors are also reported, e.g., if `Ignore objects referencing foreign namespaces` is not checked but a Concept Association has a foreign "to Concept" that does not exist. These errors are not fatal; the error will only prevent creation of the associated Attribute.

Based on the types of errors found, press **Cancel** to terminate the wizard or press **Start** to initiate the actual import step. To avoid unresolved, i.e., "forward", references, the import step is performed in two passes through the XML file. The first pass creates all the Type objects, Terms, and Concepts, and the second creates the associated Attributes. As with text imports, during the second import step the importer writes a log file of the form `DTSImport-yyyy-mm-dd.log` in the execution directory. This file contains a record of all import actions including errors detected and objects created.

During either import step, press the **Stop** button (renamed from **Start**) to terminate processing.

**Operation Complete:** As with text imports, the import log file can be viewed by selecting the check box and clicking **Done**.

## Subset Importing from Text and Excel Files

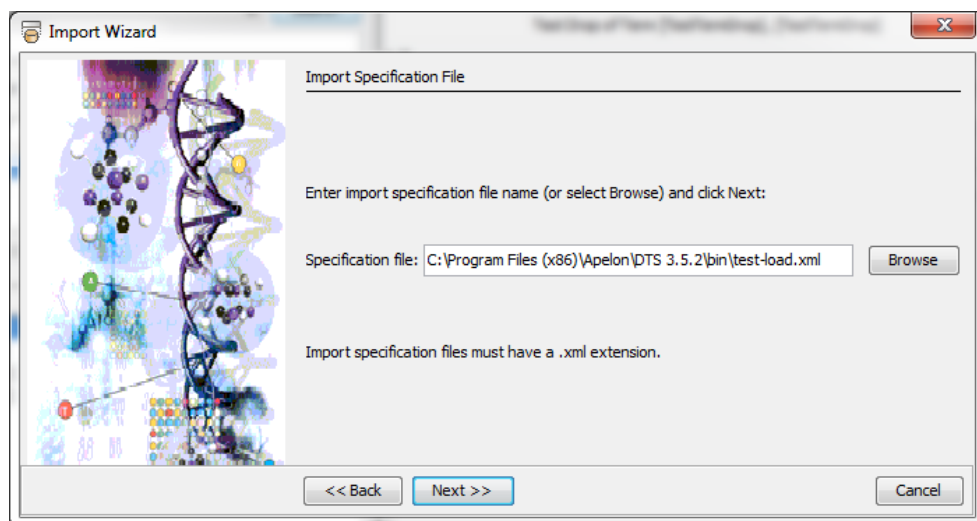
### Text/Excel File Structure

Data to be imported into a Subset Expression from a delimited text or Excel import file must be structured so that each line/row describes a single member Concept for the Subset. This Concept is called the line's Concept Key. The *name* (or other unique attribute) of the Concept Key must be in one of the tab positions. One other tab position is optionally available to specify a set of Concepts related to the Key Concept, for example its Descendants, that should be part of the Subset's membership.

### Text/Excel File Import Pages

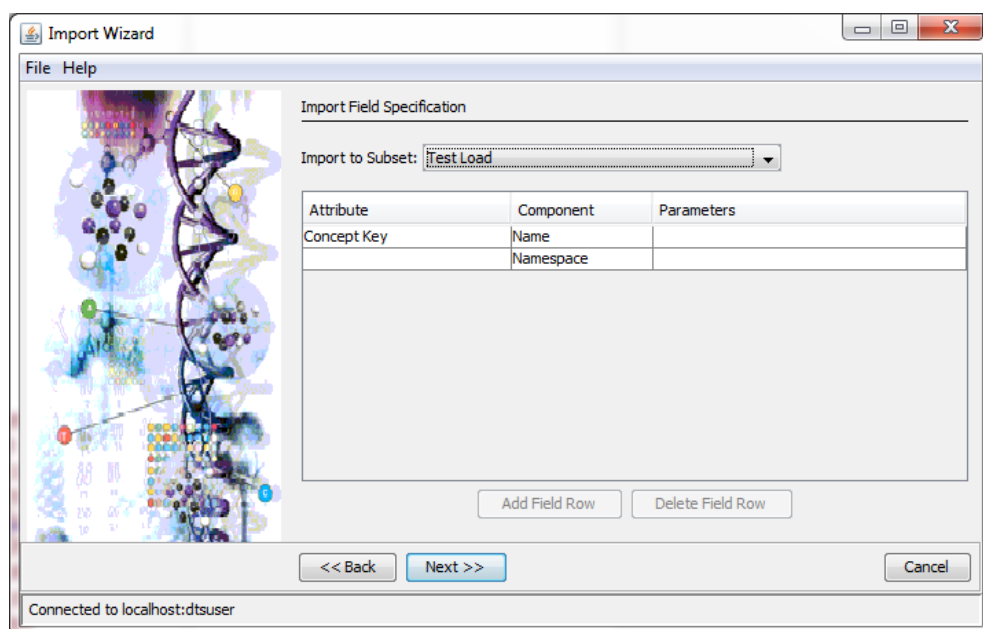
This section describes the wizard pages used in text/Excel file Subset imports.

**Import Specification File:** As with Namespace imports, Subset import from a text or Excel file is driven by an Import Specification. Import Specification files are XML files and can theoretically be created in any standard editor. It is recommended, however, that Specification files be created and edited (or at least validated) through the Import Wizard to avoid program errors. Enter the path and name of the desired specification file in the text field, or click the **Browse** button to search the file hierarchy. The wizard will supply a default extension of “.xml”. Click **Next** to continue.



**Import Field Specification:** A Subset import is defined by specifying (i) the Subset whose Expression will be created, and (ii) the DTS Concepts, with optional modifiers, which will be added to the Expression. These two types of data constitute the Import Specification which will be saved in the Import Specification File.

The Subset is selected from the Combo Box at the top of the **Import Field Specification** page (see screen shot below), while the Attribute parameters (collectively the **Field Specification**) are selected and/or modified in the table at the center of the page. Usually, the values for the parameters come from fields (columns) in the import file. During import, each row of the import file is processed according to the Field Specification defined on this page. Each import row is processed independently.



The first Attribute in any specification is the `Concept Key`. The `Concept Key` Attribute is the “base” Concept for additions to the Subset expression. In addition to the `Concept Key`, a specification can optionally have one `Modifier` Attribute that provides additional Subset membership information. Like Attributes in the `Namespace` Field Specifications, the `Concept Key` and `Modifier` Attributes have associated `Components` and `Parameters` as shown in the screen shot.

The steps to complete the specification for the `Concept Key` and `Modifier` Attributes are described below:

#### Concept Key Attribute Class

The `Concept Key` Attribute class has two components, a `Name` and a `Namespace`. The `Name` component represents the name of the concept/term and the `Namespace` component represents its `Namespace`. To specify the parameters that will define the import values for either of these components, double click in the `Parameters` (third column) cell for the component. This will bring up an associated parameters dialog box. The specification of `Concept Key` parameters is the same as that described earlier for `Namespace` Specification Files, except that a “default” is not available for the `Namespace` value. An input file field position or a literal value is required.

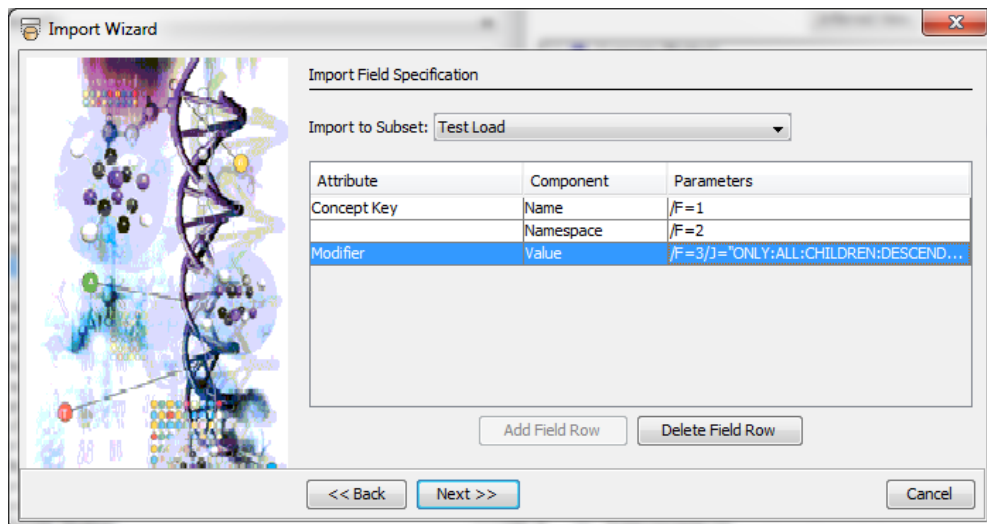
#### Modifier Attribute Class

The `Modifier` Attribute class is used to specify that “relatives” of the Key Concept should be made a part of the Subset expression. Eight relative options are available. The options can be determined by the value of an import file field, or can be fixed for the specification. Double click in the component’s `Parameters` cell to define the modifier’s field location and names. See the **Parameters Dialog Boxes** section below for a description of this dialog. Note: Only one instance of the `Modifier` attribute is permitted in any specification.

Press `Next` when the specification is completed.

The screen shot below shows a completed Subset Import Specification.





**Specification Complete:** Click on the **Finish** button to simply save the specification and exit. Click on the **Next** button to save the specification then go on to run an import.

**Import File:** See the discussion of the **Import File** page in **Namespace Importing from Text and Excel Files** above.

**Confirm Import Operation:** Press **Next** to start the import operation.

**Run Import Operation:** See the discussion of the **Run Import Operation** page in **Namespace Importing from Text and Excel Files** above.

**Operation Complete:** See the discussion of the **Operation Complete** page in **Namespace Importing from Text and Excel Files** above.

## Parameter Dialog Boxes

Double clicking in a `Parameters` table cell opens a **Parameter Dialog** box. The **Parameter** box for the **Subset Key Concept** component is identical to that for **Namespace Imports**. The **Subset Modifier** **Parameter** box also contains similar elements to those presented earlier. The paragraphs below describe the parameters specific to the **Modifier** component. The screen shot below shows an example of the **Modifier Parameter Dialog** box.

The first section is the `Source` section. The `Source` section is used to specify the source for the value of the Modifier component. The value is specified as coming either from a field in an import file row or an explicit literal.

If the Modifier value is to be taken from the input file, select the `Use value from import file Field` radio button and use the “spinner” to select the (one-based) input field number. Then enter the (case-insensitive) name or phrase used in the import file to denote each the eight Concept options. For example, the name for the `Children` option could be “kids”. The default names and their meanings are:

ONLY:	Only the specified Key Concept will be added to the Subset. ONLY is the default behavior if no Modifier is given in the specification.
ALL:	The Key Concept and all of its descendants will be added to the Subset.
CHILDREN:	The Key Concept’s children (direct sub-concepts) will be added to the Subset. The Key Concept is <u>not</u> included.
DESCENDANTS:	All of the Key Concept’s descendants will be added to the Subset. The Key Concept is <u>not</u> included.
EXCLUDE:	Excludes the Key Concept from the Subset.
EXCLUDE_ALL:	Excludes the Key Concept and all of its descendants from the Subset.
EXCLUDE_CHILDREN:	Excludes the Key Concept’s children from the Subset. The Key Concept is not included or excluded.
EXCLUDE_DESCENDANTS:	Excludes all the Key Concept’s descendants from the Subset. The Key Concept is not included or excluded.

If the Modifier data value is to be constant, select the `Use literal Value` radio button. Then select the modifier option from the combo box. This modifier will be applied to all the imported Key Concepts.

The second section is `Filters`. Operation of this section is the same as that for Namespace Parameter Dialog boxes.

## ***Subset Importing from XML Files***

### **XML Schema Files**

The Import Wizard distribution kit includes one XML Schema (.xsd) file that can be used to create and validate XML subset import files:

<code>subset-schema-v4.xsd</code>	This schema is used to load data into a DTS Subset. Subset Name, Description, and Expression for a specific Subset Version are represented along with Subset and Subset Version Property and Qualifier Type definitions and any associated instance values.
-----------------------------------	---

### **XML File Import Pages**

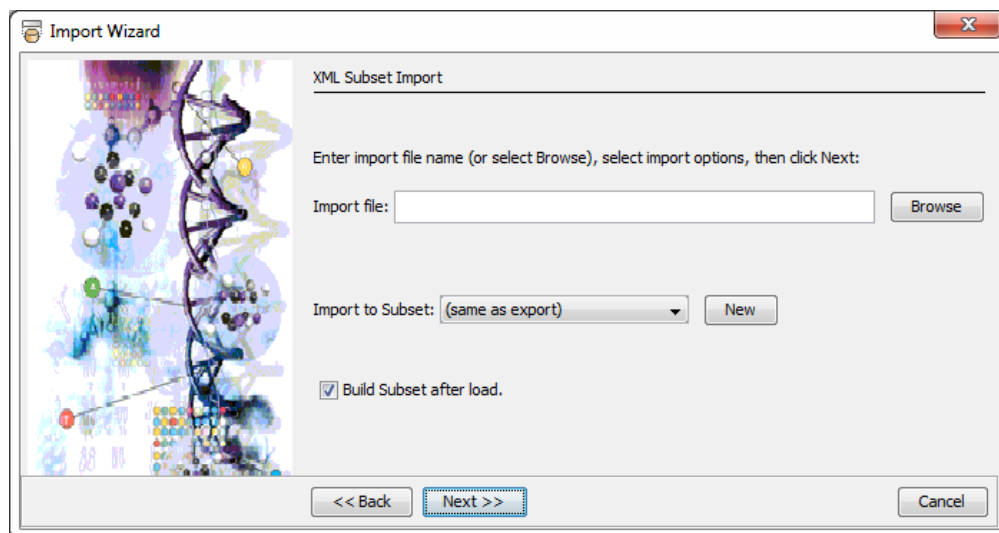
This section describes the wizard pages used in XML subset file imports.

**XML Subset Import:** XML data files can also be used to load definitional and Attribute information into DTS Subsets. These files are typically created using the TQL Editor plug-in module but can also be created independently by following the XML Schema specifications (\*.xsd files) in the distribution kit. See the **TQL Editor User Guide** for further information on creation of export data files, and **XML Schema Files** above for information on these files.

The XML data files contain all the information necessary to populate DTS Subsets. Importing of Subset Properties and Subset Version Properties are supported for all local, writable Subsets. Note that since only the 'Working' Subset Version is writable in a local Subset, saved Version Properties (from any named Version) are always written into the Working Version of the target Subset. Property Type definitions are loaded as well as the Subset Description and Subset Expression.

The XML Subset Import page (see screen shot below) collects all the additional (operational) information needed to execute an import. As with Namespace imports, the import file name is entered into the top text field (or press the Browse button to search the file hierarchy). Next, select the target Subset from the Import to Subset Combo Box. All definitional information and Attributes associated with the source Subset of the data file will be created in the selected target Subset. To create a new Subset, press the New button. (This button is only enabled if the user has the SUBSET\_ADMIN permission.) Enter the new Subset name and associated Authority as described for Namespace imports.

There is one XML load option for Subset loads. Select `Build Subset after load` to have the wizard build the Subset from the Subset Expression if the load completes successfully. If this box is not checked, the Subset will need to be built manually from the DTS Editor.



Confirm Import Operation: Click on the Next button to run the import.

Run Import Operation: Like text imports, XML imports are performed in two steps. During the first step, the import file is parsed and checked for errors. Any errors are displayed in the page's text box. An XML structural error is "fatal" in the sense that it will terminate the import process at that point.

Based on the types of errors found, press Cancel to terminate the wizard or press Start to initiate the actual import step. During import, the importer writes a log file of the form `DTSImport-yyyy-mm-dd.log` in the execution directory. This file contains a record of all import actions including errors detected and objects created.

During either import step, press the Stop button (renamed from Start) to terminate processing.

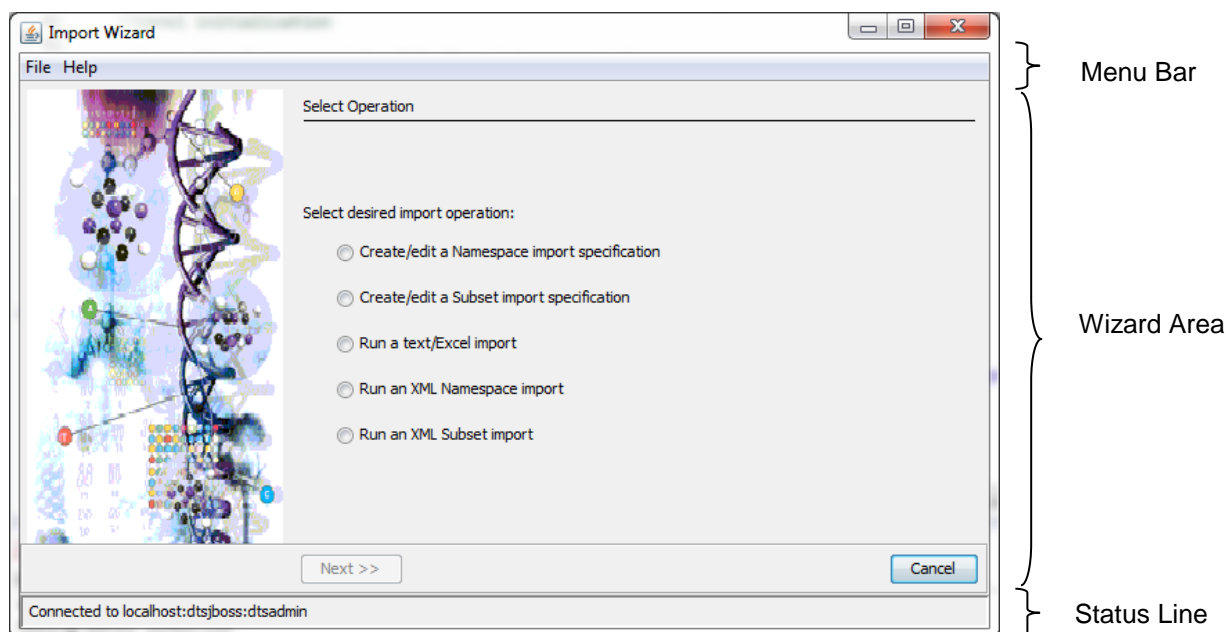
Operation Complete: As with text imports, the import log file can be viewed by selecting the check box and clicking Done.

## Using the DTS Import Wizard Application

The ImportWizard application (see screen shot below) provides a “standalone” presentation of the wizard pages. The application window consists of a central *Wizard Area* which displays the wizard pages, with a *Menu Bar* above and *Status Line* below. Use of the ImportWizard significantly increases import performance by eliminating the overhead associated with refreshing the DTS Editor GUI.

Run the ImportWizard application by executing the ImportWizard.bat file found in *DTSInstall\bin\importwizard*. When the application starts, it opens a Connect to DTS Server dialog. Fill in the appropriate EJB connection parameters, and click Connect. If successful, the host name will be shown in the application’s *Status Line*, and the wizard’s Select Operation page appears in the *Wizard Area*. Wizard page flow and operation is identical to that described in **Using the DTS Import Wizard Plug-In**.

The *Menu Bar* gives access to additional application functions. The File menu includes the Connect, Disconnect, and Exit options. Connect opens the Connect to DTS Server dialog, while Disconnect breaks the connection. Note that disconnecting terminates the active wizard operation. The Help menu includes options to display the **Import Wizard User Guide** (this document) and shows an explanatory About dialog.



## Using *DTSImport*

Data imports can be run in standalone, or “batch” mode, using a batch file such as `DTSImport.bat` found in the `DTSInstall\bin\importwizard` folder. Use of the batch file ensures that the DTS context, e.g. `classpath` variable, are set up appropriately. Importing via a batch file avoids DTS Editor GUI overhead which significantly reduces load time.

Import processing using `DTSImport.bat` (or similar script) is the same as described for the Import Wizard with the following exceptions:

1. Imports have only one “pass”. The log file is written, but errors are reported immediately and may terminate the import depending on the type of error.
2. Import options (see below) are similar to those offered in the Import Wizard. To create a new Namespace or Subset in XML imports, use the “/N” option. If this option is included, the “/A” option must also be present to give the importer the name of the associated Authority. (An Authority is required to create Namespaces and Subsets in DTS.)

As shown in `DTSImport.bat`, importing is performed by the `DTSImport` Java class. Parameters for the import such as connection values and file names are supported via position-independent key word/value pairs. If no parameters are provided on the `DTSImport` command, the program will prompt for values from the console. The available parameters are:

<code>-host:hostname</code>	Required. The name of the DTS host (server) system.
<code>-port:portnumber</code>	Required. The port number of the DTS host (server) system.
<code>-instance:instance</code>	Required. The name of the DTS Server instance on the app server.
<code>-user:username</code>	Required. The application server username.
<code>-psw:password</code>	Required. The application server user password.
<code>-infile:filename</code>	Required. The name of the import file.
<code>-specfile:filename</code>	The name of the specification file if not an XML import. Either <code>specfile</code> or <code>space</code> must be present.
<code>-space:namespace</code>	The name of the load namespace (or subset) if an XML load. Either <code>specfile</code> or <code>space</code> must be present.
<code>-options:optionstring</code>	Optional. The set of import options (see details below).
<code>-log:filename</code>	Optional. The name of the log file to be used. If not present, a rolling daily log file is used.

The options string is slash-delimited parameter string of the form:

```
[/A='authority' ][/C] [/D='|' ] [/E='encoding' ] [/F] [/H] [/L] [/N] [/S='Sheet1' ]
```

where:

A specifies the name of an existing DTS Authority (XML imports only, N parameter required).

B means build the subset after import (XML subset imports only).

C means use the Attribute Code and Id from the import file (namespace XML imports only). If not present, the DTS Code and Id generator will be used for imported elements.

D specifies the single character delimiter for the import file (text import only).

E specifies the encoding for the (text) import file; "8859\_1" (the default) or "UTF8".

F means do not create attributes to foreign namespaces, i.e., namespaces other than the import namespace (namespace XML imports only).

H means a header line is included in the import data file(s) (text/Excel import only, first import line will be ignored).

L means that the import file specified in the command line (text only) contains a *list* of other import data files. Each of these data files will be read and processed by DTSTImport.

N means create the target namespace/subset if it does not already exist (XML imports only, requires the A parameter).

S specifies the name of the sheet (in an Excel import file) to be used.

An example DTSTImport command line is shown below:

```
DTSTImport -user:manager -psw:password -host:localhost -port:4447  
-infile:demo-load.xml -specfile:demo-load.txt -log:"demo log.log"
```

Note the use of quotes around the log file parameter value since the value contains a space. Quote marks can be used around any value but are required if the value contains a space.

## ***Revision History***

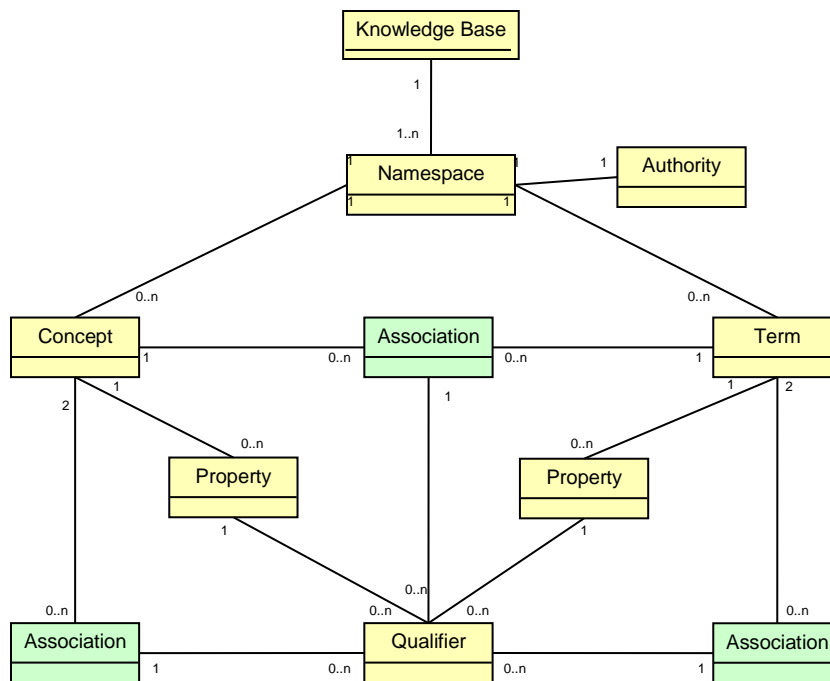
Version 1.0	Initial release. Concepts cannot be added.
Version 2.0	Major rewrite. Limitations: Does not use WriteManager to update other Editor panels.
Version 2.1	Required parameter added. Quote parameter processing updated. Bug fixes.
Version 2.2	Rebuild with Eclipse. Renamed jars.
Version 2.3	Updated for DTS V3.4. Added Designate and Preferred parameters and Concept Code and Concept Id Attributes. XML imports. Inverse Associations supported.
Version 2.4	Icon in DTS Icon Bar. Updated documentation with introduction, bug fixes.
Version 2.5	Extension Namespace attributes (Defining Concept, Defining Role, Primitive/Defined) supported. Default Namespace added.
Version 2.6	Importing of Terms and Term Attributes (Properties, Associations, and Qualifiers) supported. Importing of Ontylog Extension Attributes (Primitive, Defining Concept, and Defining Role) supported. Encoded parameter added.
Version 3.0	Import XML Schemas included in distribution. Run time specification (and creation) of Property, Synonym, and Association Types. Header parameter added. Major revision of User Guide including addition of Data Modeling Appendix.
Version 3.1	XML import two pass modification to reduce memory usage for concept loads. Much larger XML files can now be imported. Improved XML error handling using new xmldigester.jar package.
Version 3.2	Encoded parameter is available for all Concept-valued Attributes. Multiple parameter added to support value lists. Text file character encoding parameter added to Import File wizard page and as DTSImport command line option. Term Association import bug fixed. JDBC connection supported in DTSImport.
Version 3.3	ImportWizard application added. Encoded parameter extended to Term-valued Attributes (Term Key, Term Association and Synonym) to facilitate Term disambiguation. Code and Id options added to Encoded Attributes. Logic additions made to Add parameter for Term Key Attribute. Update parameter added. XML loader handles identically-named Terms in Synonyms (requires TQL V3.0 export). Corrected bug on setting defining concepts in extension namespaces.
Version 3.4	Specification file extension defaults to “xml”. Integer and Number validation limits. RegEx validation and filtering. Improved Parameter error reporting on specification load and Parameter edit. Input string tokenization bug (delimiter imbedded in quoted strings) corrected. About dialog.



Version 3.5	Excel import files supported. Concept/Term Key Delete option. Namespace Name Parameter Panel uses drop-down for Namespace names. Support for loading of lists of import data files when running in batch mode ("L" option). Filter/Validator Tester. Support for Defining Role Groups in Extension Namespaces added and Extension import bugs corrected. Error reporting delay in Import Wizard UI corrected.
Version 3.6	Action Attribute class added. Blank (all spaces) Property values supported (with updated <code>xmldigester.jar</code> ). Encoded option and Validation parameters supported on Concept/Term Key. Importing of Defining Roles using Base Namespace Role Types supported. Primitive and Term/Synonym load bugs corrected. Improved error reporting.
Version 3.7	Name Attribute class added for renaming and Concept/Term creation. Add and Encoded are simultaneously permitted as Key Concept/Key Term Parameters. The DTSImport log file is written with a "rolling daily" protocol: a new log file is created every day and all log messages from all imports that day go to the log.
Version 4.0	Updated for DTS 4.0. New batch parameter syntax. Subset imports. Batch parameter available to override default log file. Additional progress/status messages in Run page. Attribute delete functionality. Attribute update and delete added for Qualifiers.
Version 4.1	Term load bug fixes. Status Attribute class and Preferred component of Synonym class added. Specify parameter on Key Concept and Key Term Attributes classes removed. New Subset XML import. XSDs redefined and extended for new features (formats compatible with TQL V4.0). XML import files validated before loading. Namespace and Subset creation supported in UI. Namespace, Subset and Version Properties supported. Over-size Property values truncated before saving. Performance improvements. Status window reports progress on large text/Excel file imports.
Version 4.2	Bug fixes.

## Appendix A – Modeling Data in DTS

The data modeler's first task is to translate the desired (or existing) data representation into a DTS representation using the elements of the DTS Data Model. Figure A1 gives a simplified view of the DTS Data Model. The basic elements of the model are described below. (In the rest of this Appendix, **bold** items represent DTS elements, *italicized* items represent instances of these elements, and fixed width font represents literal elements, e.g., menu items or prompts, in the DTS Editor.)



**Figure A1: Simplified DTS Data Model**

**Namespaces** – A **Namespace** is the unit of management for knowledgebase access, maintenance, and reporting purposes. A **Namespace** can contain any information but typically consists of data in a specific subject area, such as a single formal terminology. Associated with each **Namespace** is an **Authority** that represents the organization responsible for the information. All other elements of the DTS Data Model described below are “owned by” (created in) a specific **Namespace**.

[This Appendix only addresses data modeling in local **Thesaurus Namespaces**. **Thesaurus Namespaces** give modelers full and independent control over the creation of DTS objects. **Ontylog** objects, created in Apelon's Terminology Development Environment tool, are not discussed as they cannot be created within DTS except as extensions to subscription **Ontylog Namespaces**. See the DTS documentation for further information on **Ontylog** and **Ontylog Extension Namespaces**.]

**Concepts** – **Concepts** are the foundational element (or object) in DTS. Historically, **Concepts** have been defined as a unique unit of thought or meaning, but in actuality **Concepts** can refer to whatever the modeler desires. DTS provides comprehensive capabilities for manipulating, locating, and updating **Concepts** based on their names and other attributes (see below).

**Terms** – **Terms** represent text strings and information about them. A **Term** is generally (but not necessarily) connected to a **Concept** via a **Synonym** relationship (see below). A **Term** can also be associated with another **Term**, but this is an infrequently used aspect of the Data Model.

**Properties** – A **Property** is a name/value pair that is attached to a **Concept** (or **Term**). The name is a **DTSPROPERTYType**, a modeler-defined **Property** “class”. An instance of this class, along with a string value, is associated with the **Concept** (or **Term**). For example, the SNOMED CT **Concept** *Myocardial infarction (disorder)* has an instance of the **DTSPROPERTYType** *UMLS CUI* with value *C0155626*. As with all DTS elements, a **DTSPROPERTYType** is owned by a **Namespace**, but an instance of the type can be associated with a **Concept** in any **Namespace**. Thus a local **DTSPROPERTYType**, *ER Code*, from the *Mass General Hospital Namespace* could be associated with SNOMED’s *Myocardial infarction (disorder)* **Concept** to provide a “local code” lookup capability.

**Associations** – An **Association** is a link (or relationship) between two other DTS Data Model objects. Specifically, DTS defines **Concept Associations** which connect one **Concept** to another (both within and between **Namespaces**), **Synonym Associations** that connect a **Concept** to a **Term**, and **Term Associations** that connect **Terms**. The latter is infrequently used in most data models. As with **Properties**, an **Association** is a name/value pair where the name refers to a user-defined **AssociationType**, and the value is another **Concept** (or **Term**). A common use of **Associations** is to create hierarchies, typically using a **Concept AssociationType** named *Parent Of*.

**Synonyms** – A **Synonym** is a shorthand notation for an **Association** that connects a **Concept** to a (synonymous) **Term**. One **Synonym** of each **Concept** can be marked as the “preferred” synonym for the **Concept**.

**Qualifiers** – A **Qualifier** is a name/value pair that can provide additional, related, information to a **Property** or **Association**. The name of the **Qualifier** is a **QualifierType** and the value is a string. Typical uses of **Qualifiers** include notes on a **Property**, e.g., *Reviewed by: Dr. Smith*, or further characterizing a **Concept Association**, e.g. *Mapping Quality: Narrow to Broad*.

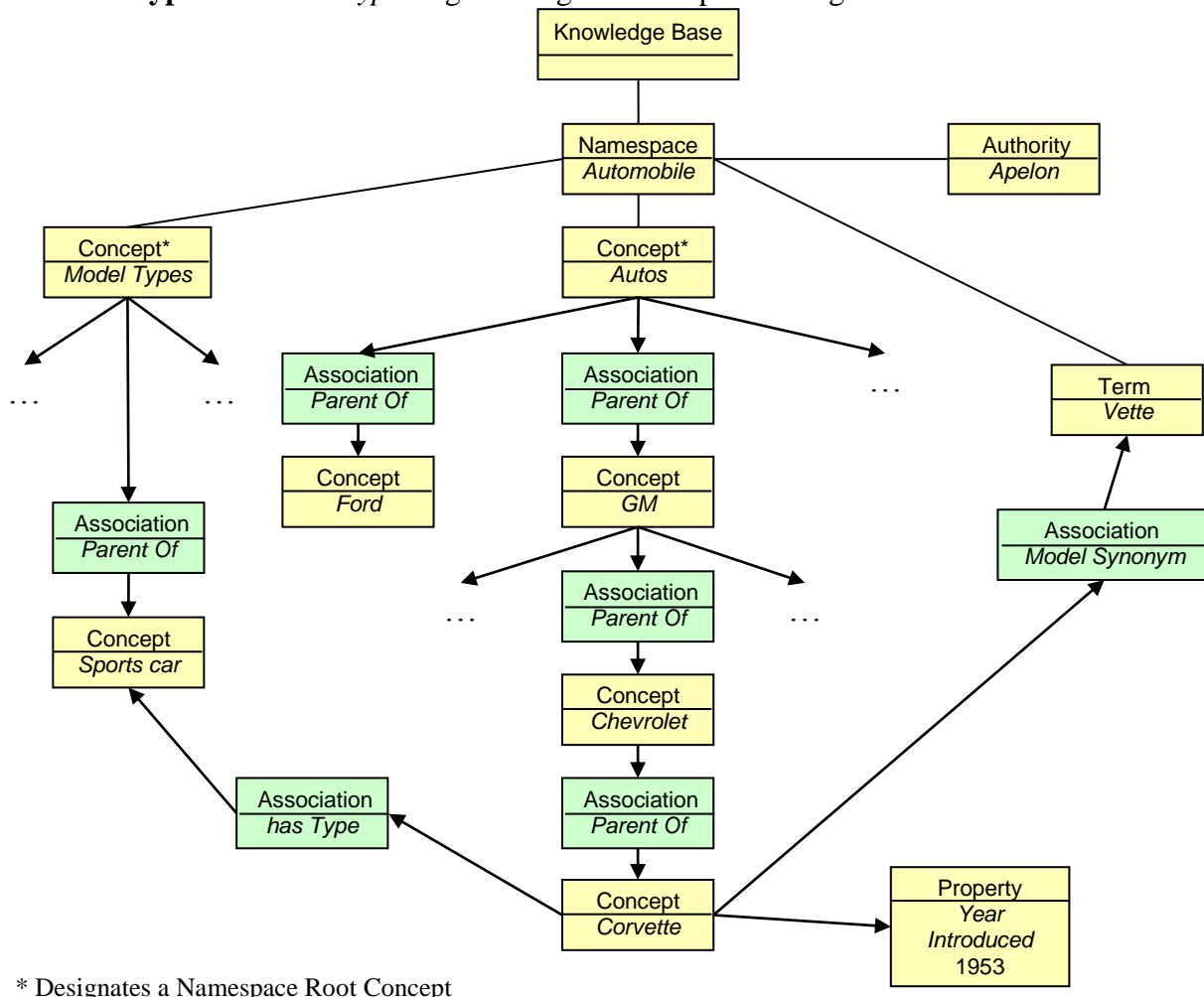
**Namespaces, Concepts, Terms, DTSPROPERTYTypes, AssociationTypes, and QualifierTypes** are all **DTS Objects**. Every **DTS Object** has three “fixed” attributes: a **Name** (a string), a **Code** (also a string), and an **ID** (an integer). The only requirement on these attributes is that they each must be unique within their enclosing **Namespace** (or knowledgebase in the case of a **Namespace**). The DTS modeler is free to assign whatever values they want to these attributes, including externally visible (user) values. It is recommended, however, that user values be placed in the variable attributes, such as **Properties**, and not in **Code** and **ID**. **Code** and **ID** are generally not explicitly specified; values are automatically created by the DTS Server. If specific values are required, these can be entered in the DTS Editor UI or by using the associated API constructor methods.

The modeler’s first task is to determine how best to represent her data within the available DTS Data Model elements. What data will be **Concepts**? What **Properties** (string values) will these Concepts have? How should the **Concepts** be connected by **Concept Associations**? What **Property** and **Association Qualifiers**, if any, should be used?

As a simple example, we will consider a data base used to classify automobiles. We want to be able to understand the manufacturers, makes, and models of the vehicles, as well as certain characteristics of the models themselves such as year introduced. We’ll call our **Namespace** *Automobiles*, and have a root (or starting) **Concept** called *Autos*. For the purposes of this example, we’ll build a specialization hierarchy based on manufacturers, so *Autos* will point to *Fords*, *GMs*, etc. We’ll use the standard *Parent Of*

**Concept AssociationType** to build this hierarchy because DTS recognizes this **Concept AssociationType** as the default for hierarchy tree display. [Ontology purists will argue that the intermediate nodes should be named *Ford Autos* to make clear the specialization, but we'll use the shorter form nonetheless.] We can then continue the hierarchy (via *Parent Of* connections) from *GMs*, to *Chevrollets*, to *Corvettes*. Let's stop at this level of detail and see what information we'd now like to associate with a Corvette model. We'll define a **DTSPROPERTYType** of *Year Introduced*. The value will be the year of introduction. Corvettes are sometimes called "Vettes" so we need a Synonym to assist in lookups. We'll use a **Synonym AssociationType** called *Model Synonym*.

Finally, we'd like to capture the type of auto, something like sports car, SUV, sedan, etc. Now we could just add another **DTSPROPERTYType**, but this would allow any string value. A better methodology is to create what is often called a Reference Code Set. A Reference Code Set is nothing more than a set of names (**Concepts**) which can be used as a controlled source for values. In DTS, we can create a Reference Code Set of model types. We'll start from a **Concept** called *Model Types*, and connect this **Concept** to a set of other **Concepts** (using *Parent of*) whose names are the types we need. We'll make *Model Types* another root **Concept** of the Namespace. The type name **Concepts** will all be children of *Model Types* for now, i.e., just be a flat list, but note that the types could have a more complex hierarchy (a convertible could be a sports car or a full-size car for example) which would permit more sophisticated navigation and aggregation of the models. We'll connect our auto model **Concepts** to their types using a **Concept AssociationType** called *has Type*. Figure A2 gives a simplified diagram of our model.



**Figure A2: Simplified DTS Automobile Data Model**