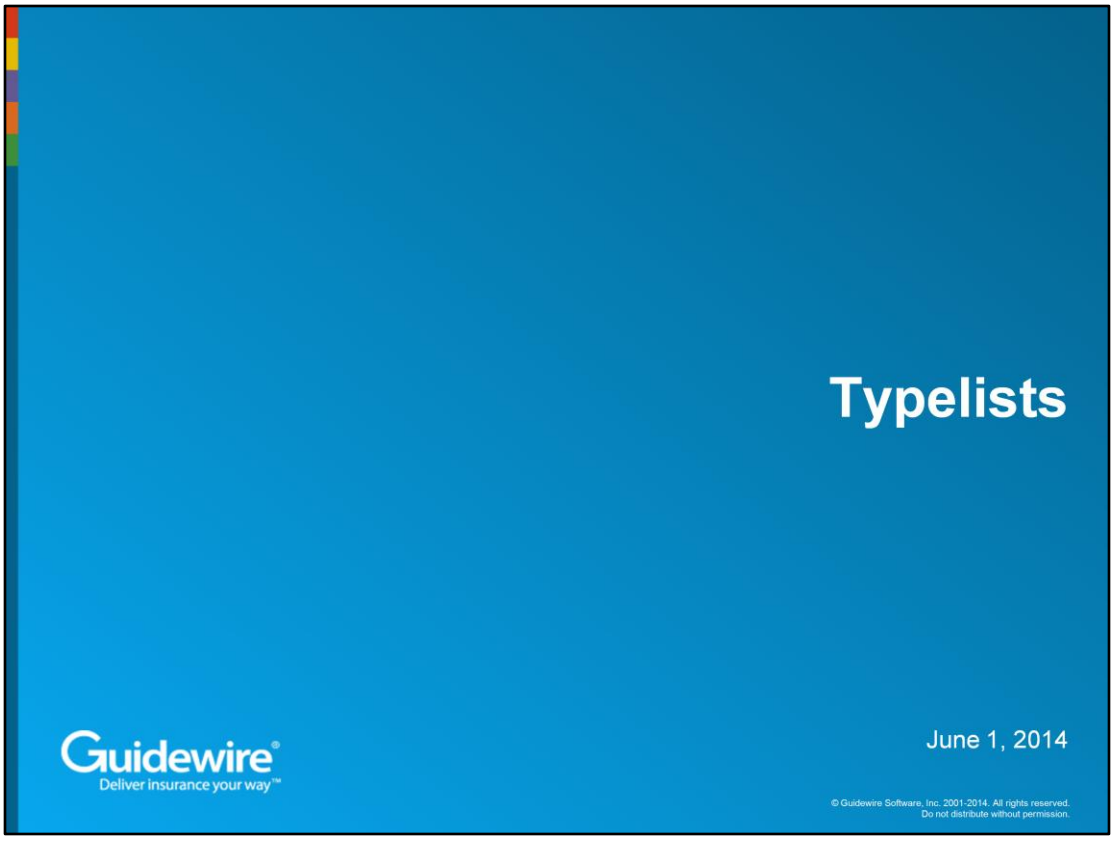


Guidewire training materials contain Guidewire proprietary information that is subject to confidentiality and non-disclosure agreements. You agree to use the information in this manual solely for the purpose of training to implement Guidewire software solutions. You also agree not to disclose the information in this manual to third parties or copy this manual without prior written consent from Guidewire. Guidewire training may be given only by Guidewire employees or certified Guidewire partners under the appropriate agreement with Guidewire.

Lesson objectives

- By the end of this lesson, you should be able to:
  - Describe the functionality of a typelist
  - Create a typelist and typelist extension
  - Create a typekey field
  - Create a typekey field that references a typelist filter

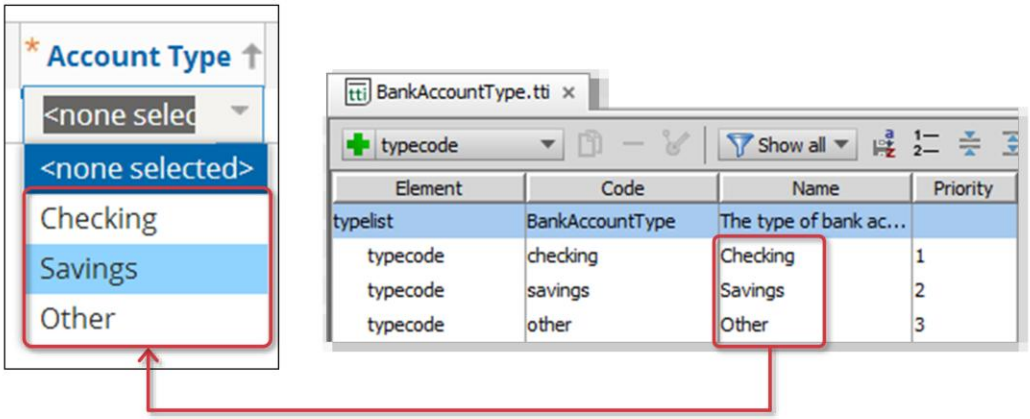
This lesson uses the notes section for additional explanation and information.
To view the notes in PowerPoint, select View → Normal or View → Notes Page.
When printing notes, select Note Pages and Print hidden slides.

Lesson outline

- Typelist basics
- Create a typelist
- Create a typelist extension
- Defining typekey fields

What is a typelist?

- A typelist is a list of defined values
- Defined values often used to constrain user input
- When rendered in UI, typically appears as dropdown list



BankAccountType typelist defines values in Account Type cell

Locations of typelist files

…\Metadata\Typelist

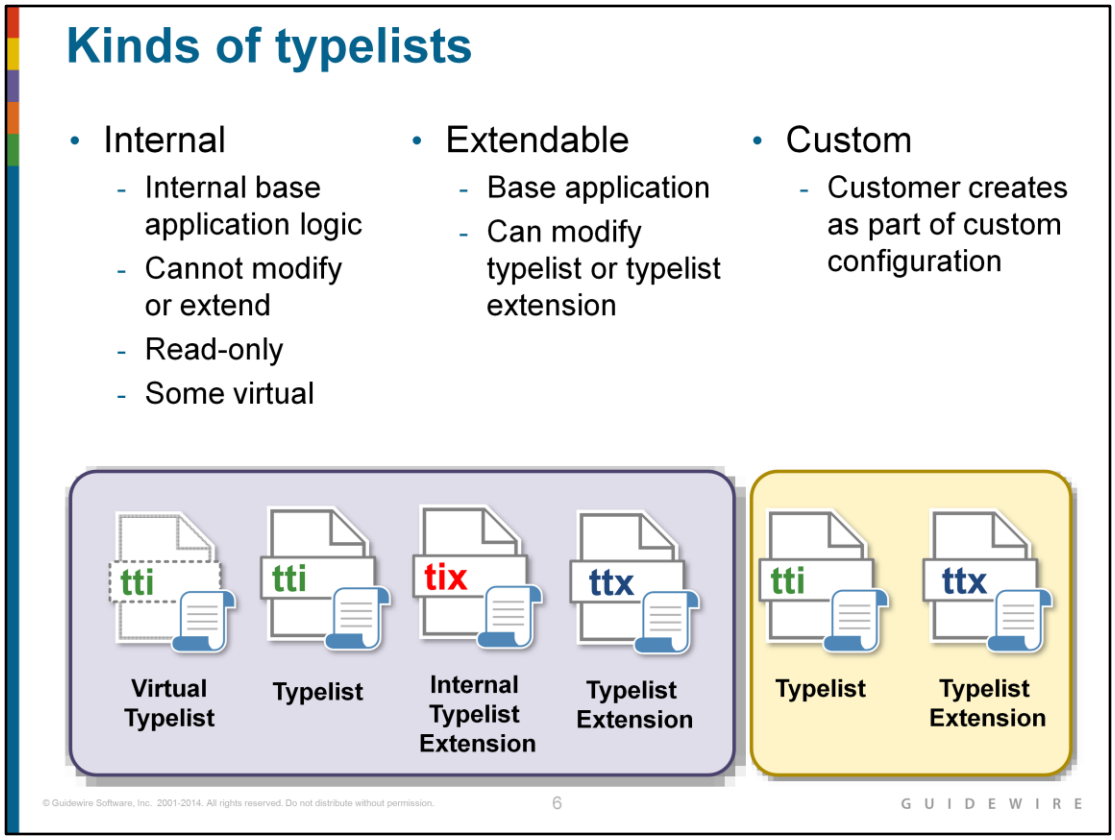
Typelist | **Internal Typelist Extension**

- Most platform and base application typelists
- All files are read-only
- Some files marked as final

…\Extensions\Typelist

Typelist | **Typelist Extension**

- Some base application typelists and typelist extensions
- Customer custom typelists and typelist extensions
- All files are editable



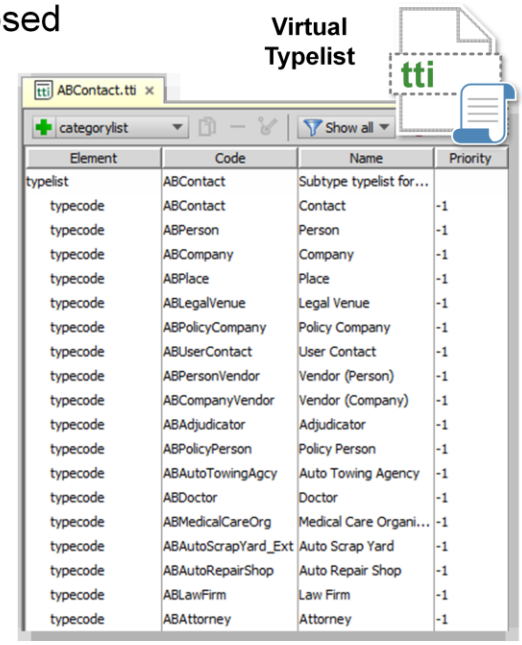
In some cases, an internal typelist does not have a physical file that is exposed in Guidewire Studio. One type of virtual typelist is a typelist of subtypes for a given supertype. In Guidewire Studio, using Class Name Search (CTRL+N) , you can find and view a virtual typelists in the Typelist Editor. However, there is no file path and no XML file view for virtual typelists. One example is the ABContact typelist which is a typelist of all ABContact subtypes.

Typelists with the final="true" attribute defined in the <typelist /> element cannot be extended. An internal typelist extension (TIX) file can only be found in the …\modules\configuration\config\metadata\typelist\ folder.

You can always determine to base configuration for a Guidewire application by looking at the <installDirectory>\modules\base.zip file.

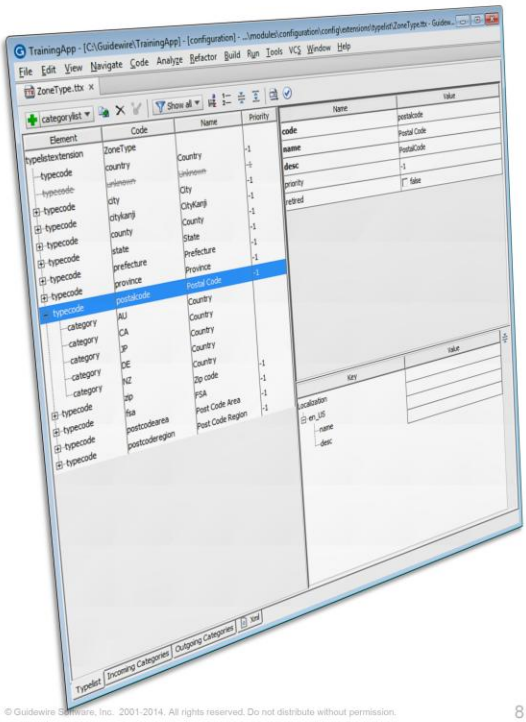
Internally managed (virtual) typelists

- Some typelists are not exposed
- No physical file exposed
  - Typelist Editor file tab shows yellow for virtual status
  - Typelist Editor XML tab shows empty file
  - Studio navigation bar is blank
- Example: Subtype typelist
  - Shows all subtypes a specific subtyped parent entity
  - New subtypes entities automatically added



Element	Code	Name	Priority
typelist	ABContact	Subtype typelist for...	
typecode	ABContact	Contact	-1
typecode	ABPerson	Person	-1
typecode	ABCompany	Company	-1
typecode	ABPlace	Place	-1
typecode	ABLegalVenue	Legal Venue	-1
typecode	ABPolicyCompany	Policy Company	-1
typecode	ABUserContact	User Contact	-1
typecode	ABPersonVendor	Vendor (Person)	-1
typecode	ABCompanyVendor	Vendor (Company)	-1
typecode	ABAdjudicator	Adjudicator	-1
typecode	ABPolicyPerson	Policy Person	-1
typecode	ABAutoTowingAgcy	Auto Towing Agency	-1
typecode	ABDoctor	Doctor	-1
typecode	ABMedicalCareOrg	Medical Care Organi...	-1
typecode	ABAutoScrapYard_Ext	Auto Scrap Yard	-1
typecode	ABAutoRepairShop	Auto Repair Shop	-1
typecode	ABLawFirm	Law Firm	-1
typecode	ABAttorney	Attorney	-1

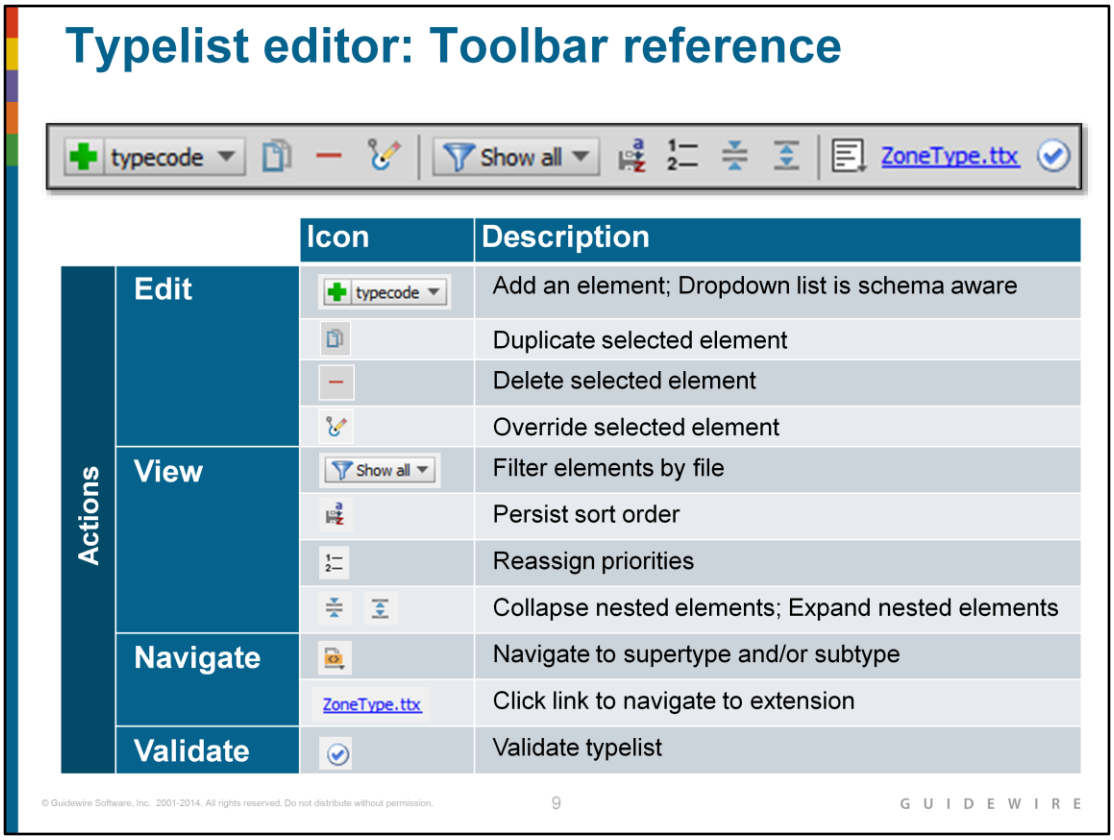
Typelist editor



- View, define and edit a typelist, typelist extension, or internal typelist extension
  - **`/extensions/typelist/`**
    - TTI and TTX files
  - **`/metadata/typelist/`**
    - TTI and TIX files
- Consists of
  - Editor toolbar
  - Element tree pane
  - Attribute pane
  - Localization pane
- Metadata Editor is base editor

With the Typelist editor, you define a typelist and any associated typecodes, typefilters, and categorylists. With the Typelist editor, you can view all (but not edit all) typelists: extendable, internal, internally exposed (typelist of subtypes, for example), and custom typelists.

You can define and edit an typelist and typelist extension located in the …\configuration\config\extensions\typelist folder. The Typelist editor also allows you to view an typelist or internal typelist extension located in the …\configuration\config\metadata\typelist folder. Files in the metadata folder are read-only.

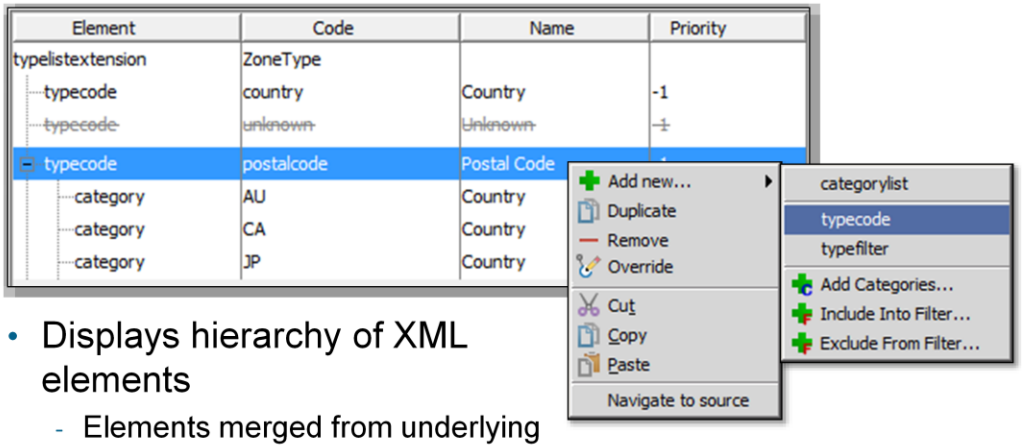


When you define and edit a typelist or typelist extension located in the /configuration/config/extensions/typelist folder, the Typelist editor toolbar is fully enabled. Typelists and internal typelist extensions in the /configuration/config/metadata/entity folder are read-only and all Edit actions are disabled.

The Typelist editor toolbar is also context sensitive based on the kind of typelist (internal, internally exposed, extendable, or custom). For internal typelists, there are no edit actions in the toolbar.

The Typelist editor toolbar is also context sensitive based on the bottom tab selection. For both the Incoming Categories and Outgoing Categories bottom tabs, the toolbar consists only of the Add Category/Typecode, Remove, Filter, Collapse, Expand, and Validate commands.

Typelist editor: Element tree pane

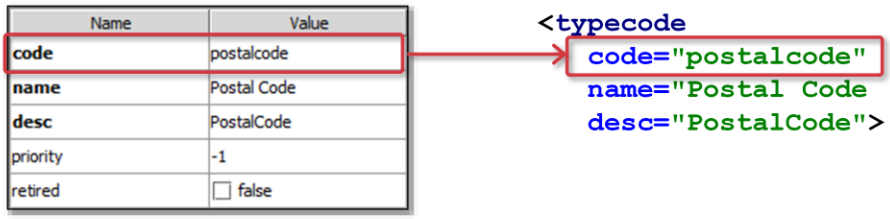


- Displays hierarchy of XML elements
  - Elements merged from underlying base typelists are read-only
  - Sortable columns
- Context menu is schema aware
  - Add new elements as siblings and children
  - Cut, Copy, and Paste elements

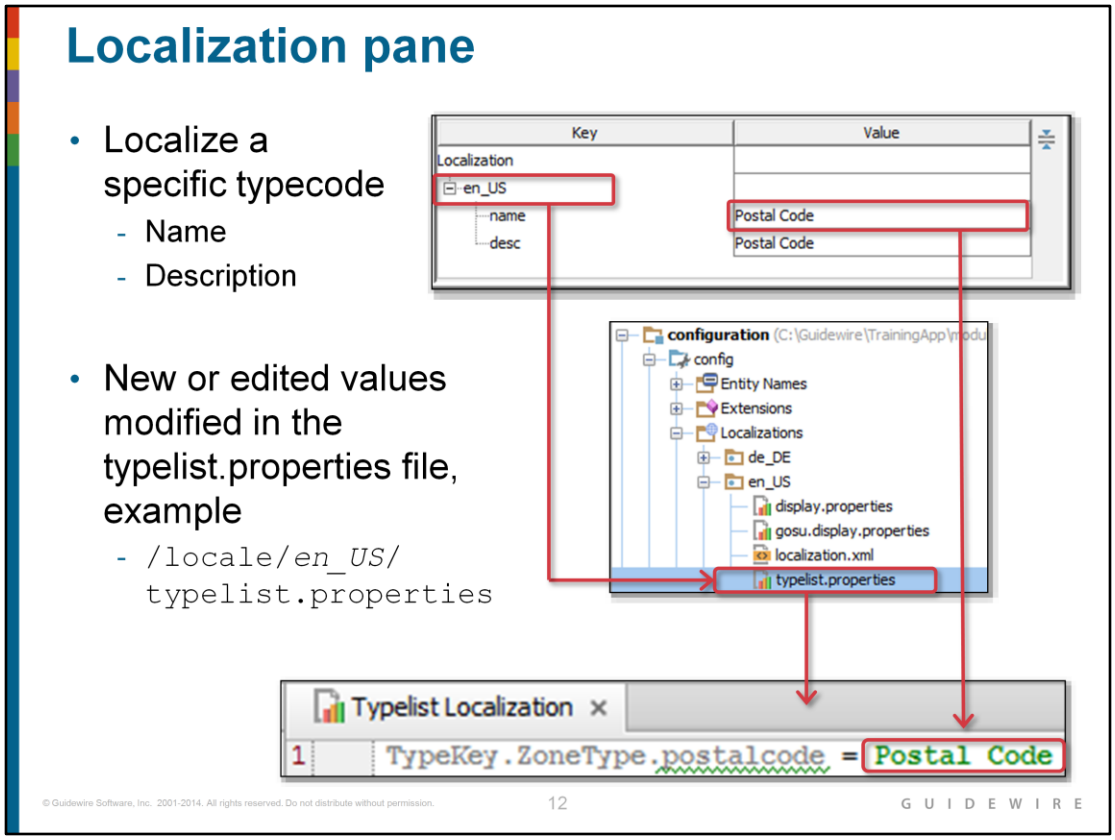
View actions in the Typelist editor toolbar influence the display of the Element tree pane. Bottom tab selection also changes the columns of the Element tree pane for the Typelist editor. For instance, Incoming Categories and Outgoing Categories changes the columns to Category/Typecode and Typelist. By default, the Element tree pane displays the hierarchy of nested elements, including any inherited elements from base typelists. You can use the toolbar to change how the editor display the various elements: Show all, This file only, Typecodes only, and Typefilters only. The toolbar and the context menu for the element tree pane is schema aware.

Typelist editor: Attribute pane

- For selected element, define attributes
  - Name is the attribute
  - Value is the attribute value

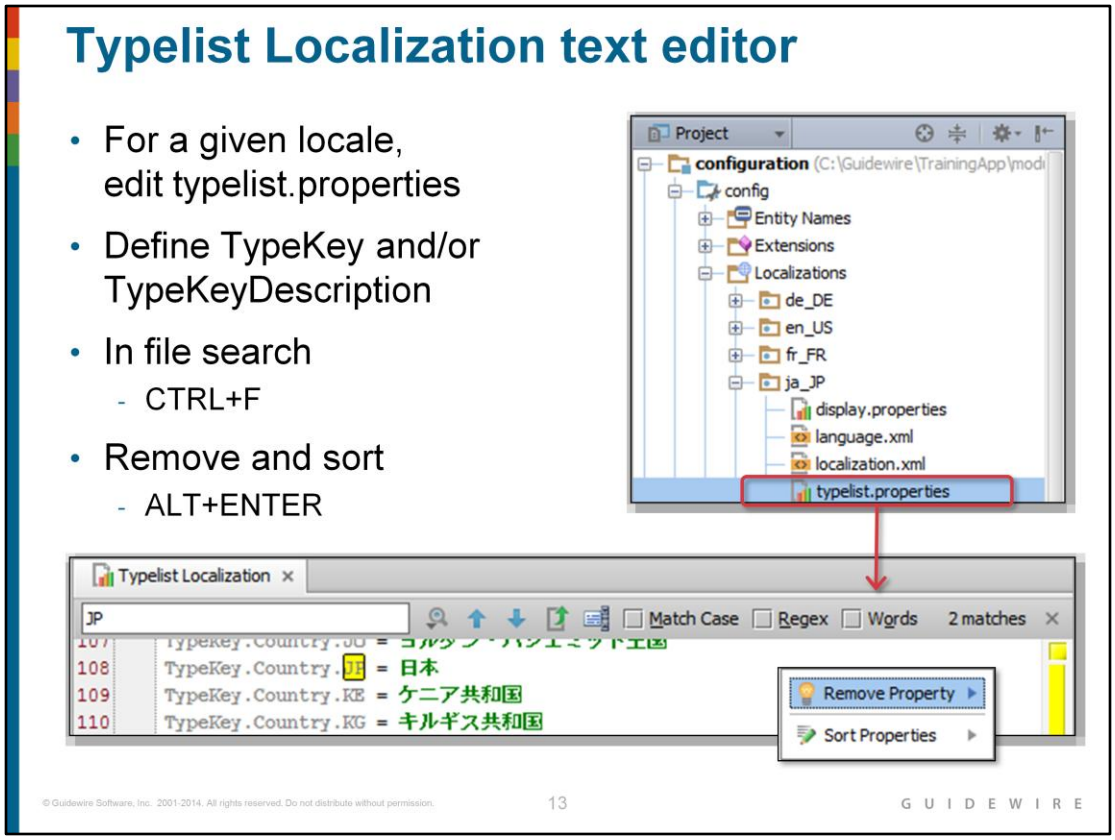


- Attribute styling
  - Bold for required and black for editable
  - Grayed-out for non-editable
    - Overridden, Inherited, Internal and Default



Use the Localization pane to select a specific locale for a given typecode. Then, create or edit a translation string for both the typecode name and description. The name and description that you specify is created in the typelists.properties file for the locale. Guidewire stores typelist typekey localizations in typelist.properties files, one file for each locale. Guidewire Studio stores typelist typekey localizations using the following format for name and description respectively:
TypeKey.typelist.typekey = localized_name
TypeKeyDescription.typelist.typekey = localized_description

You can edit typecode name and descriptions in the Typelist editor Localization pane or in the Typelist Localization editor.



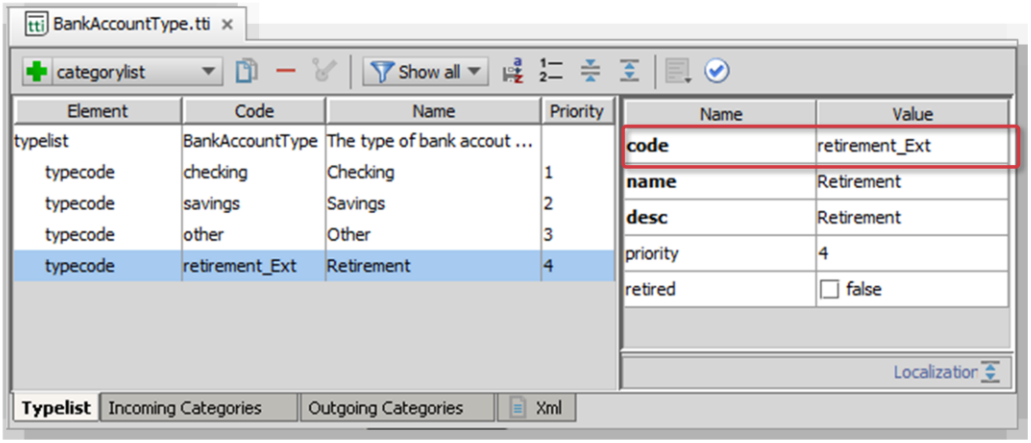
Guidewire stores typelist typekey localizations in typelist.properties files, one file for each locale. Guidewire Studio stores typelist typekey localizations using the following format for name and description respectively:

- TypeKey.typelist.typekey = localized_name
- TypeKeyDescription.typelist.typekey = localized_description

You can edit typecode name and descriptions in the Typelist editor Localization pane or in the Typelist Localization editor.

There are several ways to view localized typekey properties:
- typekey.Code = String that represents the typecode
- typekey.DisplayName = Localized language version of the entity name
- typekey.UnlocalizedName = Name listed in the data model

Typecodes for base application typelists



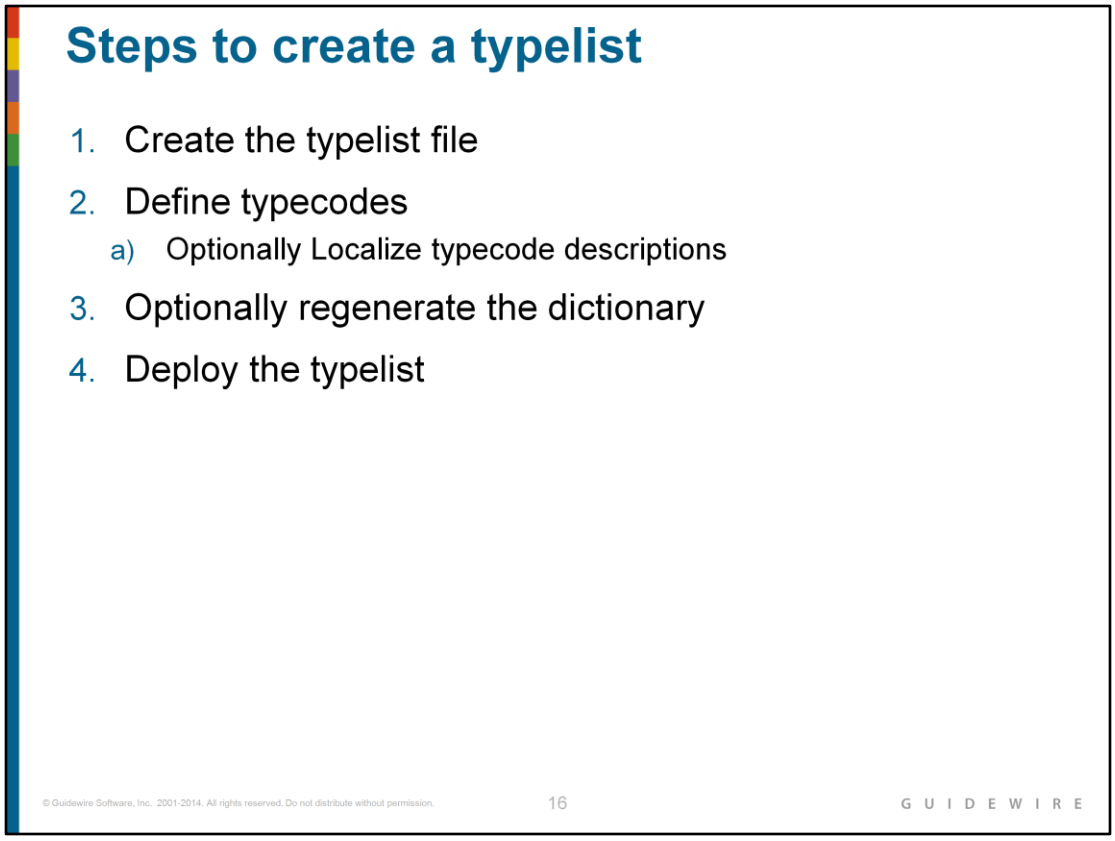
- When extending base application typelists or typelist extensions in the …**\Extensions\Typelist** folder, create new typecodes with codes that end in _Ext

The BankAccountType.tti is an application typelist found in the …\modules\configuration\config\extensions\typelist\ folder.

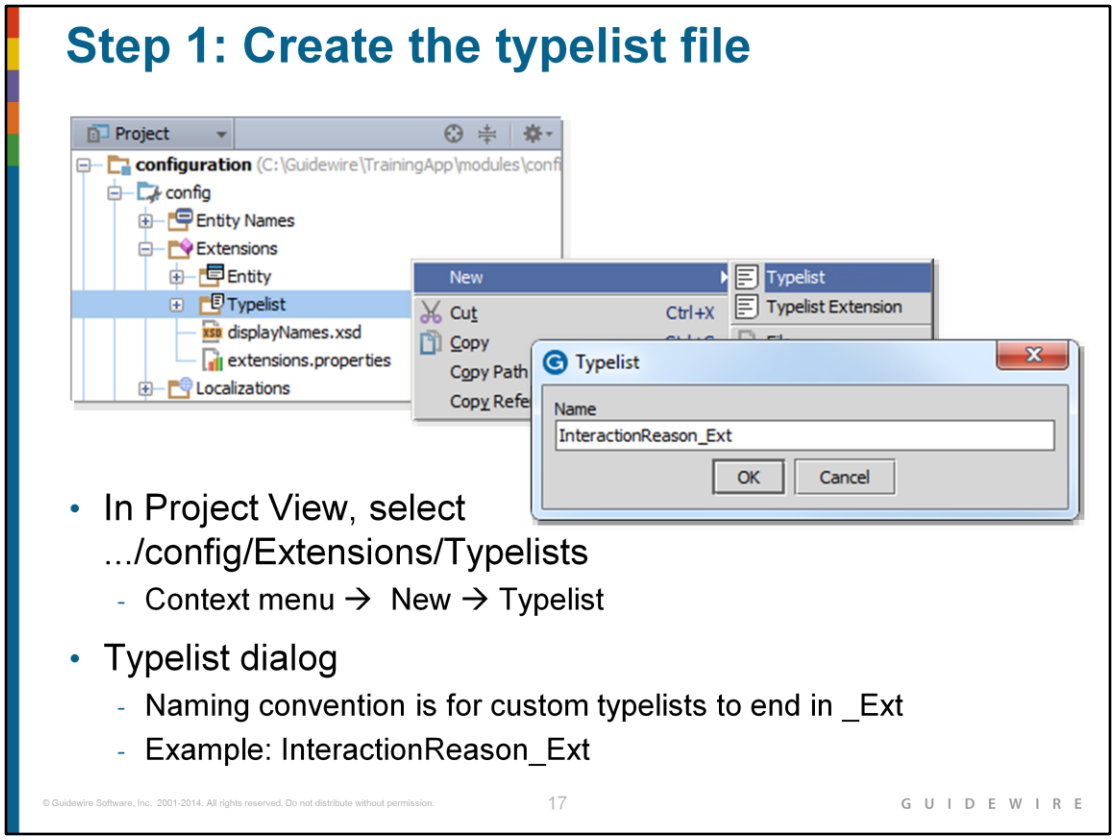
For TTI and TTX files that are extendable (not marked as final) and editable in the \extensions\typelist\ folder that are part of the base application, use the _Ext naming convention by adding _Ext to the name of the typecode.

Lesson outline

- Typelist basics
- Create a typelist
- Create a typelist extension
- Defining typekey fields



To create an typelist, in Project view, navigate to the /configuration/config/Extensions/Typelist folder. Open the folder context menu. Select Typelist. In the Typelist dialog, name the typelist. Click OK.



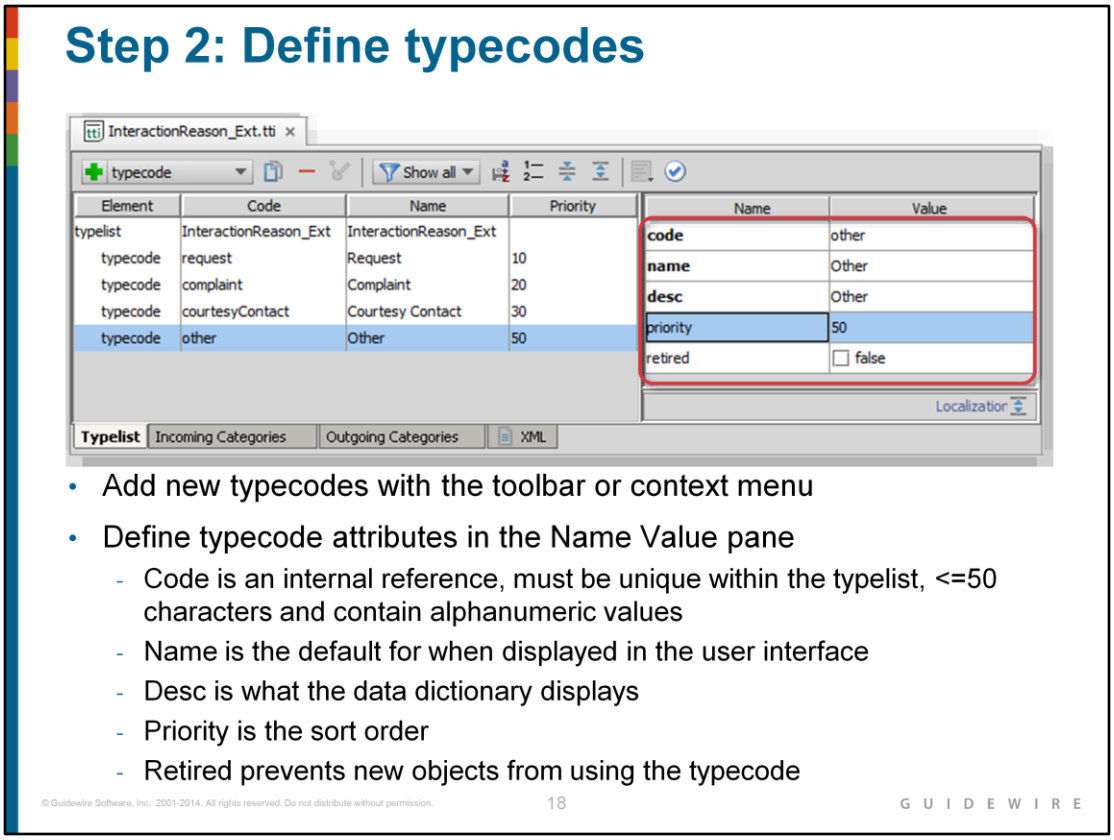
The maximum length for a typelist name is 25 characters.

By default, Guidewire uses xxtl_<*typelist name*> as the name of the typelist table (where "xx" is the application's code). If you want a different table name, you can override the default value by specifying a value in the "Table name" field in Studio. If you override the default value, the table name becomes xxtl_<*table name*>.

Guidewire restricts the typelist table name to ASCII letters, digits, and the underscore character. Guidewire also places limits on the length of the name.

Guidewire imposes these restrictions because of the restrictions imposed by the various relational database management systems (RDBMS) with which Guidewire applications interact. If you choose, however, you can override the name of the typelist and thus the table name stored in the database.

A "final" typelist is a typelist to which no additional typecodes can be added. All custom typelists are non-final.



Use the Typelist Editor to add typecode, categories, and filters.

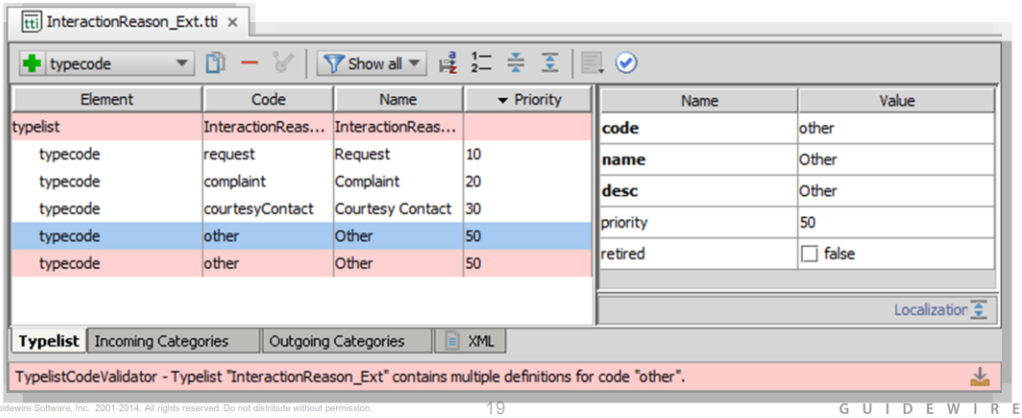
If a list view is sorted based on a typekey field, then the rows are sorted using standard typecode sorting. Standard typecode sorting is first by priority and then by name, and items with a priority of -1 appearing at the beginning of the list.

In a production environment, you should never delete a typecode or modify the typecode's code value. Either of these behaviors can cause problems because there could be existing objects that reference the altered or deleted typecode. In situations like this, you should mark the typecode's retired value as true. Retired=true preserves the typecode for objects that already make use of the typecode, and prevents the use of the typecode by new objects or by existing objects that don't already use the typecode. For example, assume you have a Color typelist that is used by the Vehicle entity and you retire the typecode brown. Any existing vehicle with the color brown will be unaffected. But new vehicles cannot be set to brown and existing vehicles that are not brown cannot be changed to brown.

Validate the schema

Click Validate in the toolbar

- Red highlight indicates schema violation warning
- Schema violations explained in pane below editor



Step 3: Optionally regenerate dictionary

- **`gwXX regen-dictionary`**
- Process builds entire entity model including base and custom typelists
- Identifies errors in the data model beyond Typelist Editor schema validation

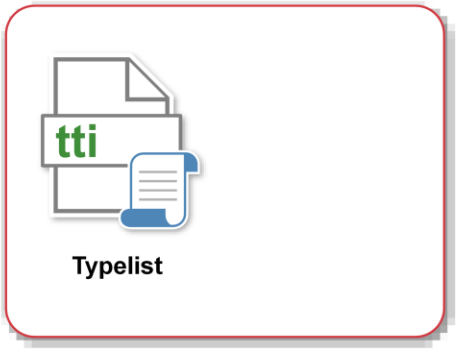
```
C:\Guidewire\TrainingApp\bin>gwta regen-dictionary
regen-entity-model-xml:
=======================================================
= Running main class:
  com.guidewire.tools.dictionary.data.EntityModelXmlTool
     [java] --- Guidewire Entity Model In Xml ---
…
ERROR Errors found in DoctorSpecialtyType
ERROR TypelistCategoriesValidator - Typecode
      DoctorSpecialtyType.Critical Care Medicine refers to a
      non-existent category
```


Regenerating the data dictionary is not required, but doing so can identify flawed XML in the data model that go beyond schema validation. For example, a deleted typecode that is referenced by other typelists or entities will throw an error.

Step 4: Deploy the typelist

Restart Server

- Typelist



- bin command window
  - **`gwXX dev-stop`**
  - **`gwXX dev-start`**
- Or, Guidewire Studio
  - Run → Stop
  - Run 'Server' or Debug 'Server'
- During start-up
  - If autoupgrade=true in database-config.xml, then Guidewire attempts to upgrade the database according to the changes in the data model

In Studio, the newly created typelist is available to reference immediately. However, the typelist is not available to the application server until the server is restarted.

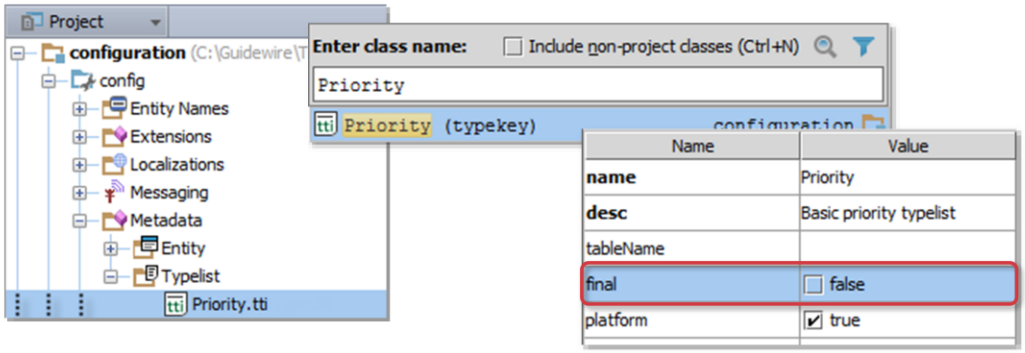
Lesson outline

- Typelist basics
- Create a typelist
- Create a typelist extension
- Defining typekey fields

Steps to create a typelist extension

1. Navigate to the typelist
2. Create a typelist extension file
3. Define typecodes
   a) Optionally Localize typecode descriptions
4. Optionally regenerate the dictionary
5. Deploy the typelist extension

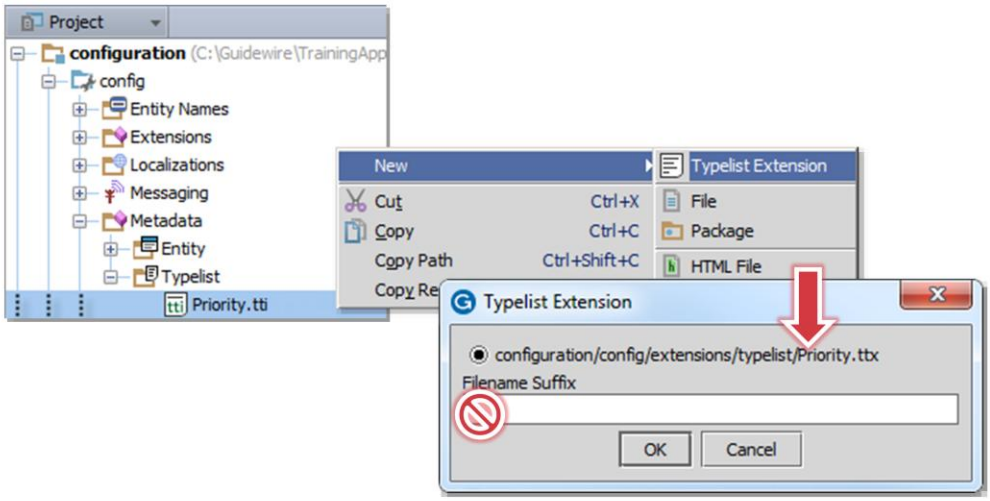
Step 1: Navigate to a typelist



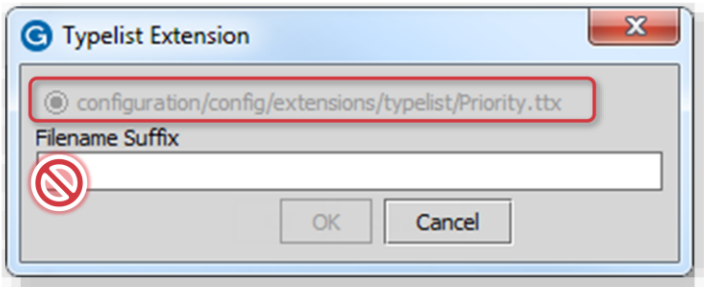
- Navigate to a typelist in …**`\Metadata\Typelist\`**
  - Project View or using CTRL+N
  - Studio will open an existing extension first!
    - If an existing extension already exists, STOP
    - Do NOT create a new extension; edit existing extension instead!
- Typelist (TTI) file must **NOT** be final=true

Step 2: Create a typelist extension file

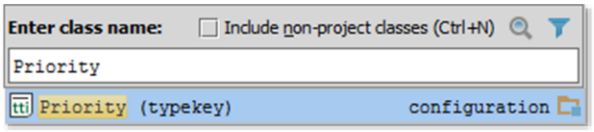
- Project View → Context menu → New → Typelist Extension
- Do **NOT** enter filename suffix; Click OK



Step 2: Create a typelist extension file (2)

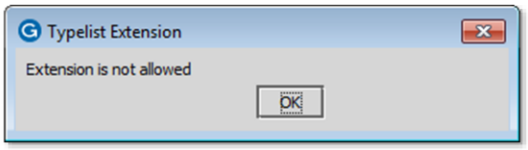


- Unable to click OK? Click Cancel
  - Grayed out path shows that a typelist extension (TTX) file already exists!
- Navigate to the typelist (CTRL+N)
  - Studio automatically opens the extension file first!
  - Edit the extension in the Typelist Editor

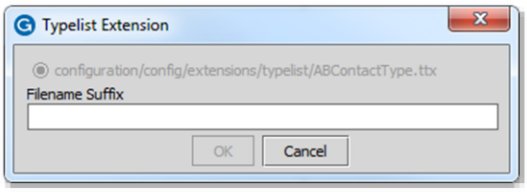


Step 2: Create a typelist extension file (3)

- For a typelist (TTI) in …**\Extensions\Typelist**
  - **NOT** possible to create typelist extension (TTX)
  - Edit the TTI file directly in the Typelist Editor
  -

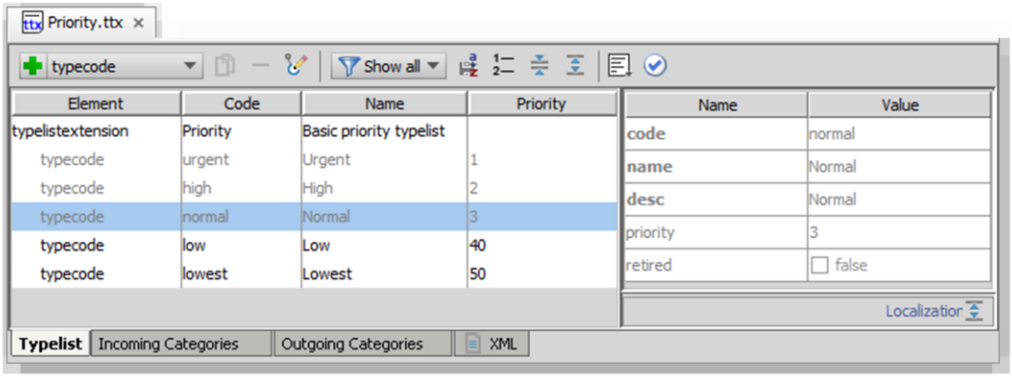


- For a typelist extension (TTX) in …**\Extensions\Typelist**
  - Do **NOT** create an extension for an extension
  - Click Cancel
  - Edit the TTX file directly in the Typelist Editor



It is not possible to create a typelist extension (TTX) for a typelist (TTI) that already exists in the /Extensions/typelist/ folder. Simply edit the existing TTI file.

Step 3: Define typecodes



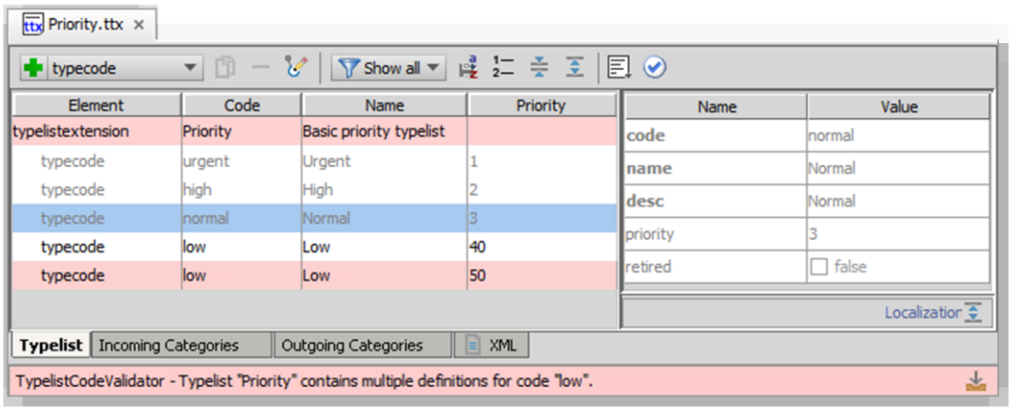
- Typelist (TTI) typecodes are in grey
  - Can change with from Show all to This file only
- Add new typecodes with toolbar or context menu to
- To change or remove typecodes, override element
  - Font color for overridden elements turn from grey to black

In the slide example, the low typecode has been overridden.

Validate the schema

Click Validate in the toolbar

- Red highlight indicates schema violation warning
- Schema violations explained in pane below editor



Step 4: Optionally regenerate dictionary

- **`gwXX regen-dictionary`**
- Process builds entire entity model including base and custom typelists
- Identifies errors in the data model beyond Typelist Editor schema validation

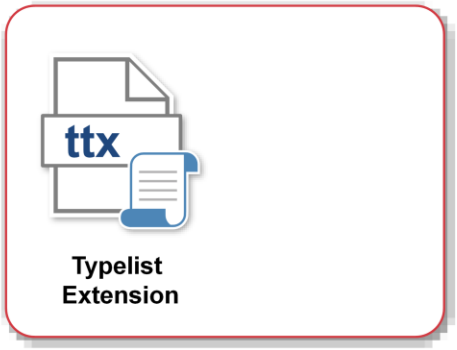
```
C:\Guidewire\TrainingApp\bin>gwta regen-dictionary
regen-entity-model-xml:
=====================================================
= Running main class:
  com.guidewire.tools.dictionary.data.EntityModelXmlTool
     [java] --- Guidewire Entity Model In Xml ---
…
ERROR Errors found in Priority
ERROR TypelistCategoriesValidator - Typecode "Priority" contains
      multiple definitions for code "low".
```


Regenerating the data dictionary is not required, but doing so can identify flawed XML in the data model that go beyond schema validation. For example, a deleted typecode that is referenced by other typelists or entities will throw an error.

Step 5: Deploy the typelist extension

Restart Server

- Typelist extension



- bin command window
  - **`gwXX dev-stop`**
  - **`gwXX dev-start`**
- Or, Guidewire Studio
  - Run → Stop
  - Run 'Server' or Debug 'Server'
- During start-up
  - If autoupgrade=true in database-config.xml, then Guidewire attempts to upgrade the database according to the changes in the data model

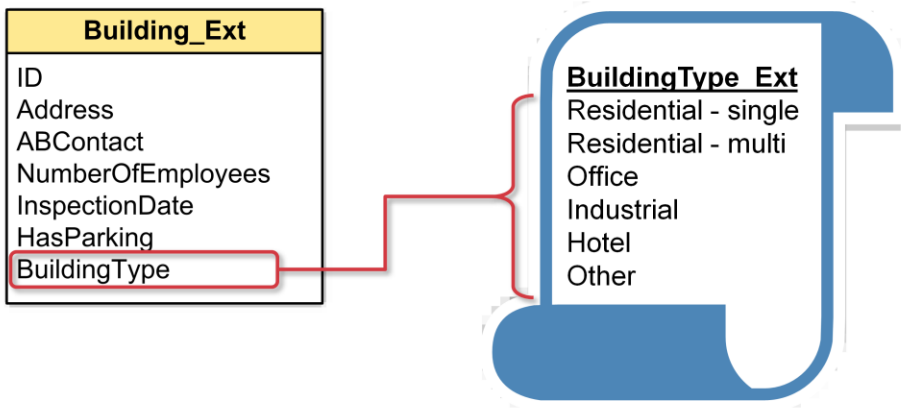
In Studio, the newly created typelist extension is available to reference immediately. However, the typelist extension is not available to the application server until the server is restarted.

Lesson outline

- Typelist basics
- Create a typelist
- Create a typelist extension
- Defining typekey fields

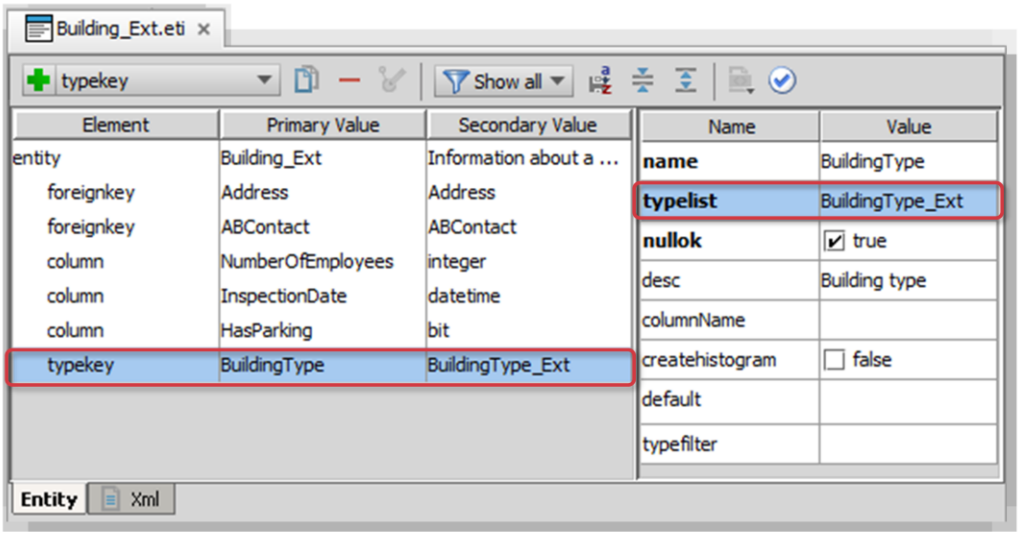
Typekey fields

- A **typekey field** is an entity defined field associated with a specific typelist
- Referenced typelist contains typecodes whose values are the only possible value for the typekey field



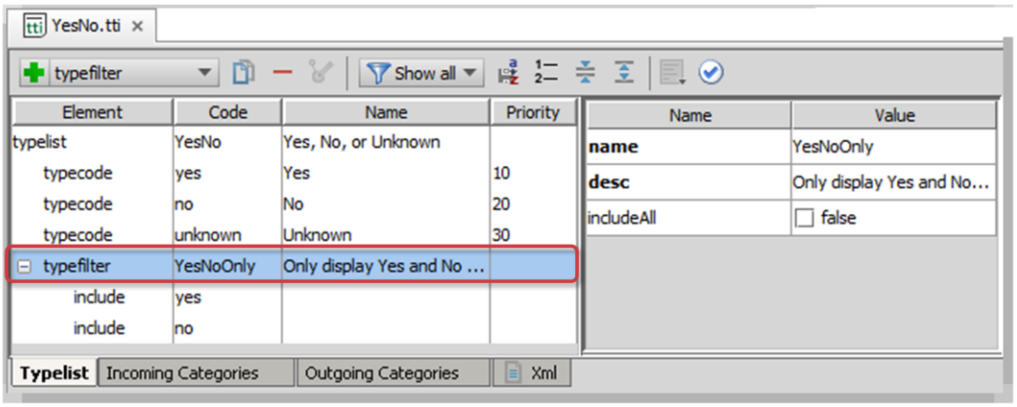
Entity configuration: typekey

- Add the typekey element to entity or entity extension
- Define the typelist attribute with specified typelist



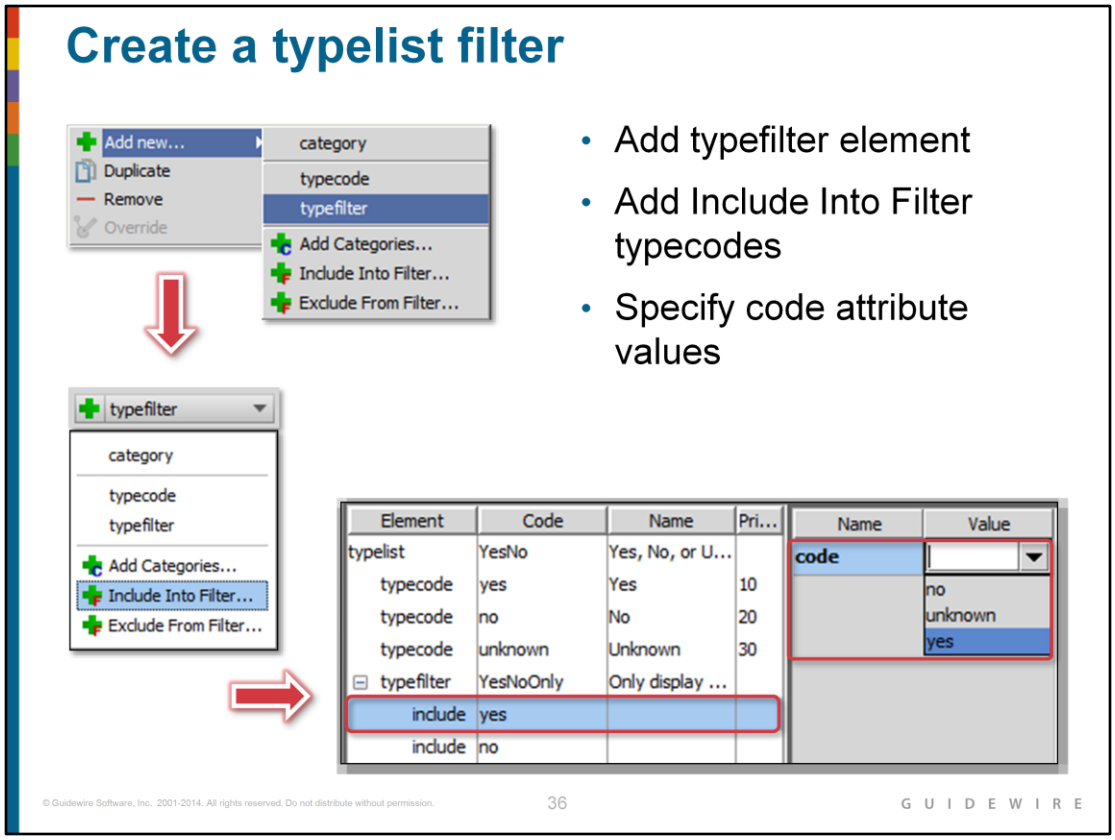
Typelist filters

- A **typelist filter** defines a subset of typecodes in the typelist
- Configure filters for typelists to filter available typecodes
- Example: YesNoOnly typefilter



It is possible to configure a typelist so that a Guidewire application can filter the typelist values so that they do not all appear in the drop-down list (typelist) in the application user interface.

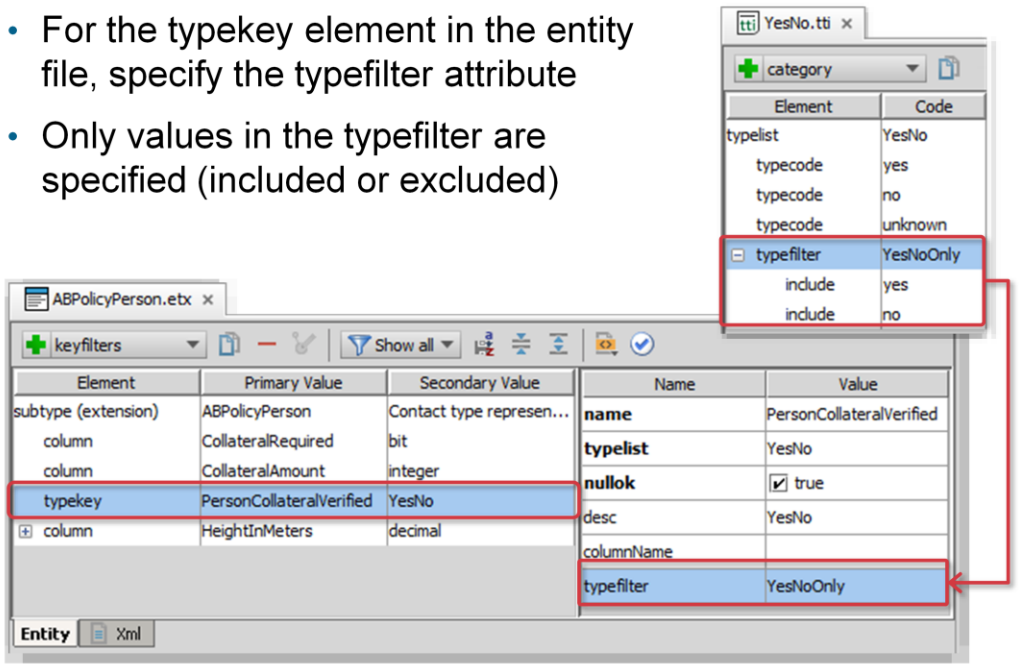
A typelist filter causes the typelist to display only a subset of the typecodes for that typelist. A filter narrows the list of typecodes to show in the typelist view in the application.

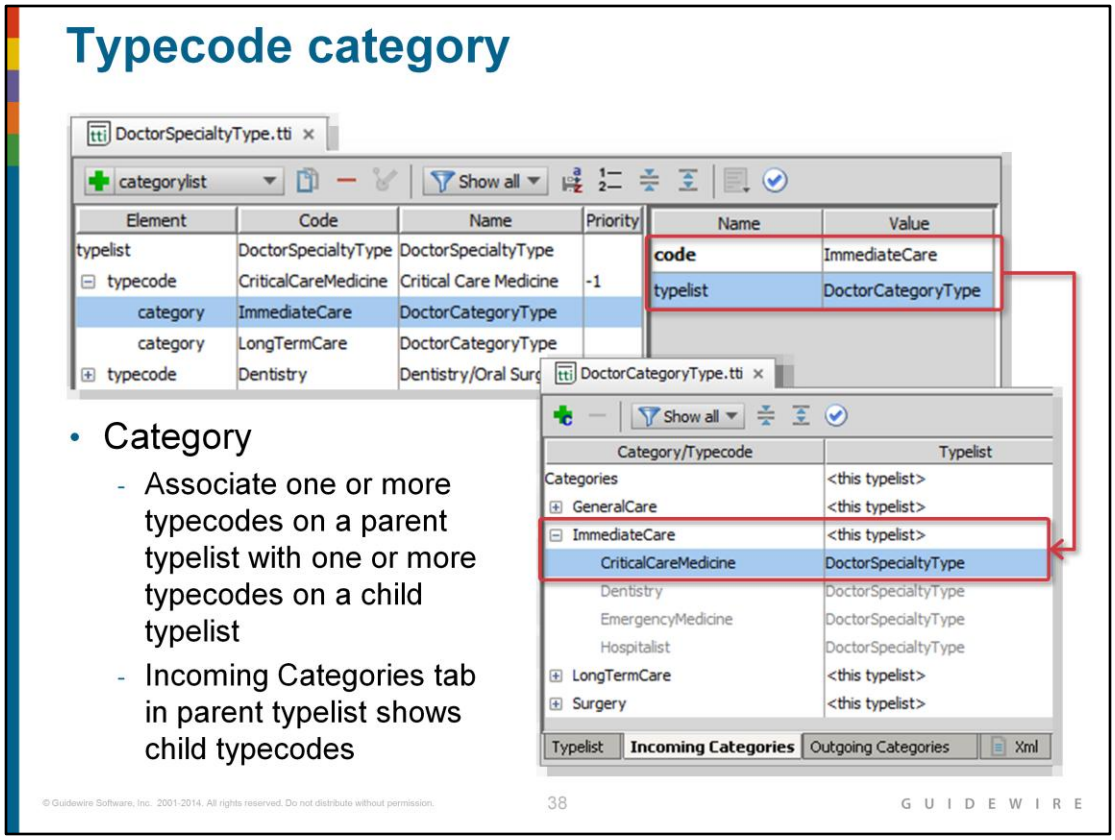


You define a typelist filter at the level of the typelist. Use the Typelist Editor to define a filter for a particular typelist.

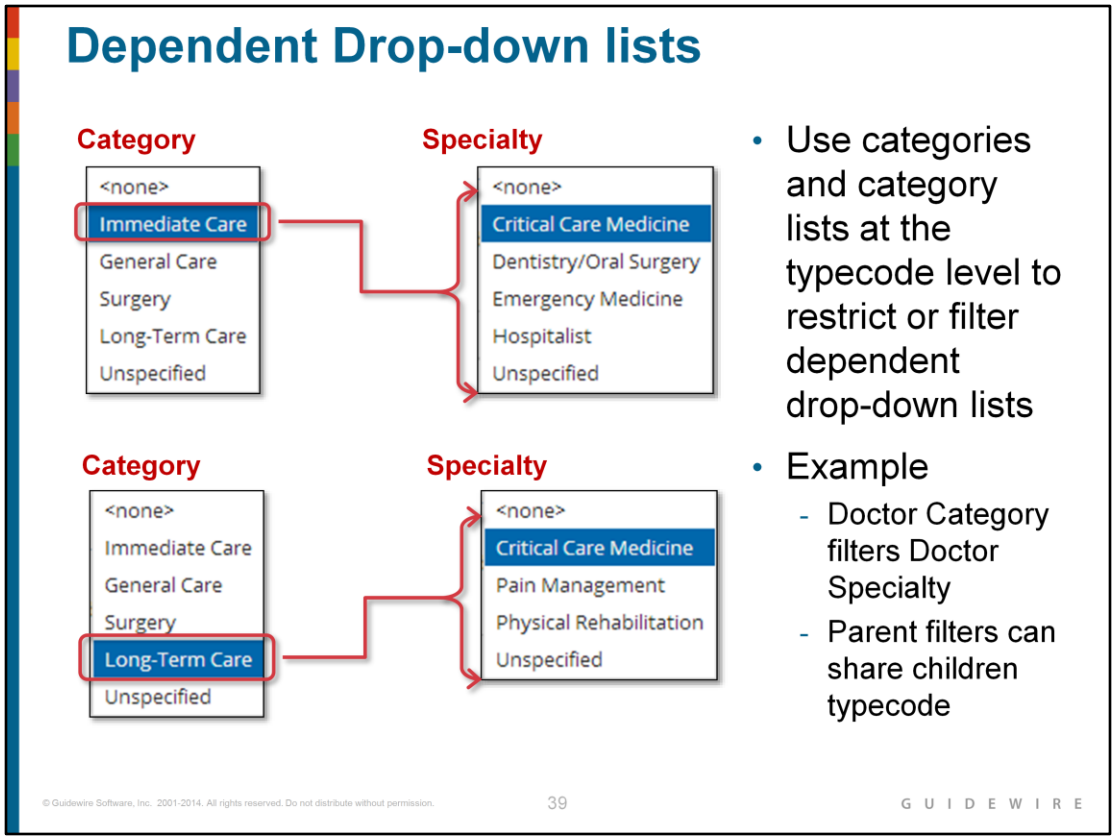
Referencing a typelist filter

- For the typekey element in the entity file, specify the typefilter attribute
- Only values in the typefilter are specified (included or excluded)





A typecode filter uses categories and category lists at the typecode level to restrict or filter a typelist.



The entity ABDoctor specifies that the DoctorSpecialtyType typekey has a keyfilter for the DoctorCategoryType typekey.

Later lessons discuss how to configure the user interface with Page Configuration Files and the Input widgets.

Lesson objectives review

- You should now be able to:
  - Describe the functionality of a typelist
  - Create a typelist and typelist extension
  - Create a typekey field
  - Create a typekey field that references a typelist filter

Review questions

1. Describe some of the differences between the files in the …**\metadata\typelist** and …**\extensions\typelist** folders.
2. Which is a valid code value for a typecode in a food coloring typelist?
   a) RedDye#1
   b) YellowDye40
   c) OrangeDye:10
3. Which typecode attribute determines the order in which typecodes are listed in the user interface?
4. What is the key difference between a typekey and a typekey with a defined typefilter attribute?

Answers

1) Files under metadata are read-only are specific to the base configuration. You cannot create typelist files in the metadata directory but can create typelist files in the extensions directory. Only internal typelist extension files (TIX) are in the metedata directory. Only external typelist files (TTX) are in the extensions directory. Both directories contain typelist (TTI) files.

2) The typecode must be unique within the typelist and must have 50 characters or fewer. The value for the code must be an alphanumeric value.

2a) Not valid

2b) Valid

2c) Not valid

3) The priority attribute determines the sort order in the User Interface for a typelist. After priority, the name (not code) determines the sort order. Entries with priority -1 appear at the end of the list

4) A typekey is only limited by the values in the typelist. A typekey with a defined typefilter attribute is limited to the named subset of values as defined by the typekey filter in the typelist.

Notices