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.

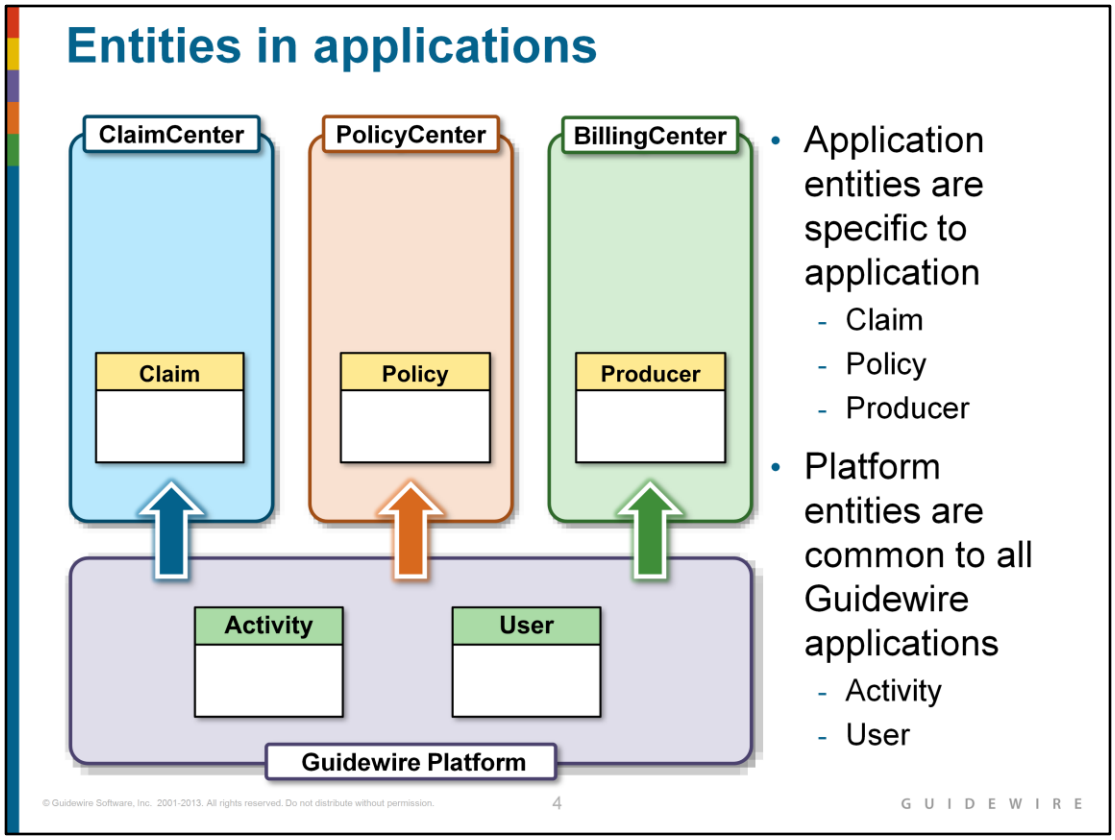
Lesson objectives

- By the end of this lesson, you should be able to:
  - Distinguish between platform, application, and customer entities
  - Edit an entity extension
  - Create an entity extension

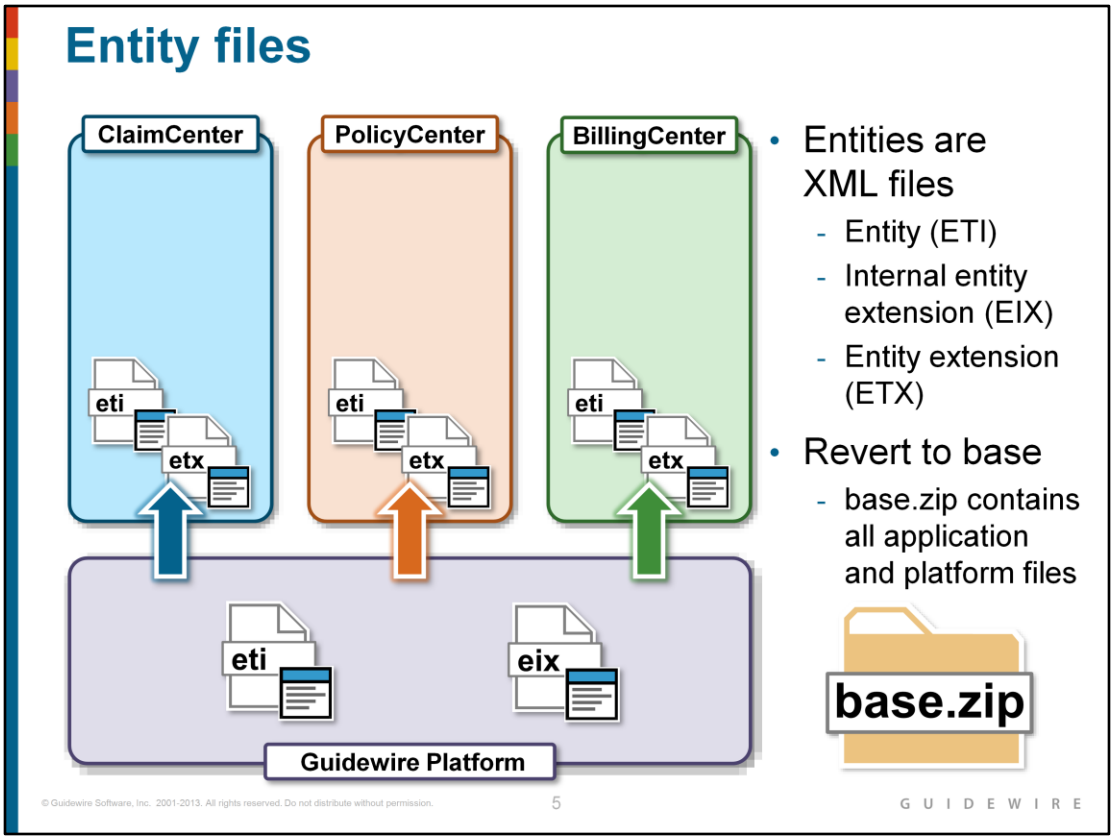
This lesson uses the notes section for additional explanation and information.
To view the notes in PowerPoint, select View → Normal or View → Notes Page.
When printing notes, select Note Pages and Print hidden slides.

Lesson outline

- Base application entities
- Entity Editor
- Edit an entity extension
- Create an entity extension



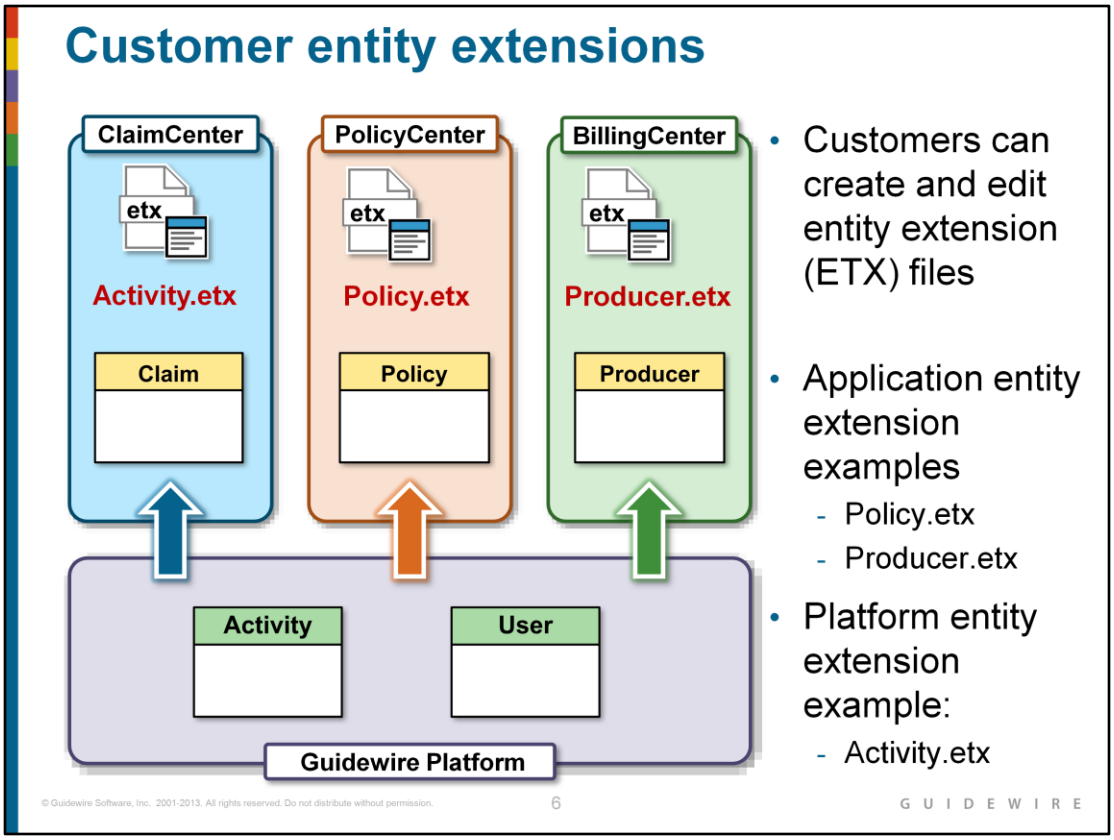
Activity and User are entities common to all Guidewire applications. Claim is specific to ClaimCenter. Quote is specific to PolicyCenter. Account, Policy, and Producer are the three key entities to BillingCenter. Entities are defined in XML files.



Platform entities are always found in the …\modules\configuration\config\metadata\entity\ folder. All ETI and EIX files in the \metadata\entity\ are read-only. You cannot edit these files directly in Guidewire Studio and should not edit these files in any other application. Many platform entities have the platform="true" attribute defined in the <entity /> element. Ignore the deprecated base="true" or base="false" attribute in the <entity /> element.

Some application specific entities can also be found in the \metadata\entity\ folder. Many application entities are extendable in the sense that they can be both subtyped and/or an entity extension can be created. Entities with the final="true" attribute defined in the <entity /> element cannot be subtyped. Entities with the extendable="false" attribute defined in the <entity /> element cannot be extended in the sense that new elements cannot be added to the entity.

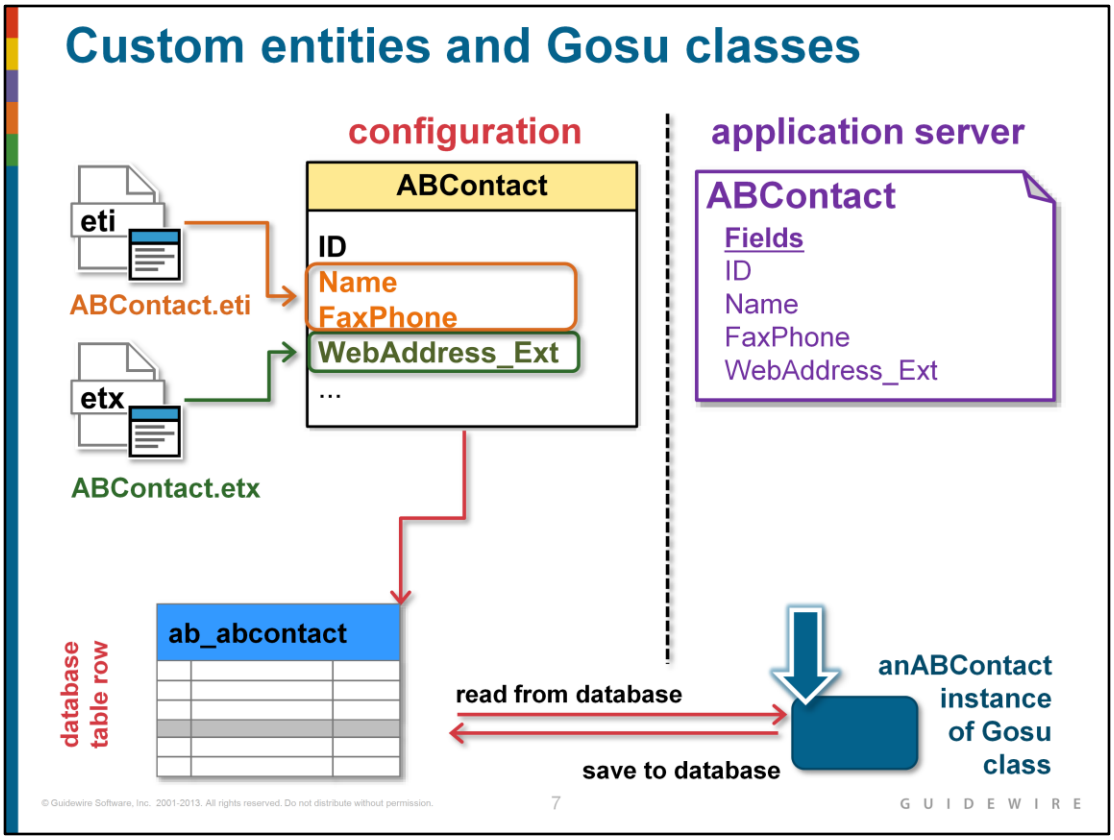
You can always determine the base configuration for a Guidewire application by looking at the <installDirectory>\modules\base.zip file. Base.zip is a compressed zip file contains the base configuration files for both the platform and the application. The base.zip file is not exposed in the Guidewire Studio project directly. However, for many base application files in Guidewire Studio, you can revert the file back to base file contained in base.zip. Rule folders and files cannot be reverted in Studio back to base.



In a given application, a platform-level entity can have both EIX and ETX files. The EIX file contains extensions to the platform-level entities that are required for the base application and cannot be modified. EIX files are extensions to platform-layer entities created by Guidewire development to meet the needs of a given application's base data model. For example, the Activity entity has different needs in each application:

- ClaimCenter needs an Activity.Claim field
- PolicyCenter needs an Activity.Job field
- BillingCenter needs an Activity.TroubleTicket field

And in the case of ClaimCenter, Activity.etx represents an platform extension that customers can also modify. Similarly, customer can modify Policy.etx and Producer.etx in their applications.



For a base application data model entity, the entity is defined by both its original ETI file and the ETX extending file, if one exists. In certain cases, there is an internal entity extension, EIX, file. Regardless of which file the field is declared in, all fields from all files become the internal Gosu class.

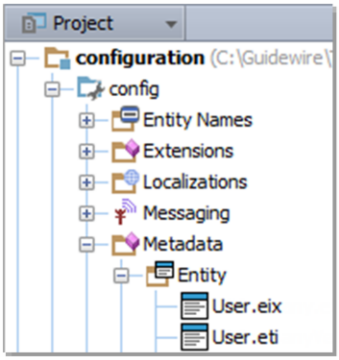
For fields that are added to a base application entity, Guidewire recommends that the field name should end with _Ext (or start with Ext_). In the slide example, the ABContact.etx entity extension file contains the WebAddress_Ext field.

Students coming from an Object-Oriented Programming (OOP) background should be aware that the term "extend" gets used in OOP differently than it does in Guidewire. In OOP, the term "extend" is used to refer to creating a new subclass that extends some parent superclass.

In Guidewire data model configuration, the term "extend" is sometimes used to refer to adding new fields to an existing base application entity. In this sense, no new subclass or subtype entity is getting created. One is simply adding additional fields to an existing base application entity.

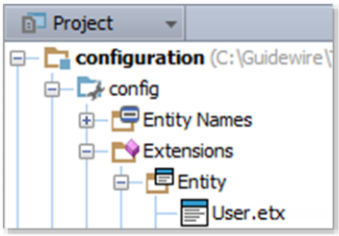
Project view entity files

…\metadata\entity\



- Read-only files
  - Entity (ETI)
  - Internal entity extension (EIX)

…\extensions\entity



- Editable files
  - Entity (ETI)
  - Entity extension (ETX)

Guidewire provides certain entity extensions as part of the base application configuration. Many of the extension index definitions address performance issues. Other extensions provide the ability to configure the data model in ways that would not be possible if the extension was part of the base data model. Do not simply overwrite a Guidewire extension with your own extension without understanding the full implications of the change.

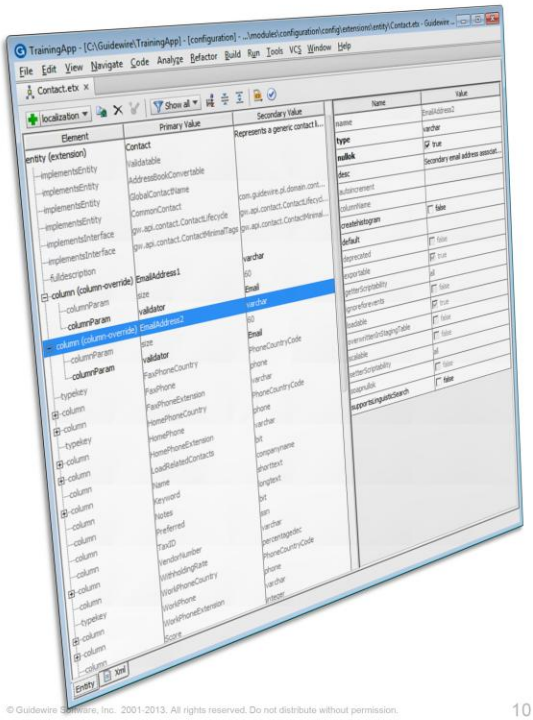
Because EIX (entity internal extension) files are neither created nor modified by configuration developers, this lesson does not discuss their structure. If you need to make multiple extensions to a single entity at different points in time, all extensions for that entity should be added to the same ETX file.

As a customer, you can create customer entities (ETI), create extension entities (ETX). You can alsoedit editable entity (ETI) and entity extension (ETX) files.

Lesson outline

- Base application entities
- Entity Editor
- Edit an entity extension
- Create an entity extension

Entity editor



- View, define and edit an entity, entity extension, or internal entity extension
  - **\extensions\entity**
    - ETI and ETX files
  - **\metadata\entity**
    - ETI and EIX files
- View file as XML
- Consists of
  - Editor toolbar
  - Element tree pane
  - Attribute pane

You can define and edit an entity and entity extension located in the …\configuration\config\extensions\entity\ folder.

The Entity Editor allows you to view an entity or internal entity extension located in the \configuration\config\metadata\entity folder. Files in the \metadata\ folder are read-only.

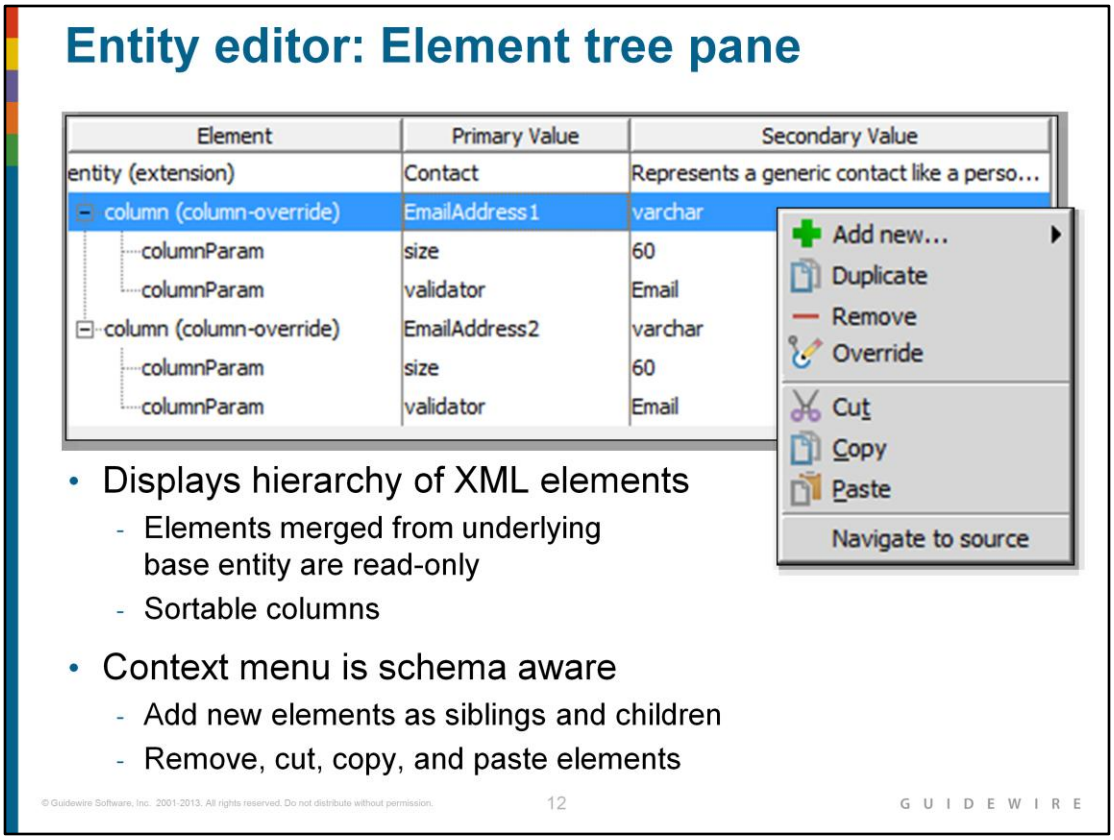
Entity editor: Toolbar reference

		Icon	Description
Actions	Edit	array	Add an element; Drop-down list is schema aware
			Duplicate selected element
			Delete selected element
			Override selected element
	View	Show all	Filter elements by file
			Persist sort order
			Collapse nested elements
			Expand nested elements
	Navigate		Navigate to supertype and/or subtype
		User.etx	Click link to navigate to extension
	Validate		Validate entity

Add, duplicate, remove, and override elements. The editor is data model / XSD aware. Editing with the editor is only available in /extensions/.

When you define and edit an entity or entity extension located in the \configuration\config\extensions\entity folder, the Entity editor toolbar is fully enabled. Entities and internal entity extension in the located in the \configuration\config\metadata\entity folder are read-only and all Edit actions are disabled.

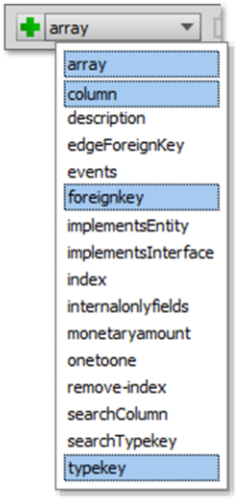
Sort, filter, and collapse columns in the element tree pane. You can also easily navigate to supertype and subtype entities. Lastly, you can quickly validate your entity XML.

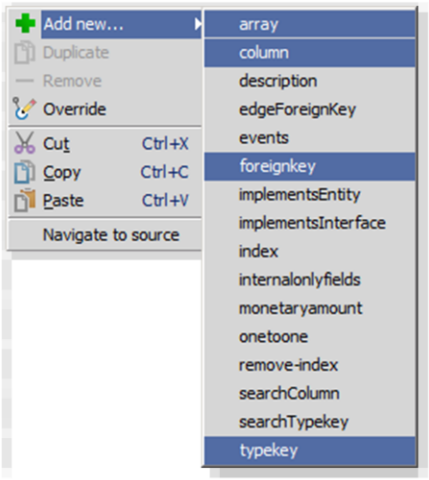


The Entity editor toolbar view actions influence the display of the Element tree pane. By default, the Element tree pane displays the hierarchy of nested elements, including any inherited elements from base entities. For entity extensions, the underlying base entities are entities and internal entity extensions. The editor is schema aware for elements and attributes.

Adding elements

- Toolbar
  - Select option in dropdown
  - Click **+** to add more of same
- Context menu
  - Add new…
  - Select option in menu

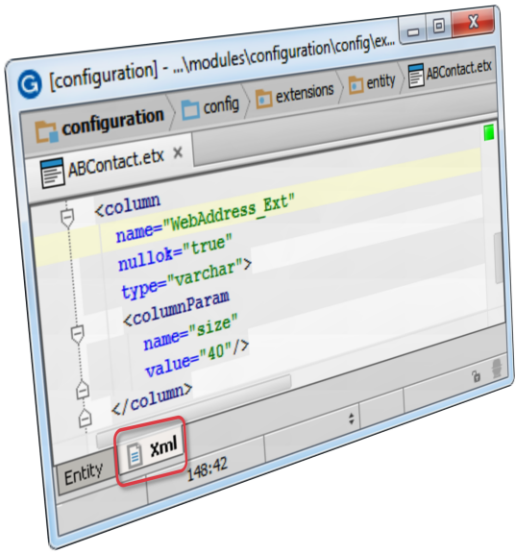




When creating elements, the Entity Editor refers to an Extensible Schema Definition (XSD) file, datamodel.xsd. The location of the file is <InstallRoot>\modules\configuration\xsd\metadata\datamodel.xsd. Not all elements define fields in a database, for example, <events />. Other elements are specific to database performance such as <index />.

The four primary elements listed above are common to entity extension (<extension>), entity declaration (<entity>), subtype extension (<extension>), and subtype declaration (<subtype>).

Common elements to add

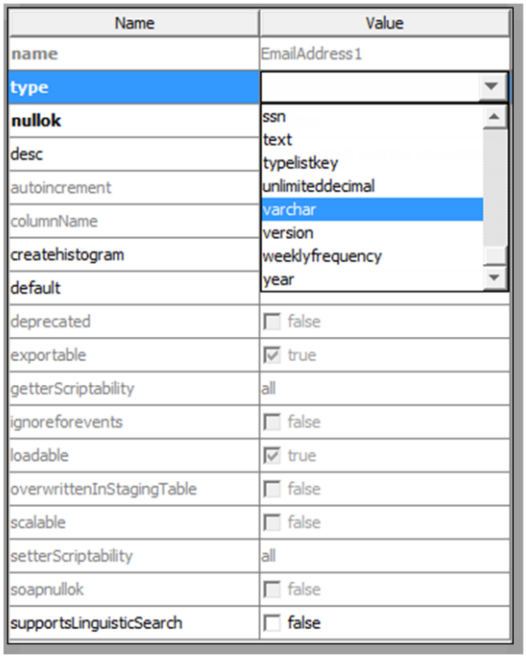


- Xml tab shows read-only view of elements

- Common elements for entity extension, entity, subtype, subtype extension are:
  - <array …/>
    - Define array entity and field
  - <column …/>
    - Define data field with defined data type
    - Bit, datetime, integer, varchar
  - <foreignkey …/>
    - Define foreign key field and entity
  - <typekey …/>
    - Define typekey and related typelist

The four primary elements listed above are common to entity extension (<extension>), entity declaration (<entity>), subtype extension (<extension>), and subtype declaration (<subtype>). You can view the XML elements by clicking on the Xml tab.

Entity editor: Attribute pane



- Name Value columns
  - For selected element, define attributes
- Schema aware values
  - Boolean controls
  - Dropdown lists
- Attribute styling
  - Bold for required; Black for editable
  - Grayed-out for non-editable
    - Overridden, Inherited, Internal, Default
- Set nullok= true in most cases as default is false

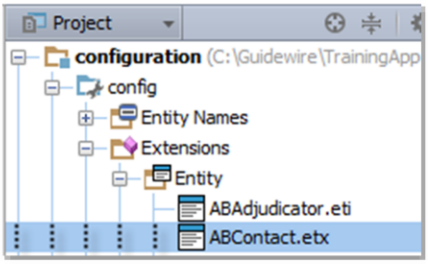
Lesson outline

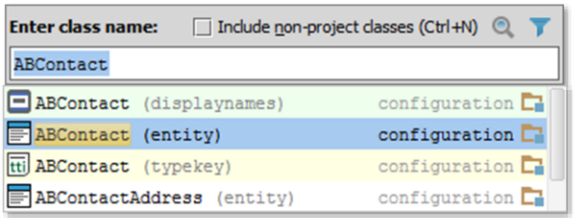
- Base application entities
- Entity Editor
- Edit an entity extension
- Create an entity extension

Steps to edit an entity extension

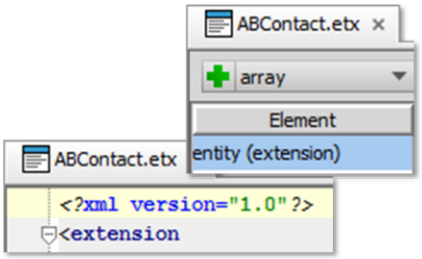
1. Navigate to the entity extension
2. Add elements (and subelements) and specify attribute values
3. Optionally regenerate the dictionary
4. Deploy the extension entity

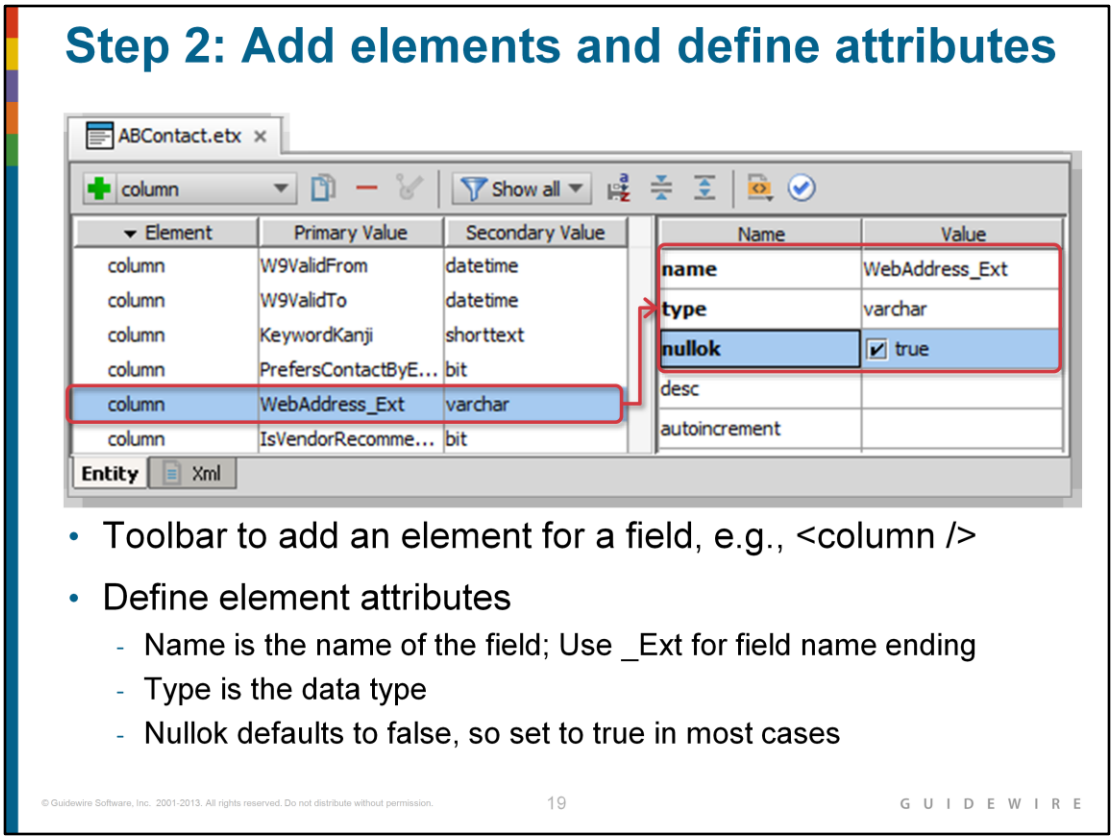
Step 1: Navigate to the entity extension





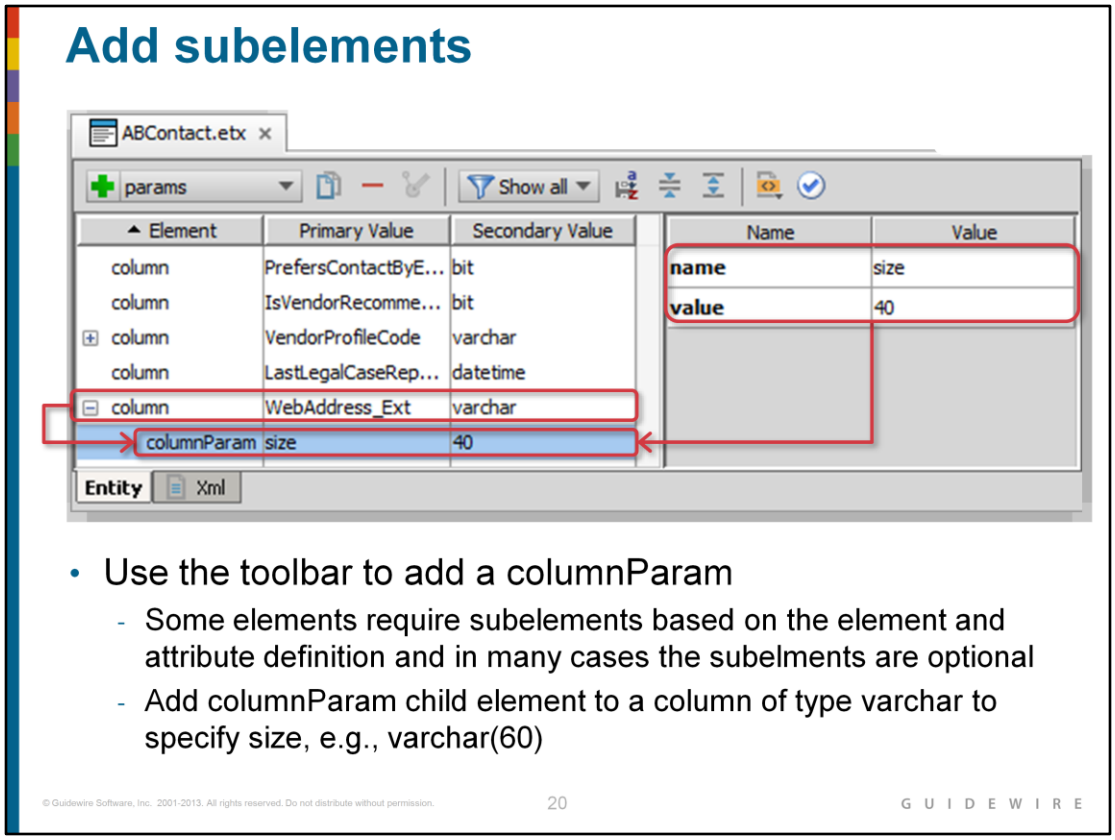
- Navigate to an entity in …**\Extensions\Entity**
  - Project View or using CTRL+N
- Verify that you have selected an entity extension file (ETX)
  - Top element reads entity (extension)
  - XML is <extension />





The nullok attribute has a default value for nullok is false. A column on a base application entity in which nullok is false may require a default value either because rows in the table exist already, or the application does a test import into the entity during the database upgrade, or both. A column on a custom entity in which nullok is true generally does not require a default value.

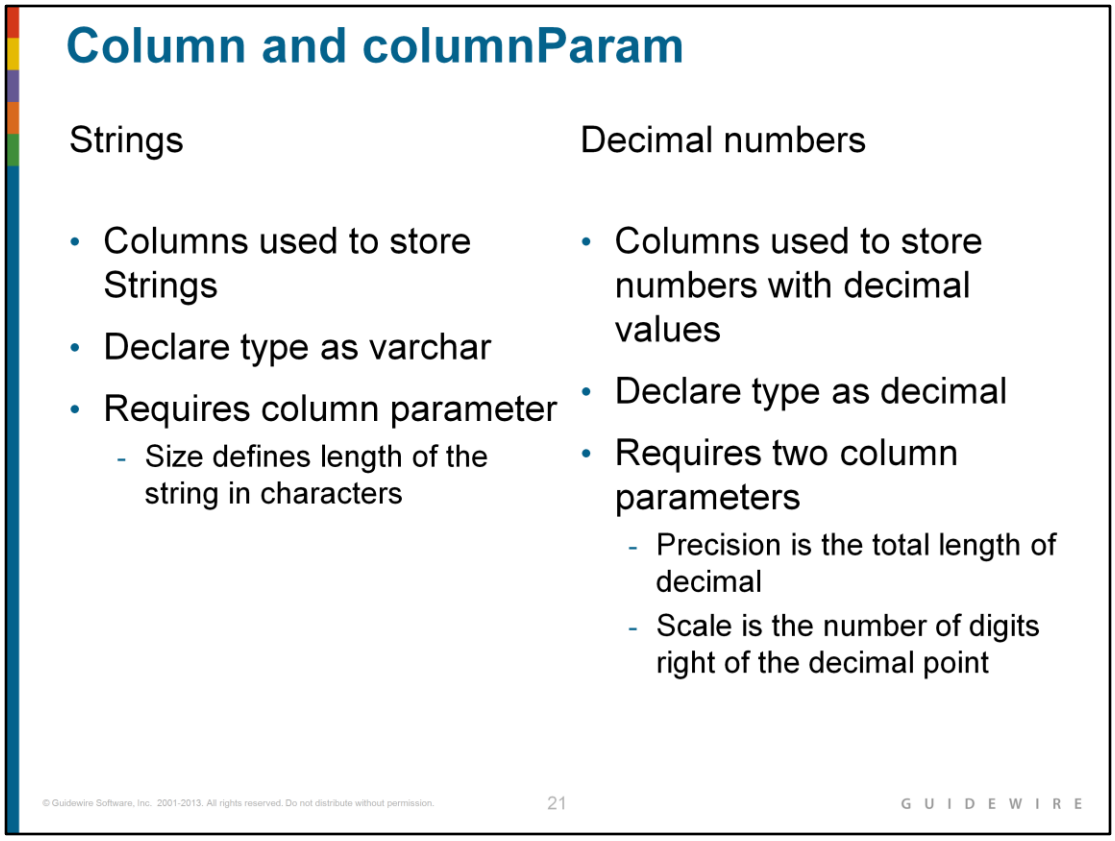
The columnName attribute exists because many databases have a restriction on the size of their column names. The lower limit for this maximum is typically 30. This can be a problem when field names are longer than 30 characters, which can occur when descriptive field names are requested by the customer. With the use of the columnName attribute, a Guidewire application can accommodate both requirements. The name attribute has no maximum size, so it can be as descriptive as desired. The columnName attribute has a 30 character limit, and should be used when the name attribute is greater than 30 characters.



You use the <columnParam> element to set parameters that a column type requires. The type attribute of a column determines which parameters you can set or modify by using the <columnParam> subelement.

You can determine the list of parameters that a column type supports by looking up the type definition in its .dti file. You can find the datatypes dti files in the datatypes folder in …\modules\configuration\config\datatypes\. For example, the varchar.dti files defines the possible parameters for the varchar column type.

The Entity Editor is schema aware and knows displays the available options for you when you select the type.

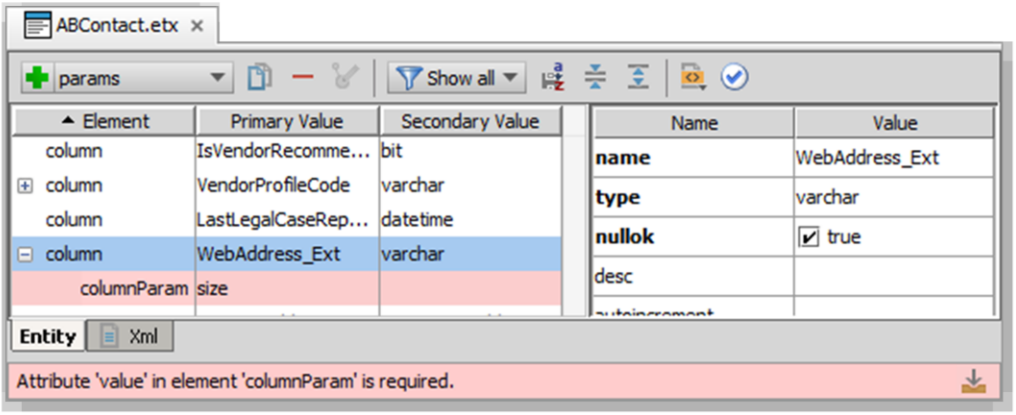


The decimal data type works with non-integer numeric values (in other words, decimal values). The precision denotes the total number of digits of the entire value. The scale denotes the number of digits after the decimal point. For example, a decimal with precision 6 and scale 2 could include values from -9999.99 to 9999.99 (6 digits in total with two digits appearing after the decimal point).

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 entities
- Identifies errors in the data model beyond Entity 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 ABContact
ERROR Attribute 'value' in element 'columnParam' is required.
ERROR ColumnIsValid - The column "WebAddress_Ext" on entity
      "ABContact" declares an invalid data type, "varchar". null
```


You can also optionally regenerate the data dictionary to add the new entity to the data dictionary and to check for problems in the data model. Regenerating the data dictionary is not required, but doing so can identify flawed XML in the data model that go beyond schema validation such as certain types of referential integrity.

Step 4: Deploy the entity extension

Restart Server

- Entity 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

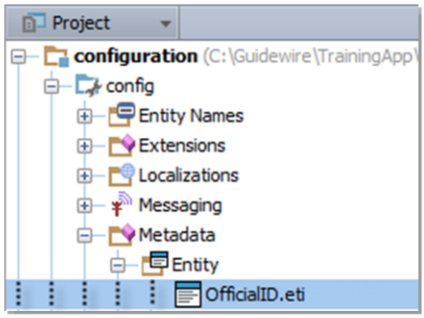
Lesson outline

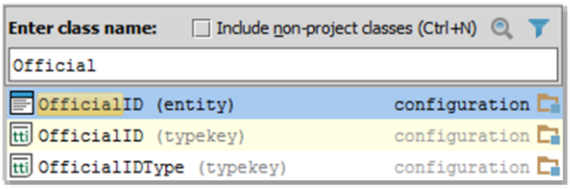
- Base application entities
- Entity Editor
- Edit an entity extension
- Create an entity extension

Steps to create an entity extension

1. Navigate to the entity
2. Create an entity extension file
3. Add elements (and subelements) and specify attribute values
4. Optionally regenerate the dictionary
5. Deploy the extension entity

Step 1: Navigate to the entity (1)

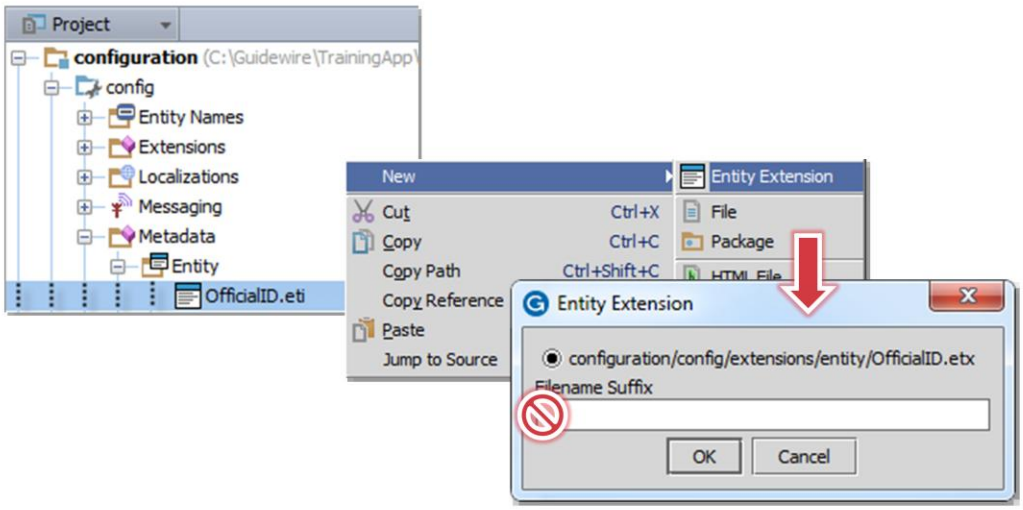




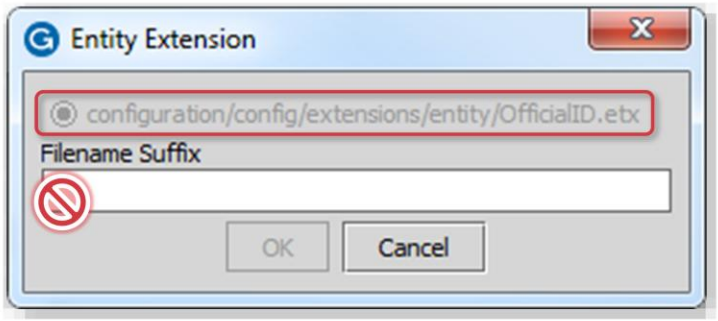
- Navigate to an entity in …**\Metadata\Entity**
  - Project View or using CTRL+N
- Studio will open an existing extension first
  - If an existing extension already exists, STOP
  - Do NOT create a new extension; edit existing extension instead!
- Verify that you have selected an entity file (ETI)
  - Not EIX or ETX

Step 2: Create an entity extension file (1)

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



Step 2: Create an entity extension file (2)

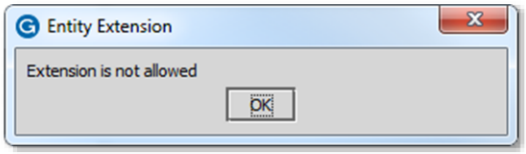


- Unable to click OK? Click Cancel
  - Grayed out path shows that an entity extension (ETX) file already exists!
- Navigate to the entity (CTRL+N)
  - Studio automatically opens the extension file first!
  - Edit the extension in the Entity Editor

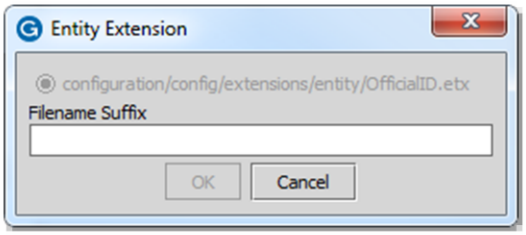
If an extension already exists, simply edit the extension rather than creating another extension using a suffix.

Step 2: Create an entity extension file (3)

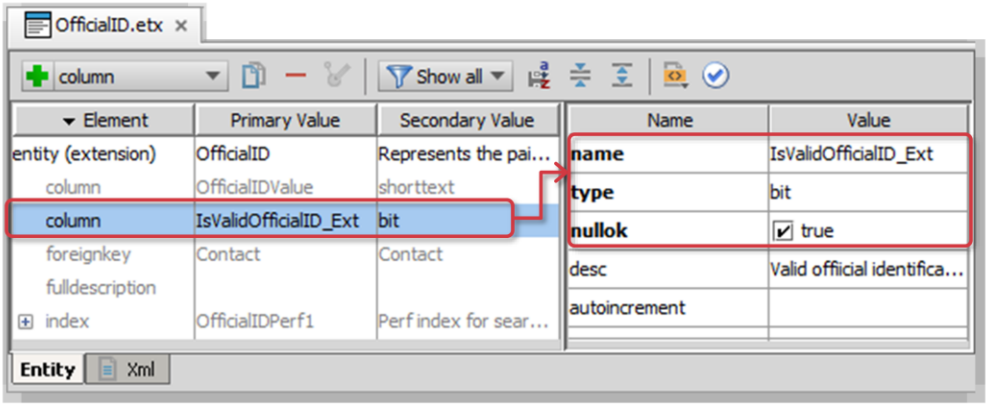
- For an entity (ETI) in …**\Extensions\Entity**
  - **NOT** possible to create entity extension (ETX)
  - Edit the ETI file directly in the Entity Editor



- For an entity extension (ETX) in …**\Extensions\Entity**
  - Do **NOT** create an extension for an extension
  - Click Cancel
  - Edit the ETX file directly in the Entity Editor



Step 3: Add elements and define attributes

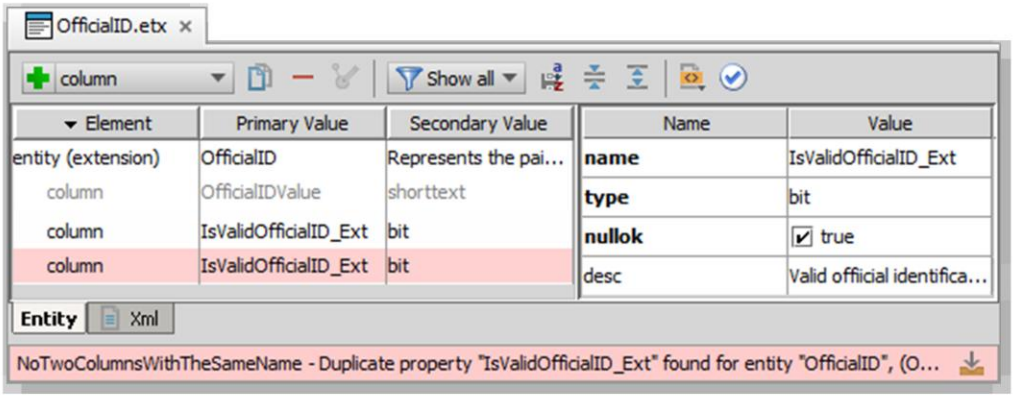


- Toolbar to add an element for a field, e.g., <column />
- Define element attributes
  - Name is the name of the field
  - Use _Ext for field name ending
  - Type is the data type
  - Nullok defaults to false, so set to true in most cases

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 entities
- Identifies errors in the data model beyond Entity Editor schema validation

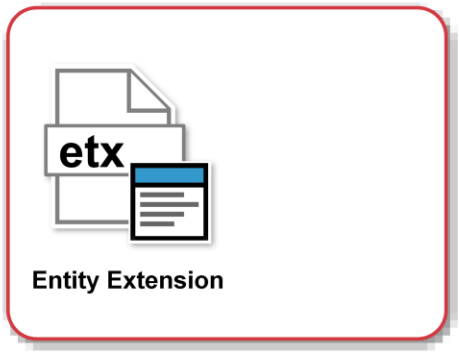
```
C:\Guidewire\TrainingApp\bin>gwta regen-dictionary
regen-entity-model-xml:
=====================================================
= Running main class:
  com.guidewire.tools.dictionary.data.EntityModelXmlTool
     [java] --- Guidewire Entity Model In Xml ---
…
ERROR Errors found in OfficalID
ERROR NoTwoColumnsWithTheSameName property "IsValidOfficialID_Ext"
      found for entity "OfficialID"
```


You can also optionally regenerate the data dictionary to add the new entity to the data dictionary and to check for problems in the data model. Regenerating the data dictionary is not required, but doing so can identify flawed XML in the data model that go beyond schema validation such as certain types of referential integrity.

Step 5: Deploy the entity extension

Restart Server

- Entity 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

Lesson objectives review

- You should now be able to:
  - Distinguish between platform, application, and customer entities
  - Edit an entity extension
  - Create an entity extension

Review questions

1. How many ETX files can a given entity have?
2. What is the primary element of an ETX file?
3. What datatype do you enter for fields that will store….
   a) String values?
   b) Boolean values?
   c) Numbers with decimal values?
4. Which datatypes require columnParam subelements?
5. Under what circumstances would you want to regenerate the Data Dictionary?
6. When does a Guidewire application actually modify the physical structure of the database?

Answers
1) One ETX file per entity in most cases.
2) The primary element is <extension>
3a) varchar
3b) bit
3c) decimal
4) varchar requires size and decimal which requires precision and scale
5) You would want to regenerate the Data Dictionary whenever you extend the data model and are concerned with data model validation beyond schema validation in the Entity Editor. Regenerating the Data Dictionary also updates the dictionary files to include the new fields (or entities) that you have created.
6) The physical structure of the database is modified only during start-up.

Notices