Setting Up Entity Registry

Understanding Entity Registry

An entity is an object that provides access (view, create, or update) to data in a record. You implement an entity through an application class.

The system delivers entities related to constituent (personal), admissions, and enrollment data. The list of delivered entities may grow with new web services or new functionality that we may deliver in the future. You can create new entities and extend delivered entities (or modify them to a lesser degree). New entities can be built based on new or existing records. How the entity relates to the record is dependant on the entity type.

Note: You define entity relationships in the Entity Registry component. However, outside of the Entity Registry component, entities are unaware of each other. This means all the code related to a particular entity is entirely encapsulated in that entity. This keeps the entities clean and avoids including child entities code inside the parent entity. For instance, even if *Emergency Contact* entity is a child of *Constituent* entity, the latter does not know anything about *Emergency Contact* other than having a relationship defined in the Entity Registry component.

Use the Entity Registry component to configure how the system should use specific entities (or properties). Entity logic is designed to be-reusable for different purposes such as:

- Web services: Based on the specified application class, the Entity Registry component enables you to automatically generate code and XML schemas. You can then directly paste the generated code and schemas inside the application classes and Integration Broker messages. The system uses these classes and messages when a transaction is processed through web services. Web services can use the entities for moving the data to and from functionalities such as XML messages, error handling, save/delete, and any other functionality coded in the entity.
- Component data validation: Components and web services can use the same logic to do data validation. Currently, the easiest way to reuse the logic is in Savepostchange code. For example, the Application Transactions and the Constituent Staging components use this way.
- Beans: Beans are objects containing multiple types of data. Entities are a natural fit to support beans. This is because you can design entities to support multiple types of data, and make any transition to web services easy.
- APIs: The system can use entities to modify record data programmatically, with full access to all the common logic used by components or web services.
- Import/Export: The system can export data to XML files easily using the entities' ability to get data and encode it into XML. Import can be done using the ability to translate data from a XML file and save the translated data.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- Batch Processing: You can use entities to handle data in batch. This can be done by loading the data using XML matching the entity style, or by any other means such as populating the data into an entity directly.

The Admissions Application Web Services (AAWS) feature illustrates an example of how to use the Entity Registry component. In AAWS, an applicant uses a user interface to enter constituent (personal) and admissions application information. In such a case, the system first needs to stage the entered information, then validate it, and finally move the information to the corresponding production tables. We have delivered entities for the constituent data (for example, Names, Addresses, Emergency Contacts, and so on) and we have delivered entities for the admissions data (for example, Academic Interests, Academic Plan, Academic Program, and so on). For each of these entities, we have configured the name of the stage record and the name of the equivalent production record in the Entity Registry component. Based on this delivered Entity Registry component configuration, the system knows how to pass the information that exists inside the incoming and outgoing web services.

Warning! Configuring or extending an entity is a technical task and should only be performed by developers with strong Integration Broker, Application Package, PeopleTools skills and record structure.

Entity Component Adapter

The entity component adapter allows entities to bind to the component row sets. By binding to the component, entity validation and presave logic can be run on live component data. This allows us to better consolidate common logic and make sure that components, services, APIS, etc. all follow the same rules for the data. The Entity Component Adapter also serves to allow rules to be run based on component data.

Configuring Entity Types

This section discusses how to:

- Set up entity types.
- View entities for an entity type.

Pages Used to Configure Entity Types

Page Name	*Definition Name*	*Navigation*	*Usage*
Entity Type	SCC_ENT_TYPE	Set Up SACR, System Administration, Entity, Entity Type	Define categories of entities. This allows for more control over how the entity functions.
Implementation	SCC_ENT_TYPE_IMPL	Set Up SACR, System Administration, Entity, Entity Type, Implementation	View the list of entity names set up for an entity type.

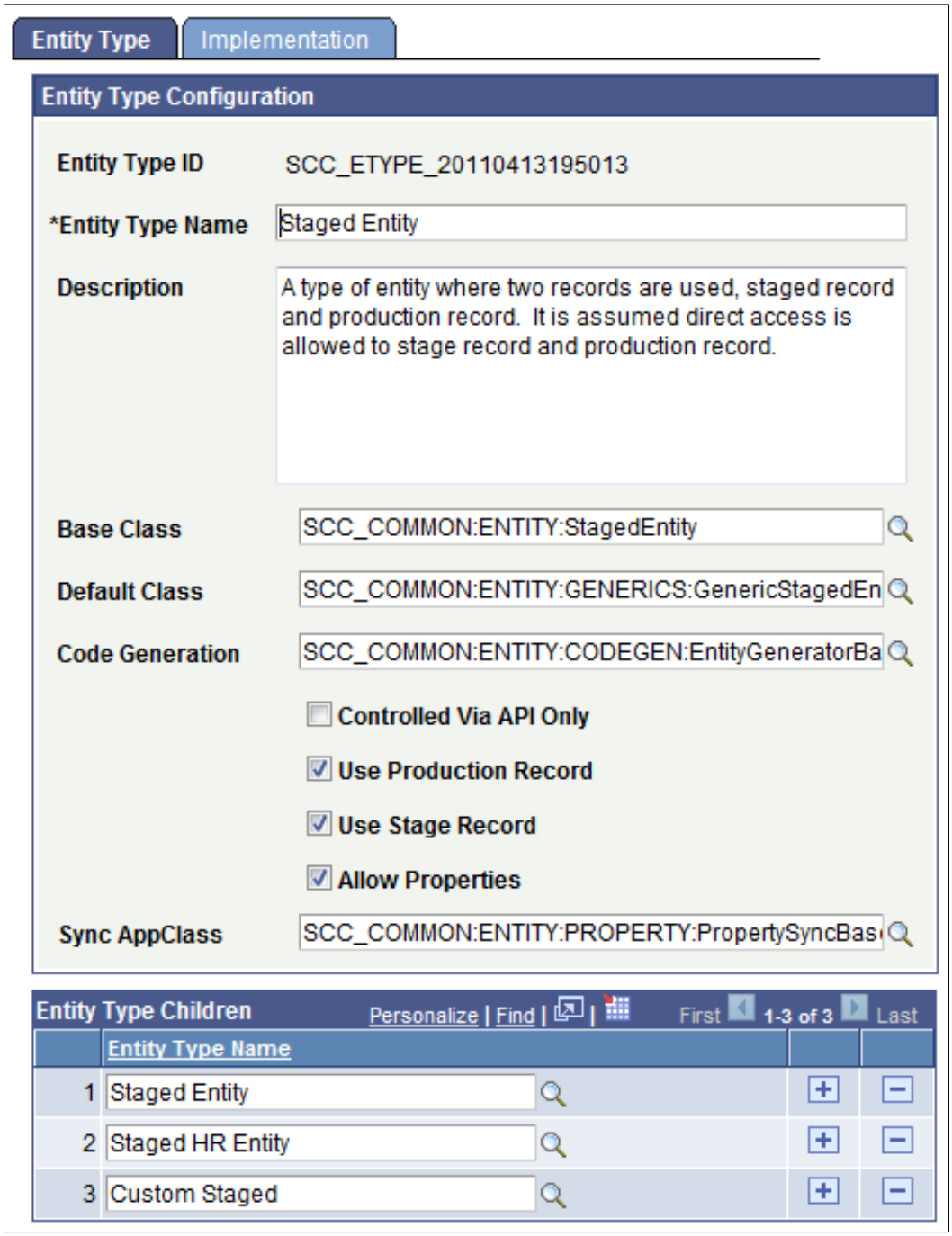
 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Setting Up Entity Types

Access the Entity Type page (Set Up SACR, System Administration, Entity, Entity Type).

Image: Entity Type page

This example illustrates the fields and controls on the Entity Type page. You can find definitions for the fields and controls later on this page.



Entity Type ID Displays the auto-generated unique ID.

When you add a new entity type, this field displays the value *NOID* until you save the record. When you save the new entity

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

type, the field displays the ID that the system assigned to the entity type.

Entity Type Name

Enter the name of the entity type.

The value that you enter here will become a prompt value for the Entity Type field on the Entity Registry page.

Description

Enter the description of the entity registry, primarily for documentation purposes.

Base Class

Enter the base application class that entities, who want to implement this type, should extend for code generation.

If you implement properties, the system uses the Base AppClass to find the baseProps for property generation. baseProps are properties that already exist on the parent and therefore do not need to exist on the child.

For an example of how a Base AppClass is coded, use Application Designer to review the application class SCC_COMMON:ENTITY:BasicEntity.

Note: Most of the code that you can use for entity functionality is located inside the SCC_COMMON:ENTITY:AbstractEntity. At a technical level all the delivered entity types, except *Custom,* extend AbstractEntity. AbstractEntity provides a lot of the functionality common across entities in the Entity Registry feature. For example, auto handling parent-child relationships, audit fields, save, delete, and so on are all handled by AbstractEntity, because they all function the same regardless of different types of data that entities handle. You will have to override the Entity Type base class if anything has to be customized for a particular type.

See Entity Application Class Reference.

Default Class

Specify the application class that can be used as a default for entities of this type. Entities using the default class cannot have any custom code, but will handle any property setup done through the Entity Registry. Providing a default class makes specifying an application class when creating an entity optional, otherwise it is mandatory.

All entities of this type must extend SCC_COMMON:ENTITY:IEntity. However, their constructor must take two arguments in the signature. The first is the parent entity. The second is the entityID for this entity as a string.

For examples of default classes see the package SCC_COMMON:ENTITY:GENERICS.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Code Generation

Enter the application class that the system uses to generate entity code when you select an Action of *Generate Code* on the Entity Registry page.

For an example of how an application class used for code generation is coded, use Application Designer to review the application class SCC_COMMON:ENTITY:CODEGEN:EntityGeneratorBase.

If you do not specify the code generation class, the system hides the *Generate Code* action when you select this entity type on the Entity Registry page.

See Setting Up Entity Registry.

Controlled Via API Only

Select if you want that no entities of the type can be added through the Entity Registry page. When set up via the API certain parts of the entity may be locked for editing through the Entity Registry component.

Use Production Record

Select if the entities tied to this entity type need a production record. If this check box is not selected, the Production Record field is not shown on the Entity Registry page. By default, every entity uses a production record.

Use Stage Record

Select if the entities tied to this entity type need a stage record. A stage record is useful if you want to temporarily store data for validation or processing before moving the data into a production record.

If you do not select this check box, the system does not display the Stage Record field on the Entity Registry page. The Apply Data Update Rule is also disabled because it is only used when stage records are involved.

See Understanding Entity Registry.

Allow Properties

Select if you want to configure the entity properties for all the entities tied to this entity type.

To configure the entity properties, select *Edit Properties* in the Action field on the Entity Registry page.

If you deselect the Allow Properties check box, the *Edit Properties* action is not available on the Entity Registry page.

Sync AppClass (Sync Application Class)

Specify an application class to synchronize the properties to the underlying records. While this is mostly needed if the entity type allows properties, you can also sync properties programmatically even if configuring the properties is not allowed. For the latter case, do not select the Allow Properties check box and enter the appropriate sync application class.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

For an example of a Sync AppClass, use Application Designer to review the application class SCC_COMMON:ENTITY:PROPERTY:PropertySyncBase.

See Synchronizing Entity Properties.

Entity Type Children

Use this grid to list the Entity Types for which you can add child entities on the Entity Registry page.

On the Entity Registry page, you can add child entities to a parent entity. You can add a child entity only if its entity type is entered in the Entity Type Children grid.

For example, if you create an entity type that uses production and stage records, you could list in this grid other entity types that use this setup. This will allow, in the Entity Registry page, to select child entities that are set up with an entity type different from the parent, but follows the same logic.

For details on how children entities are defined, see Setting Up Entity Registry, “Configuring Entity Registry.”

Rules Engine Data Set and Read Only Entity Types

The Rules Engine Data Set entity type supports entities that are not record based. These entities are purely meant to act as data structures. This version of data sets is used by the rules engine, and controlled via a custom, simplified interface.

See Understanding the Rules Engine.

Read Only entities are essentially basic entities for which save and delete are disabled. They make it easy to create entities on top of records for rules engine access. This makes it easy to add new entities accessible from the rules engine without introducing back door data updates. If at some point proper validation and/or presave logic is added, the entity can be switched to Basic Entity and save and delete would be enabled.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Viewing Entities for an Entity Type

Access the Implementation page (Set Up SACR, System Administration, Entity, Entity Type, Implementation).

Image: Implementation page

This example illustrates the fields and controls on the Implementation page. You can find definitions for the fields and controls later on this page.

Click an Entity link to access the Entity Registry page for the entity.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Setting Up Entity Registry

This section discusses how to:

- Configure entity registry.
- Set up entity properties.
- Set up entity property details.
- Create entity views.
- Generate entity schema.
- Generate entity code.
- View the entity hierarchy.

Pages Used to Set Up Entity Registry

Page Name	***Definition Name***	***Navigation***	***Usage***
Entity Registry	SCC_ENTITY_REG	Set Up SACR, System Administration, Entity, Entity Registry	Maintain the delivered entities and extend the data structure to process your transactions. Generate code or schema based on the setup data that you enter on this page. You can paste the generated code inside the application class. The application class can be used by a mechanism to process data, for example a web service. Generate schema that can be pasted inside the Integration Broker schema. Synchronize an entity.
Entity Properties	SCC_ENTITY_FIELDS	Select *Edit Properties* in the Action field on the Entity Registry page.	Review the list of property names and field names contained inside the stage record (if it exists) and the production record for the entity. Modify the default configuration if you want a specific property name to behave differently when passed inside request or response XML messages.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Page Name	***Definition Name***	***Navigation***	***Usage***
Entity Property Details	SCC_ENT_PROP_DTLS	Click the Details link on the Entity Properties page.	Maintain the label, element name and default value for a specific property. Set up list of values if needed.
Entity Registry Views	SCC_ENT_REG_VWS	Select *Edit Entity View* in the Action field on the Entity Registry page.	Create subsets of entity meta-data.
Entity Schema	SCC_ENTITY_XSD	Select *Generate XSD* in the Action field on the Entity Registry page.	View the auto-generated entity schema code. You can copy and paste this code inside the schema of an Integration Broker message.
Entity Code Generation	SCC_ER_CGEN	Select *Generate Code* in the Action field on the Entity Registry page.	Set up how you want to generate the application class code. After generating the application class code, you can paste this code inside an application class in Application Designer.
Entity Hierarchy Display	SCC_ENTITY_HRCHY	Select *View Hierarchy* in the Action field on the Entity Registry page.	Review the children hierarchy for the selected entity. **Note:** The page does not display the entity's parent relationships. Only child relationships are displayed.

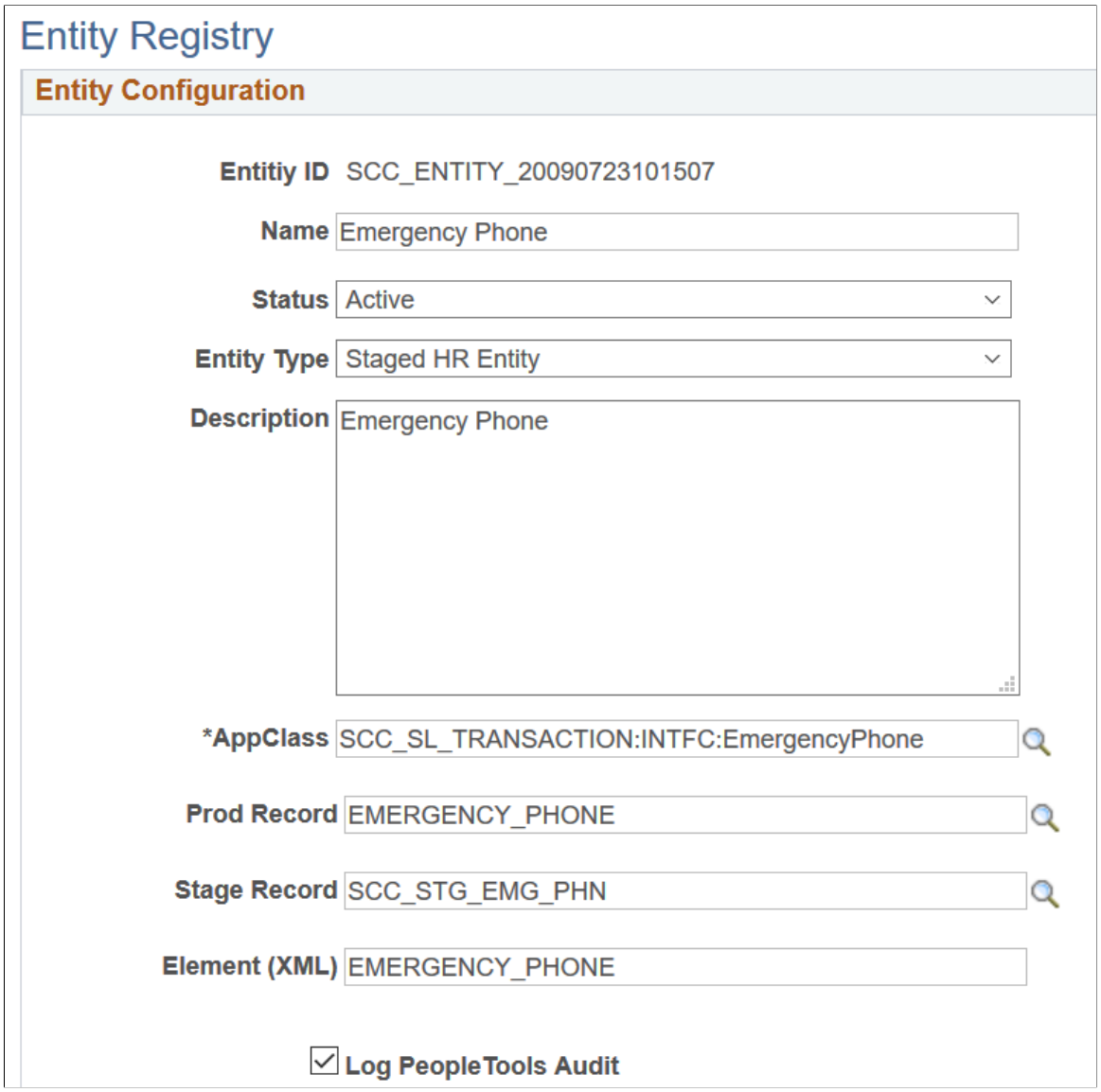
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Configuring the Entity Registry

Access the Entity Registry page (Set Up SACR, System Administration, Entity, Entity Registry).

Image: Entity Registry page (1 of 2)

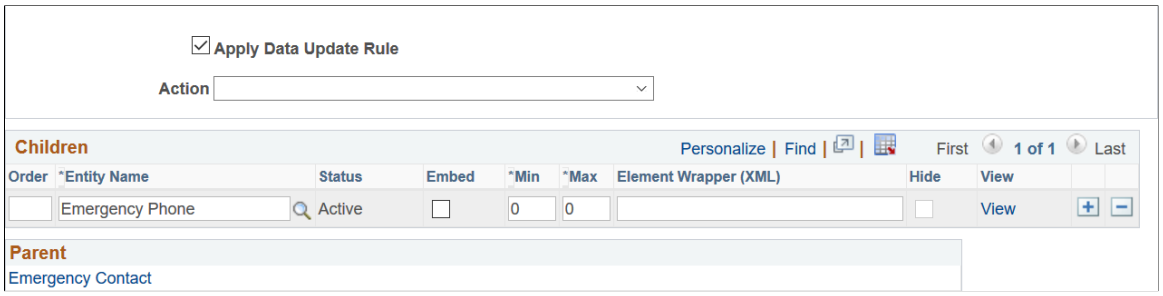
This example illustrates the fields and controls on the Entity Registry page. You can find definitions for the fields and controls later on this page.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Image: Entry Registry page (2 of 2)

This example illustrates the fields and controls on the Entry Registry page (2 of 2). You can find definitions for the fields and controls later on this page.



You can use the Entity Registry page to not only create new entities but also to maintain existing entities.

For example, you can use the page to:

- Specify that you do not want to apply a data update rule to an entity.
- Change the application class that handles the entity data.
- Change the staging or production table record.
- Require a property to be included inside the incoming or the outgoing xml message.
- Generate the schema based on specific entity settings that you can later use to configure the messages in Integration Broker.
- Generate the base code for the entity based on the configuration provided.
- Add entities as children, extending the entity tree.
- Edit entity views.
- Sync entities.

Entity Configuration

Entity ID

Displays the auto-generated unique ID.

When you add a new entity type, this field displays the value *NOID* until you save the record. When you save the new entity, the field displays the ID that the system assigned to the entity.

Name

Required. Enter the entity name.

Status

Required. Enter the entity status.

If you select *Inactive,* the system ignores the entity and its children.

Entity Type

Select the entity type. The system prompts you with the entity types defined on the Entity Type page.

See Configuring Entity Types.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Description Enter the description of what the entity is used for.

AppClass (Application Class) Specify the application class that implements the entity's logic.

Generally, this application class provides the properties, and adds any custom validation, pre-save, or fill logic. However, because it is an application class, this class can override any functionality from the base class.

The delivered base classes are stored inside the application package called: SCC_COMMON:ENTITY (for instance StagedEntity, StagedHREntity, BasicEntity, ChildEntity, and WorkEntity application classes). The application class that you enter here extends these existing application classes depending on the entity type.

Note: When creating a new entity, you can create an empty application class in Application Designer and enter its name here. Once the entity setup data is entered, you can select *Generate Code* in the Action field to generate the code that can be pasted inside the application class peoplecode.

If the entity is of a type that supports a default class the application class will not be marked required, and if no application class is specified the default will automatically be used. The default can be overridden at any time by specifying an appclass here.

Prod Record (Production Record) Enter the name of the production record where the entity data is permanently stored.

Note: This field appears only if you have indicated that the selected entity type will use production record. You indicate this on the Entity Type page.

Stage Record Enter the name of the stage record where the entity data is temporarily stored.

Note: This field appears only if you have indicated that the selected entity type will use stage record. You indicate this on the Entity Type page.

Element (XML) Enter the element name that the system will use for this entity inside the XML messages.

The name that you enter here should not match any other entity’s element name and cannot contain spaces. For example, for the delivered Emergency Contact entity name, we gave the element name EMERGENCY_CONTACT.

Log PeopleTools Audit

Bundle 48. Enhancement allows saving to PeopleTools audit records.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Select the check box to allow saving to PeopleTools audit records created to track changes to data. By default, tools auditing in entity registry is turned off.

Entity application classes with their own *populateAudit* and/or *Save* methods must add calls to these methods for auditing to work:

- Method populateAudit—add call to Abstract Entity's *populateToolsAudit();*
- Method Save—add call to Abstract Entity's *saveToolsAudit( );*

If a record's entity application class has overridden the *populateAudit* and/or *Save* methods, it must be modified to add methods that log audit data.

When a record is marked for auditing, the tools auditing feature caches the record and its field details, preventing recurring requests to the database for the record's audit setup. For that reason, you need to update or reset the entity registry cache whenever you update the audit setup of a record in the application designer. You can update the cache from the Entity Registry and Entity Property Sync pages.

Apply Data Update Rule

Select if you want to apply a data update rule to an entity. You use the Data Update Rule Entry page to define data update rules for an entity. Defining data update rules for an entity may require additional coding.

Note: SCC_COMMON.ENTITY.AbstractEntity application class handles the basic logic for data update rules. For special cases, where rules could be subdivided by type, the logic resides in the particular entity's application class.

Note: To populate an entity on the Data Update Rule Entry page, you must first select the Apply Data Update Rule check box for the entity on the Entity Registry page. Then, click the Refresh button on the Data Update Rule Entry page. Conversely, to remove an entity from the Data Update Rule Entry page, first deselect the Apply Data Update Rule check box for the entity. Then, click the Refresh button on the Data Update Rule Entry page.

See Setting Up CTM.

Action

You can select from the following actions:

- *Edit Entity View*
- *Edit Properties*
- *Generate Code*

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- *Generate XSD*
- *Sync Entity*
- *Unit Test*
- *View Hierarchy*

See the “Selecting Actions on the Entity Registry Page” section for information about these actions.

Selecting Actions on the Entity Registry Page

Edit Entity View

Select to access the Entity Registry Views page. Entity views allow the creation of subsets of entity meta-data.

Edit Properties

Select to access the Entity Properties page.

The Entity Properties page enables a finer control of entity properties through a UI controlled meta-data.

Note: This action is available only if you have indicated that the selected entity type will use properties. You indicate this on the Entity Type page.

Generate Code

Select to access the Entity Code Generation page after you complete the entity registry setup. The Entity Code Generation page enables you to configure and generate the application class code.

See Setting Up Entity Registry, “Generating the Entity Code.”.

Generate XSD

Select to access the Entity Schema page. The Entity Schema page displays the generated schema that you can paste inside an Integration Broker message schema.

Note: While you can generate schemas from any entities, it is useful only for entities that a service directly calls. In other words, it depends on what the base entity needs from a service that uses the entity. For example, the *Names* entity (which is a child of the *Constituent* entity) is not directly used. Therefore, if you extended the Names entity or make modifications to it, do not generate the XSD for *Names,* but instead generate the XSD for its root entity (that is, *Constituent*).

Note: It is useful to select this action when the root entity is modified. For instance, select this action for a parent entity, if you add or remove its children entities inside the Children grid.

See *PeopleSoft Integration Broker* for more details on schemas.

Sync Entity

Select this action to run the sync process. A pop-up appears to indicate that the sync process has completed.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The sync process:

- Updates properties to match underlying records.

  The sync process syncs the entity properties with the fields of the underlying records (stage record if it exists and production record). The sync process ties a property to each field of the underlying records. This is very important because the system uses the properties as the source of truth to drive the functionality of the underlying entity, as well as to generate the entity code and schema (not the field names or the record objects).

- Keeps entity views up to date with changes to entity properties and children.

- Keeps entity properties/children synced with the Common Attribute Framework.

You should run the sync process after creating or modifying an entity or after you have set up or changed the underlying records for the entity.

You must run the sync process before you generate any entity code or schemas. Therefore, the sync process is automatically triggered when you save the Entity Registry component or select an Action of *Generate Code* or *Generate XSD* (selecting these actions would also trigger a save) when one of the following conditions is met:

- A new entity is created.

- The stage or production record has been changed.

- The last updated datetime of either record is more recent than the last updated datetime of the entity.

The sync process is supported based on entity type. Currently, all delivered entities, except *Custom,* support syncing.

Note: The Sync Entities action is available only if you have specified a Sync AppClass for the entity type. If you have selected the Allow Properties option but did not specify a Sync AppClass on the Entity Type page, even then the system does not display the action.

Note: The sync process does not reset the settings you have entered on the Entity Properties and the Entity Property Details pages. The process only synchronizes the entities and field names with the underlying records.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Warning! If you add, remove, or rename a field of an underlying record, then all the entities using the record must be re-synchronized. The Sync process will handle property associations. After the sync, you should update the code to incorporate the changes the sync process had made. The easiest way to do this is to simply regenerate the code and re-add any customizations.

Note: Properties that are specified on the entity definition that do not exist at all in the code will be automatically handled. If the property maps to a field on the record that will be taken care of. If the property is completely new it will be stored in a hash table. Either way the safest way to access that data is by using the new methods getProperty and setProperty. Defaults all use these methods to access properties.

You can also manually update the code. To do this, remove database references from fields that are no longer part of the record. Add properties for new fields that should now be in the property list.

See Synchronizing Entity Properties.

Unit Test

Select to access the Entity Unit Tests page. The page lists the unit tests that you have created for the entity and enables you to run the unit tests.

See Executing Unit Tests for Entities.

View Hierarchy

Select to view the children hierarchy for the entity.

Children

Use the Children grid to list the entity names that are children of the main entity name. For example, in the preceding Entity Registry page screen shot, the Emergency Contact is the main entity and the Emergency Phone is a child. A parent-child relationship does not imply dependency. Each entity can exist independently and is sometimes used as such, that is, you can directly create an instance of Emergency Phone entity and use it to modify an Emergency Phone Number, without needing the parent. Entity logic is encapsulated inside the entity and is unknown from the related entities.

Warning! To create parent-child relationships, a good understanding of the data structure is required.

Order

Specify the sequence of the child entities in the XML schema.

Entity Name

Enter the name of the child entity.

The system prompts you with only the entities that are configured with an entity type defined as Entity Type Children for the parent entity type. For example, suppose on the Entity Type page, for *Staged Entity* type you have added two entity type children, *Staged Entity* and *Custom*. Now, suppose Academic Program and Academic Plan entities have an entity

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

type of *Staged Entity* on the Entity Registry page. Two other entities - Academic Interests entity has an entity type of *Custom* and Prospect Program entity has entity type of *Basic Entity* on the Entity Registry page. Therefore, on the Entity Registry page you can add Academic Plan and Academic Interests as child entities for the Academic Program entity. However, you cannot add Prospect Program entity as a child for the Academic Program entity.

See Configuring Entity Types.

Status

Displays the status of the child entity.

Embed

Specify how you want the XML messages to display the child entity data with the parent entity data.

Select the Embed check box if you want the child entity data to be embedded in the parent entity data. This is useful for certain sibling record purposes where you do not want the XML to show the child at a separate level from the parent.

- Example when non-embedded (that is, you do not select the Embed check box):

```
<Entity>
    <Data1/>
    <Data2/>
    <Entity2>
         <Data3/>
    </Entity2>
</Entity>
```

- Example when embedded (that is, you select the Embed check box):

```
<Entity>
       <Data1/>
       <Data2/>
       <Data3/>
</Entity>
```

Min

Enter the minimum number of occurrences of this entity under the parent, which can be the number 0 or any number greater than 0.

Max

Enter the maximum number of occurrences of this entity under the parent.

Note: The number 0 means the maximum is unbounded, or any number greater than the min value can be used.

Element Wrapper (XML)

Optional. Enter a value that the system can use as a tag to wrap all the child occurrences of this entity under the parent.

- Example when Element Wrapper (XML) is left blank (no wrapper):

```
<Entity>
    <Data1/>
```

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

```
    <Data2/>
    <Entity2>
        <Data3/>
    </Entity2>
    <Entity2>
        <Data3/>
    </Entity2>
    <Entity3>
        <Data4/>
    </Entity3>
</Entity>
```

- Example when Element Wrapper (XML) is defined (with wrapper):

```
<Entity>
    <Data1/>
    <Data2/>
    <Wrapper>
        <Entity2>
            <Data3/>
        </Entity2>
        <Entity2>
            <Data3/>
        </Entity2>
    </Wrapper>
    <Entity3>
        <Data4/>
    </Entity3>
</Entity>
```

View Click to access the Entity Registry page for the child entity.

Parents

This region displays the parent entities. More than one parent can exist. Click the link to access the Entity Registry page for the parent.

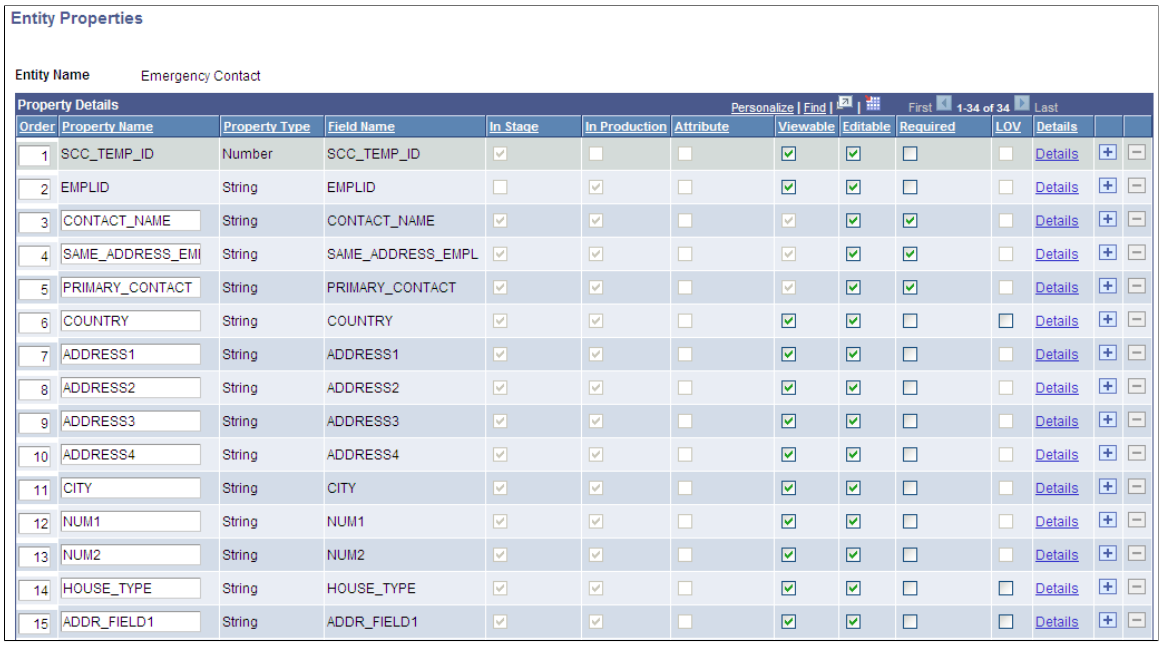
 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Setting Up Entity Properties

Access the Entity Properties page (select *Edit Properties* in the Action field on the Entity Registry page).

Image: Entity Properties page (1 of 2)

This example illustrates the fields and controls on the Entity Properties page (1 of 2). You can find definitions for the fields and controls later on this page.



Entity Properties

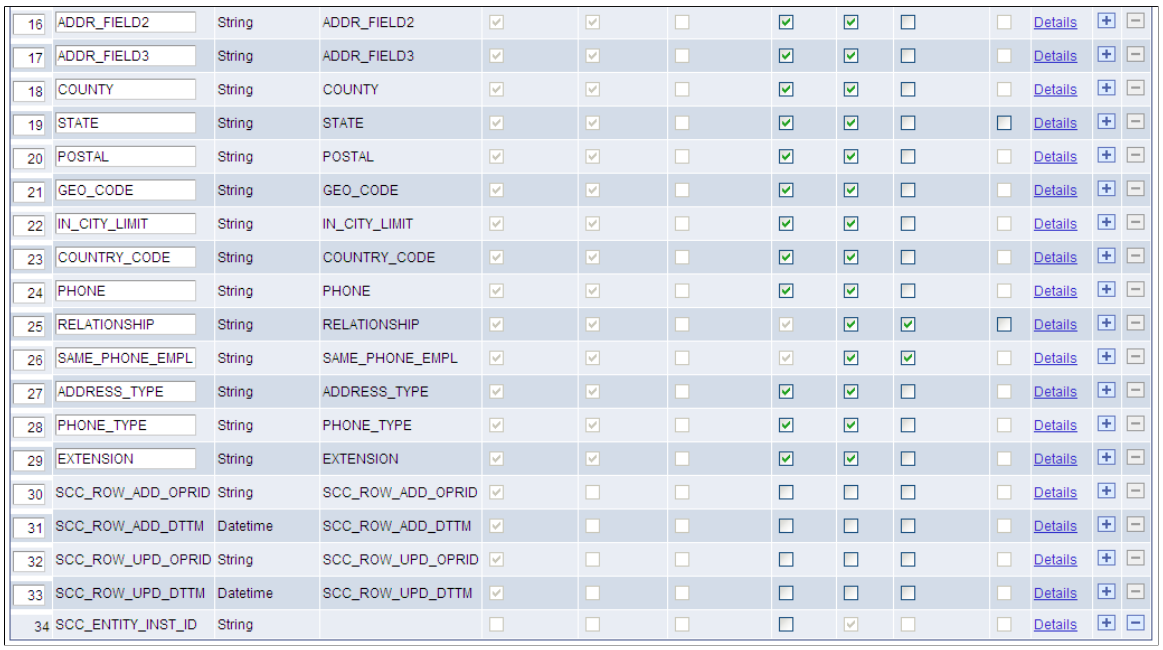
Entity Name Emergency Contact

Property Details — Personalize | Find | First 1-34 of 34 Last

Order	Property Name	Property Type	Field Name	In Stage	In Production	Attribute	Viewable	Editable	Required	LOV	Details
1	SCC_TEMP_ID	Number	SCC_TEMP_ID	✓			✓	✓			Details
2	EMPLID	String	EMPLID		✓		✓	✓			Details
3	CONTACT_NAME	String	CONTACT_NAME	✓	✓		✓	✓	✓		Details
4	SAME_ADDRESS_EMI	String	SAME_ADDRESS_EMPL	✓	✓		✓	✓	✓		Details
5	PRIMARY_CONTACT	String	PRIMARY_CONTACT	✓	✓		✓	✓	✓		Details
6	COUNTRY	String	COUNTRY	✓	✓		✓	✓			Details
7	ADDRESS1	String	ADDRESS1	✓	✓		✓	✓			Details
8	ADDRESS2	String	ADDRESS2	✓	✓		✓	✓			Details
9	ADDRESS3	String	ADDRESS3	✓	✓		✓	✓			Details
10	ADDRESS4	String	ADDRESS4	✓	✓		✓	✓			Details
11	CITY	String	CITY	✓	✓		✓	✓			Details
12	NUM1	String	NUM1	✓	✓		✓	✓			Details
13	NUM2	String	NUM2	✓	✓		✓	✓			Details
14	HOUSE_TYPE	String	HOUSE_TYPE	✓	✓		✓	✓			Details
15	ADDR_FIELD1	String	ADDR_FIELD1	✓	✓		✓	✓			Details

Image: Entity Properties page (2 of 2)

This example illustrates the fields and controls on the Entity Properties page (2 of 2). You can find definitions for the fields and controls later on this page.



16	ADDR_FIELD2	String	ADDR_FIELD2	✓	✓		✓	✓			Details
17	ADDR_FIELD3	String	ADDR_FIELD3	✓	✓		✓	✓			Details
18	COUNTY	String	COUNTY	✓	✓		✓	✓			Details
19	STATE	String	STATE	✓	✓		✓	✓			Details
20	POSTAL	String	POSTAL	✓	✓		✓	✓			Details
21	GEO_CODE	String	GEO_CODE	✓	✓		✓	✓			Details
22	IN_CITY_LIMIT	String	IN_CITY_LIMIT	✓	✓		✓	✓			Details
23	COUNTRY_CODE	String	COUNTRY_CODE	✓	✓		✓	✓			Details
24	PHONE	String	PHONE	✓	✓		✓	✓			Details
25	RELATIONSHIP	String	RELATIONSHIP	✓	✓		✓	✓	✓		Details
26	SAME_PHONE_EMPL	String	SAME_PHONE_EMPL	✓	✓		✓	✓	✓		Details
27	ADDRESS_TYPE	String	ADDRESS_TYPE	✓	✓		✓	✓			Details
28	PHONE_TYPE	String	PHONE_TYPE	✓	✓		✓	✓			Details
29	EXTENSION	String	EXTENSION	✓	✓		✓	✓			Details
30	SCC_ROW_ADD_OPRID	String	SCC_ROW_ADD_OPRID	✓							Details
31	SCC_ROW_ADD_DTTM	Datetime	SCC_ROW_ADD_DTTM	✓							Details
32	SCC_ROW_UPD_OPRID	String	SCC_ROW_UPD_OPRID	✓							Details
33	SCC_ROW_UPD_DTTM	Datetime	SCC_ROW_UPD_DTTM	✓							Details
34	SCC_ENTITY_INST_ID	String						✓			Details

The Entity Properties page enables you to define how a property should behave when its data is encoded as XML (for instance, when the system uses the property inside web services). A property is defined

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

as an individual value on the entity that is controlled by the Entity Registry component. Properties are generally based on the fields of the underlying records (that is, staging record if it exists and production record). However, properties can exist that are independent of the underlying records. We refer to these independent properties as transient properties. You must write custom code to save or populate the transient properties. Entities can have an unlimited number of properties.

The system uses the properties, and not the fields of the underlying records, as the source of truth to drive the functionality of the underlying entity, and to generate the entity code and the schema.

Order

Order the properties as you want then to be displayed inside an XML message.

Property Name

Indicates the property name on the entity object.

By default, the property name matches the field name. You can change a property name, unless the property is already handled by the base application class, which causes the property to be read-only.

Note: You cannot delete property names that match the fields in the staging record (if it exists) or in the production record.

Note: Inside the delivered entities, the property name SCC _ENTITY_INST_ID is listed even though it is not included inside the underlying records (we refer to these extra properties as transient properties). The primary use for this transient property is to provide a unique identifier for a particular instance of an entity. When an error occurs, this ID enables the system to link an error to a specific entity instance.

Warning! If you make any changes to the property name after the code has been generated, then you must either regenerate the code or manually update the property names to match the new name.

For information on how to use property name SCC_ENTITY _INST_ID, see "User Interface Considerations" (PeopleSoft Campus Solutions 9.2: Recruiting and Admissions), “Error Handling.”

See Configuring Entity Types.

Property Type

Indicates whether the property is a *String, Number, Date, Datetime,* or *Time.*

If the property is based on a field, the system automatically determines the property type based on the field type. You cannot edit such a property type.

Note: The Property Type field is enabled only when the entity is set up with the entity type *Custom* or when you add a new property name (transient property).

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Note: When the system triggers the Sync process, the process makes sure that the Property Type displayed for each property on this page matches the field type changes that may have occurred in the underlying records.

Warning! If you make any changes to the property type after the code has been generated, then you must either regenerate the code or manually update the property data types to match the new type.

Field Name

Displays the name of the matching field inside the underlying records. Only the sync process can enter and maintain this value.

Note: A Field Name value appears only when the property is field based.

Note: The sync process ensures that the correct Field Name value appears.

In Stage

Indicates whether the Field Name is included in the stage record. Only the sync process can select or deselect this check box.

Note: This field appears only if you have specified a stage record on the Entity Registry page.

Note: The sync process verifies whether the field is still included inside the stage record. If this property changes, you will need to change the code in the get and set methods to match the changes.

Example code for a property that is only in stage record:

```
get PropertyName
   /+ Returns String +/
   If %This.STAGE_MODE Then
      Return %This.data. PropertyName.Value;
   End-If;
   Return "";
end-get;

set PropertyName
   /+ &NewValue as String +/
   If %This.STAGE_MODE Then
      %This.data. PropertyName.Value = &NewValue;
   End-If;
end-set;
```

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Warning! If you remove a field from both production and stage records, the Entity Properties page will still display the property. Properties are never removed and have to be manually deleted, to avoid changes to schema. Notice that after you manually remove the property in the code, the system enables the minus sign in the grid to allow you to delete the property.

In Production

Indicates whether the Field Name is included in the production record. Only the sync process can select or deselect this check box.

Note: The sync process verifies whether the field is still included inside the production record. If this property changes, you will need to change the code in the get and set methods to match the changes.

Note: This field appears only if you have specified a production record on the Entity Registry page.

Warning! If you remove a field from both production and stage records, the Entity Properties page will still display the property the Entity Properties page. Properties are never removed and have to be manually deleted, to avoid changes to schema. Notice that after you manually remove the property in the code, the system enables the minus sign in the grid to allow you to delete the property.

Example code for a property that is only in production record:

```
get PropertyName
   /+ Returns String +/
   If not %This.STAGE_MODE Then
      Return %This.data. PropertyName.Value;
   End-If;
   Return "";
end-get;

set PropertyName
   /+ &NewValue as String +/
   If not %This.STAGE_MODE Then
      %This.data. PropertyName.Value = &NewValue;
   End-If;
end-set
```

Attribute

Attributes from the Common Attribute Framework can be accessed as properties on an entity. This allows attributes to be treated more like fields on a record.

See the “Common Attribute Framework Integration” documentation later in this section.

Viewable and **Editable**

When you select the Viewable check box, the system adds the property to any generated outgoing XML, JSON or Rules Engine message that includes the entity (the system does this

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

using the toXMLNode method). This outgoing message from a system can either be a request message or a response message.

When you select the Editable check box, the system parses the property from any incoming XML, JSON or Rules Engine message that it receives (the system does this using the fromXMLNode method). If the Editable check box is not selected, the system will ignore any incoming data.

Note: While entities are used for services, it is not their only use. Options such as Viewable and Editable have implications outside of services (for instance, importing and exporting XML, JSON or Rules Engine data). Therefore, while request and response messages do make use of the Viewable and Editable options, these messages are not the only usage scenarios for the options.

Note: If you do not select either the Viewable check box or the Editable check box, the property will not appear in the schema.

Note: The values for transient properties are not stored, but the system can include these values inside the messages. Because at configuration time the sync process cannot tell if it should have an incoming or outgoing value or not, the Viewable and Editable options are always valid for transient properties.

Note: The sync process, by default, selects the Viewable and Editable check boxes for every field except for fields that the system has programmatically marked to be ignored based on the entity type (for instance SCC_ROW_ADD_OPRID, SCC_ROW_ADD_DTTM, SCC_ROW_UPD_OPRID, and SCC_ROW_UPD_DTTM). You can manually select the Viewable and Editable check boxes for these ignored fields.

Required

Indicates if the field is required. If the check box is selected, the system must populate the data into the field. The system will check whether the data exists for the field inside the XML or JSON messages during validation.

When a property is marked as required, the system automatically selects and disables the Viewable check box. Selecting a field as required means it will be marked minOccurs 1 in the schema, that is, the field must be included in any occurrence of the XML.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Note: For entities configured with an entity type that uses a stage record, the sync process, by default, selects the Required check box if the field is included in both production and stage records. If the field is only included in one of the underlying records, even it if is required in the record definition, the sync process does not mark the field as required. This is because the field may not be required in all circumstances.
For entities configured with an entity type that does not use a stage record, the sync process, by default, selects the Required check box if the field is marked as required inside the production record.

LOV (List of Values)

Select if you want to send the value description, with the value in the outgoing XML (the Viewable check box must be selected).

The system enables this check box for only the fields that have list of values, that is field names defined with a prompt table or field names defined as translate fields inside the record definition in Application Designer. The system includes the description inside the outgoing XML message under a new tag labeled <*name of the property Element* (XML)_LOVDescr>.

By default, this option uses the following logic to return the description:

- For *translate fields:* the value's long description is returned.
- For *prompt fields:* the logic looks for the existence of the following fields in the record (respecting the order given): DESCR, DESCRSHORT, DESCR30, DESCR100. If none of these works, it uses the first non-key field marked as a list box item.

For example, in the Emergency Contact entity, the production record is SCC_EMERG_CNT_2. If you want to include a description in the outgoing XML message for the COUNTRY property (which is a field with a prompt table), selecting the LOV check box will result in including the following tag inside the outgoing XML message: <COUNTRY_LOVDescr>, where COUNTRY is the Element (XML) name defined for the COUNTRY property. Suppose the COUNTRY value gets populated with *USA*. The LOVDescr tag will then contain: <COUNTRY_LOVDescr>United States</COUNTRY_LOVDescr>. The value *United States* corresponds to the DESCR field found inside the prompt table (COUNTRY_TBL) defined inside the SCC_EMERG_CNT_2 record for the COUNTRY field name.

If you do not want to use the default logic, click the Details link and set the LOV Unique ID field on the Entity Property Details page.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Note: The sync process enables the LOV check box only if the field is defined with a prompt table or defined as a translate field inside the record definition. The sync never selects or deselects the check box. Only you can manually select or deselect this check box.

Note: The default logic returns the field description from the record that is being used. Therefore, if the service operation is using the stage record, it will evaluate the record definition for the stage record. If instead the operation uses the production record, it will use the field description stored inside the record definition for the production record.

Details Click to access the Entity Property Details page.

Last Synced Shows the last time this entity was synced. If the entity has not been synced since the last synced tracking was introduced, this date will not appear. Clicking the date time link will show the detail log for the sync. This log is identical to the one seen when viewing the details from the Entity Property Sync component.

Common Attribute Framework Integration

Attributes from the Common Attribute Framework are treated as entity properties. By adding the properties and syncing the entity the new attributes appear as properties.

Attribute types map to equivalent entity property types:

Attribute Type	*Property Type*
Short Text, Text, Long Text, Yes/No, LOV	String
Number	Number
Date	Date
Time	Time

Repeatable maps to the list entity property list attribute.

If the Attribute Type is a *list of values* the Property LOV Unique ID is set.

For each attribute record a child entity is created. It is automatically added as a child. It is maintained and should never be accessed directly. This entity will never show up in XML/JSON and will not be accessible in the rules engine.

See Understanding Common Attribute Framework.

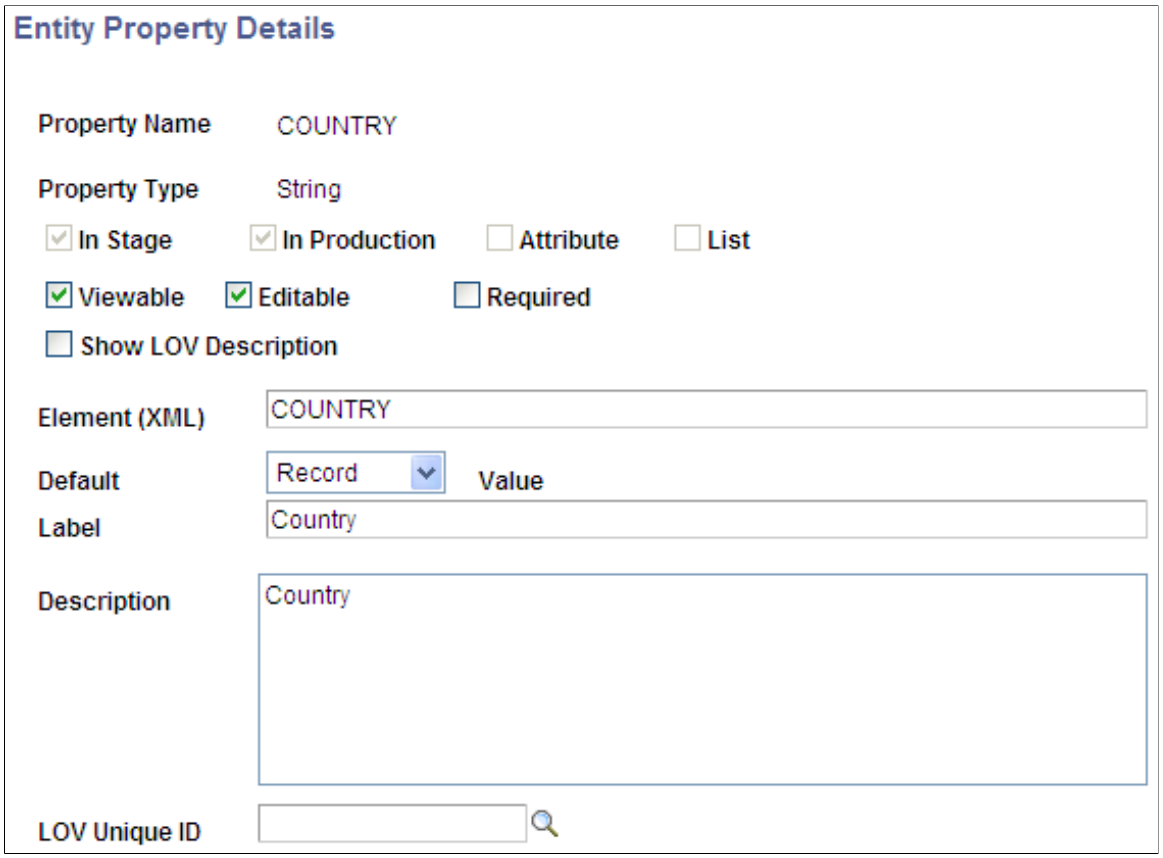
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Setting Up Entity Property Details

Access the Entity Property Details page (click the Details link on the Entity Properties page).

Image: Entity Property Details page

This example illustrates the fields and controls on the Entity Property Details page. You can find definitions for the fields and controls later on this page.



List

If this check box is selected, the property is treated as a list. This supports repeatable attributes. Non-field based properties can also be marked as lists, in which case they are treated as arrays in the entity application class.

Show LOV Description

This check box is the same as the LOV check box that appears on the Property Details grid of the Entity Properties page. When one is selected, the other one is also selected. They both have the same purpose.

Element (XML)

Indicates the label for the XML tag name used for this property. This is the tag that the system uses in the XML schema of the incoming and outgoing XML messages.

Warning! If you change this value, you must select an Action of *Generate XSD* on the Entity Registry page to regenerate the schema and paste it inside the Integration Broker message schema.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Note: The sync process, by default, sets this value to match the Field Name.

Default

Define the default value. There are three types of defaults:

- *Record:* Uses the default value defined in the record definition in Application Designer for the underlying production record. This is the selected choice by default.
- *Custom:* If you select *Custom*, the Value field appears. Use the Value field prompt to select a default value that will be included inside the outgoing XML.
- *Skip:* Skips the property at the time of populating default values.

Note: The sync process always sets this value to *Record.*

Label

Indicates the label for the property as shown in the comments for the XML schema. This is primarily a descriptive attribute.

Note: If you want to see the new label in the schema, regenerate the schema.

Note: The sync process sets this value with the default label defined in the field properties of the production record.

Description

Indicates the description for the property as shown in the comments for the XML schema. This is primarily a descriptive attribute.

Note: If you want to see the new description in the schema, regenerate the schema.

Note: The sync process sets this value with the description defined for the field definition in Application Designer.

LOV Unique ID

This field allows you to select a specific LOV prompt definition for the entity property to use.

Creating Entity Views

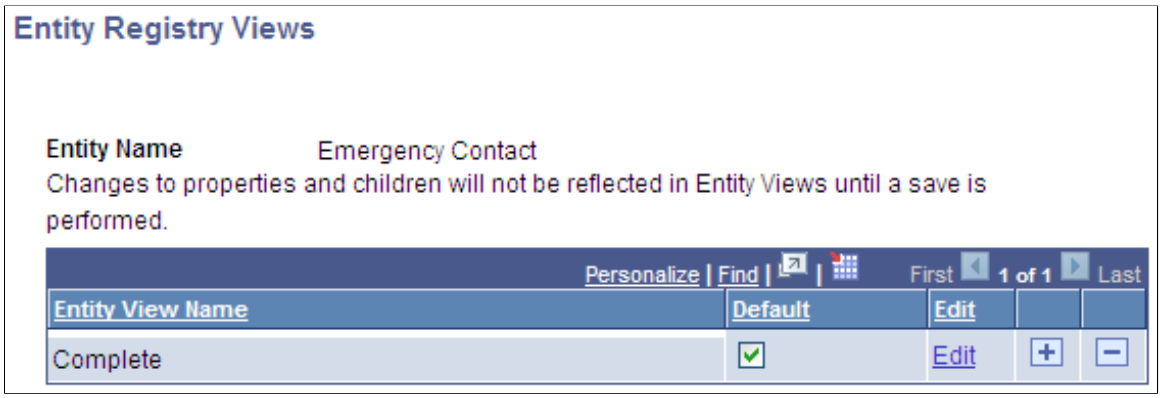
Entity views allow the creation of subsets of entity meta-data. Entity views can only reduce access from what is specified at the base entity. So if a property is set to Viewable but not Editable, you cannot set it to Editable from a view, but Viewable can be turned on or off.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Access the Entity Registry Views page (select *Edit Entity View* in the Action field on the Entity Registry page).

Image: Entity Registry Views page

This example illustrates the fields and controls on the Entity Registry Views page. You can find definitions for the fields and controls later on this page.



Every entity has a “Complete” view. The complete view always matches what has been set on the entity property and children. This view cannot be edited and is maintained during the sync process. A single view, per entity, is always marked default. The default view is used if no profile is selected, or if that entity does not belong to the current profile. Default also specifies the view assigned when a new entity is added to a profile. The default is used by all entity based services built prior to profiles existing, such as EWS and AAWS.

See Setting Up Entity Profiles.

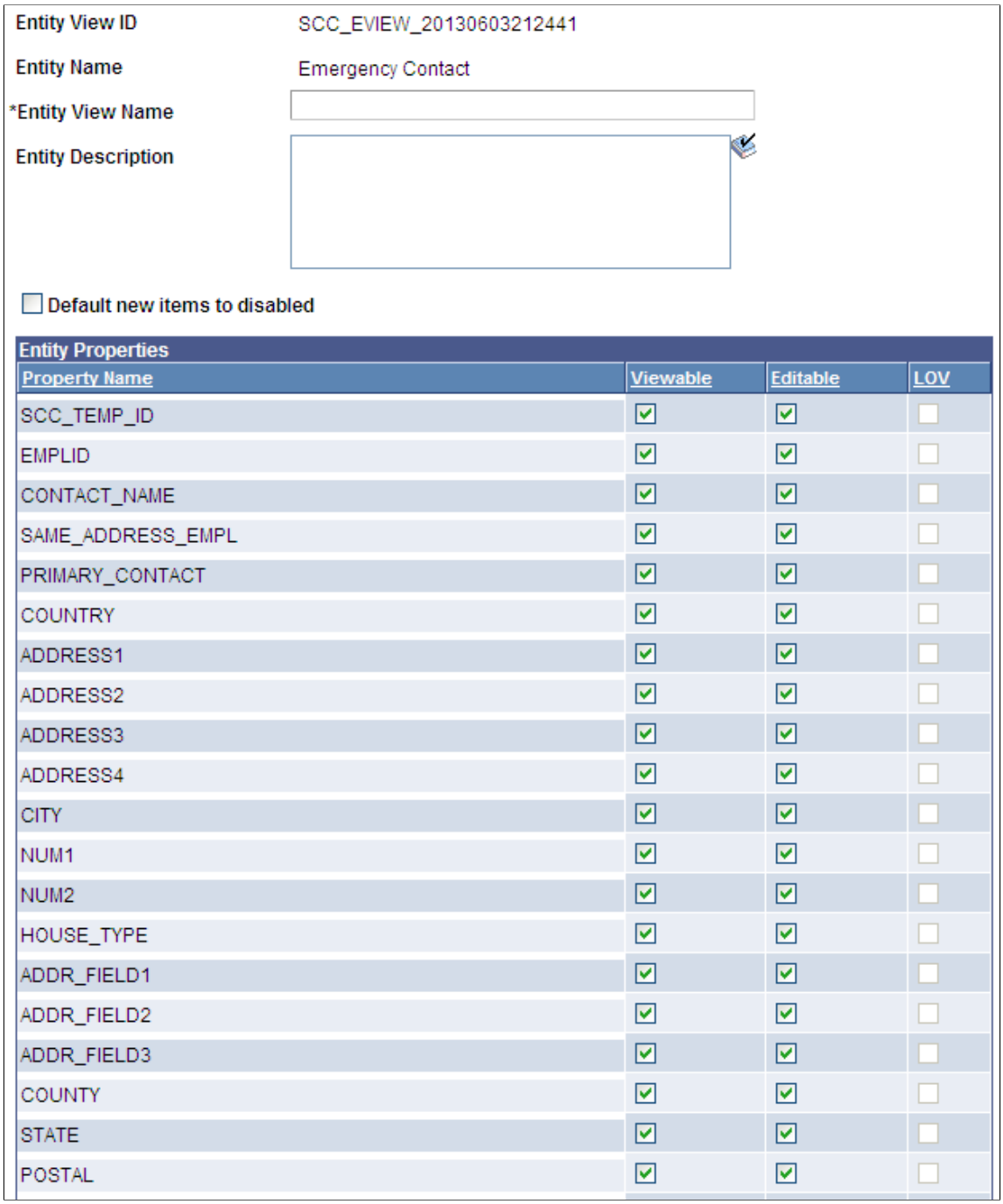
The sync process maintains the entity view matching the entity properties and children. No data is currently logged for entity view syncs.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To create a new entity view, add a row on the Entity Registry Views page. You are taken to the Entity Registry View Details (SCC_ENT_VW2) page:

Image: Entity Registry View Details page (1 of 2)

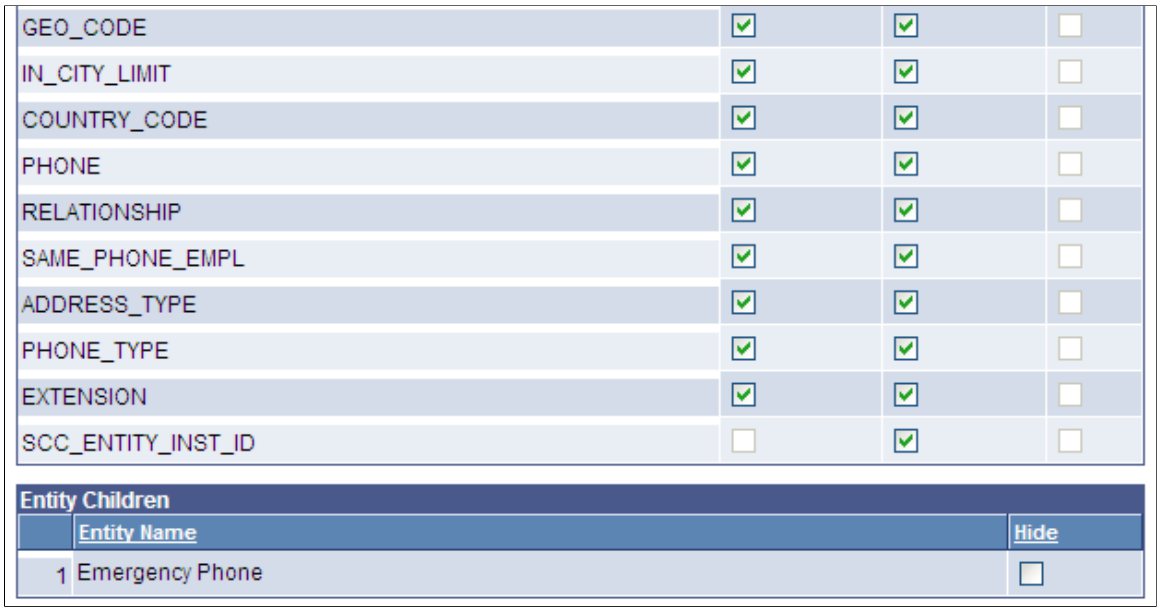
This example illustrates the fields and controls on the Entity Registry View Details page (1 of 2). You can find definitions for the fields and controls later on this page.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Image: Entity Registry View Details page (2 of 2)

This example illustrates the fields and controls on the Entity Registry View Details page (2 of 2). You can find definitions for the fields and controls later on this page.



Entity Properties

Viewable and **Editable**

You can select or deselect these check boxes on a per property/view basis.

LOV

You can select or deselect this check box on a per property/view basis, but only if the LOV check box is available and selected for the property.

Entity Children

Hide

Select this check box to hide entity children. The entities that are a direct child of the current entity (and any children of those entities) will not be shown.

Note: Selecting this check box just hides the entity child; it does not prevent that child from being filled as part of the initial fill process. This avoids certain complications in the rules engine.

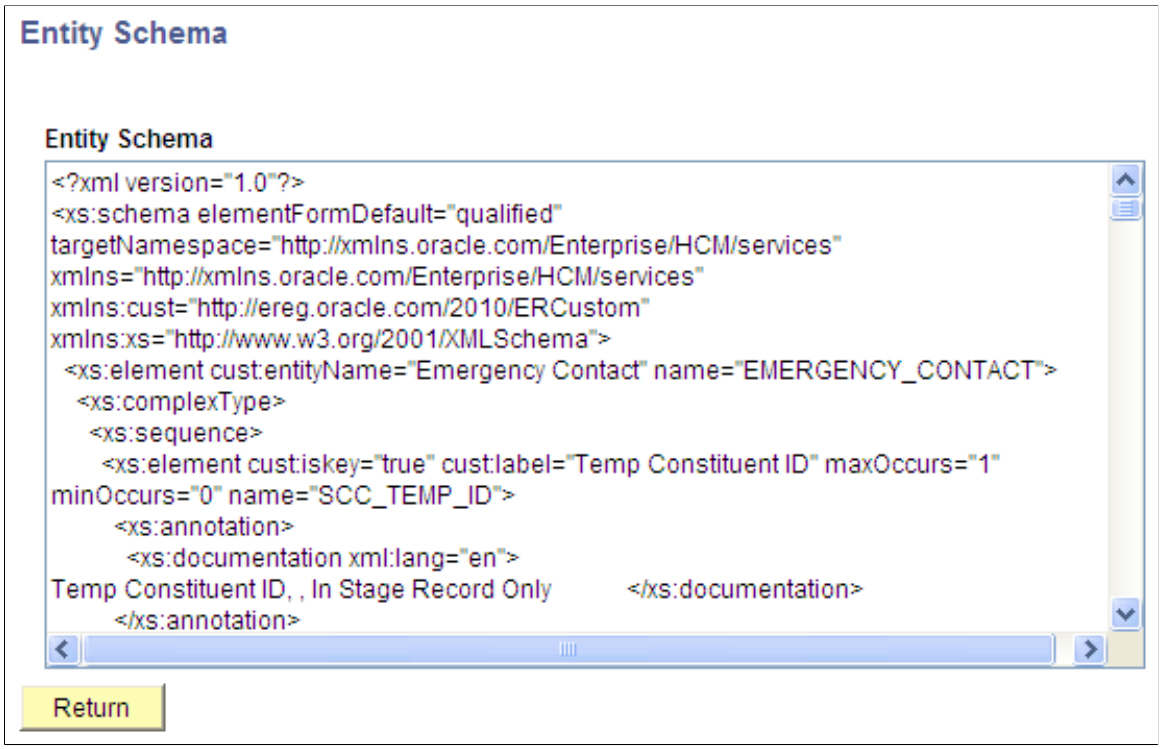
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Generating the Entity Schema

Access the Entity Schema page (select *Generate XSD* in the Action field on the Entity Registry page).

Image: Entity Schema page

This example illustrates the fields and controls on the Entity Schema page. You can find definitions for the fields and controls later on this page.



The system generates the schema based on the property settings. If the following settings are changed on the Entity Properties and Entity Property Details pages, you must regenerate the schema:

- Element (XML)
- Show LOV Description and LOV check box
- In Stage
- In Production
- Required
- List

Any changes made to the following property settings do not require regenerating the schema as their function is primarily descriptive:

- Label
- Description

You can paste the generated schema to the appropriate message name inside the Schema page (PeopleTools, Integration Broker, Integration Setup, Messages, Schema).

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

For information on schemas, see *PeopleTools: Integration Broker*.

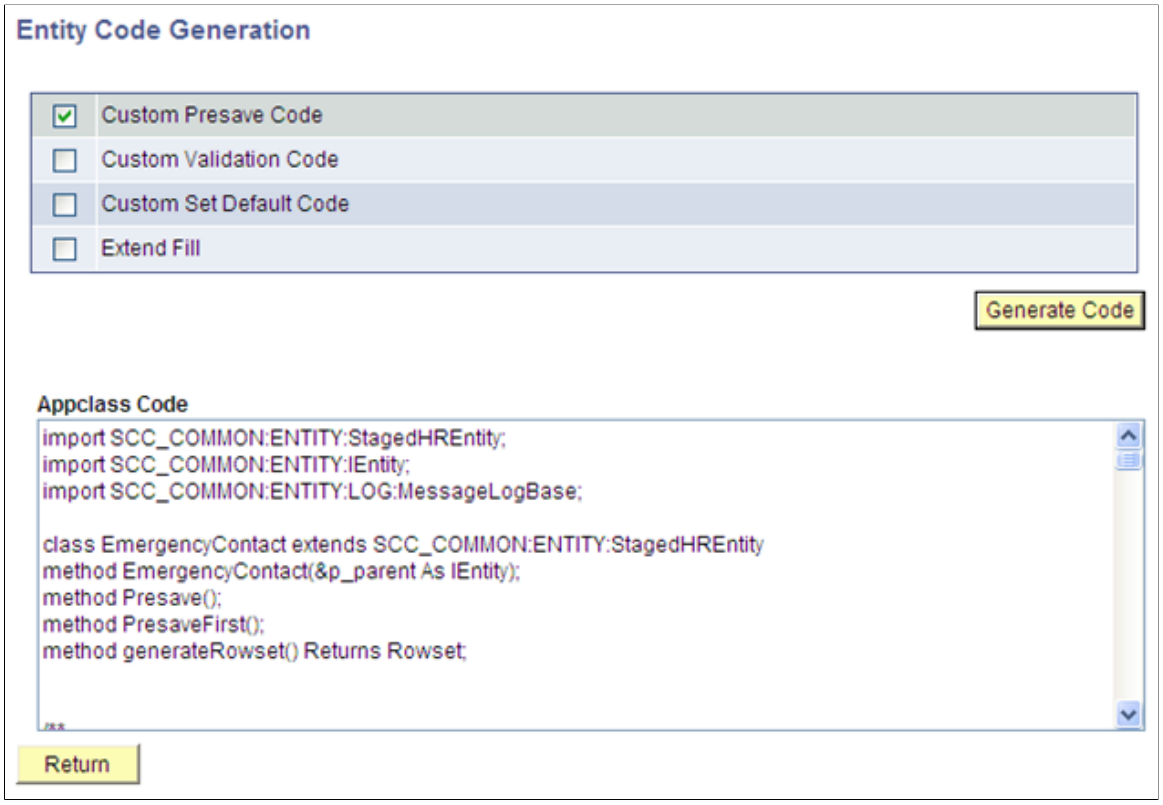
Generating the Entity Code

Access the Entity Code Generation page (select *Generate Code* in the Action field on the Entity Registry page).

Note: This should only be done for entities where the application class is specified. Entities using the default application class approach have no need to do this.

Image: Entity Code Generation page

This example illustrates the fields and controls on the Entity Code Generation page. You can find definitions for the fields and controls later on this page.



The system generates the code based on the property settings. If the following settings are changed on the Entity Properties and Entity Property Details pages, you must regenerate the code:

- Property Name
- Property Type
- In Stage
- In Production
- Attribute
- List

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The four options that the grid displays are dynamic, based on the entity type and are coded in the Code Generation Appclass. Therefore, while these options are available in most of the delivered entity types, they are not static.

Custom Presave Code

Select to add the methods *preSave* and *preSaveFirst* to the generated code. These methods are called as part of the preSave process.

- *preSave:* This is called prior to the save process to do any final data processing.
- *preSaveFirst:* This is called once per childEntity before preSave. This is useful if you have code that needs to be run across all child entities of a particular type, or you need to remove entities prior to saving.

Custom Validation Code

Select to add the methods *Validate* and *validateFirst* to the generated code.

- *Validate:* Runs as part of the validation process for this entity. Generally, this method should call commonValidate, which is responsible for field level validations. Any custom validation can also be added.
- *validateFirst:* This is called once per childEntity, before any other validation. This method is good for running validations that may need to compare entities against each other that you do not need to run during the validation for every entity.

Custom Set Default Code

Select to add an override for *setDefault.* Normally, *setDefault* is based on the field defaults specified on the record. It is recommended to first call %Super.setDefault() to set the default field values and then set any special overrides.

Extend Fill

Select to override the method called *Fill.* The system runs this method when it takes data from a record and populates this data into an entity. You may need to override this method to modify data after retrieval from the database or to populate transient values.

Generate Code

Click to generate the code template that can later be pasted directly inside the Application Class that you have specified on the Entity Registry page. Use Application Designer to paste the generated code.

To generate the code, the system uses the staging record (if it exists) and the production record, the entity type, the entity property settings, and the selected code generation options.

When you click the button, the sample code template is displayed inside the Appclass Code box.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

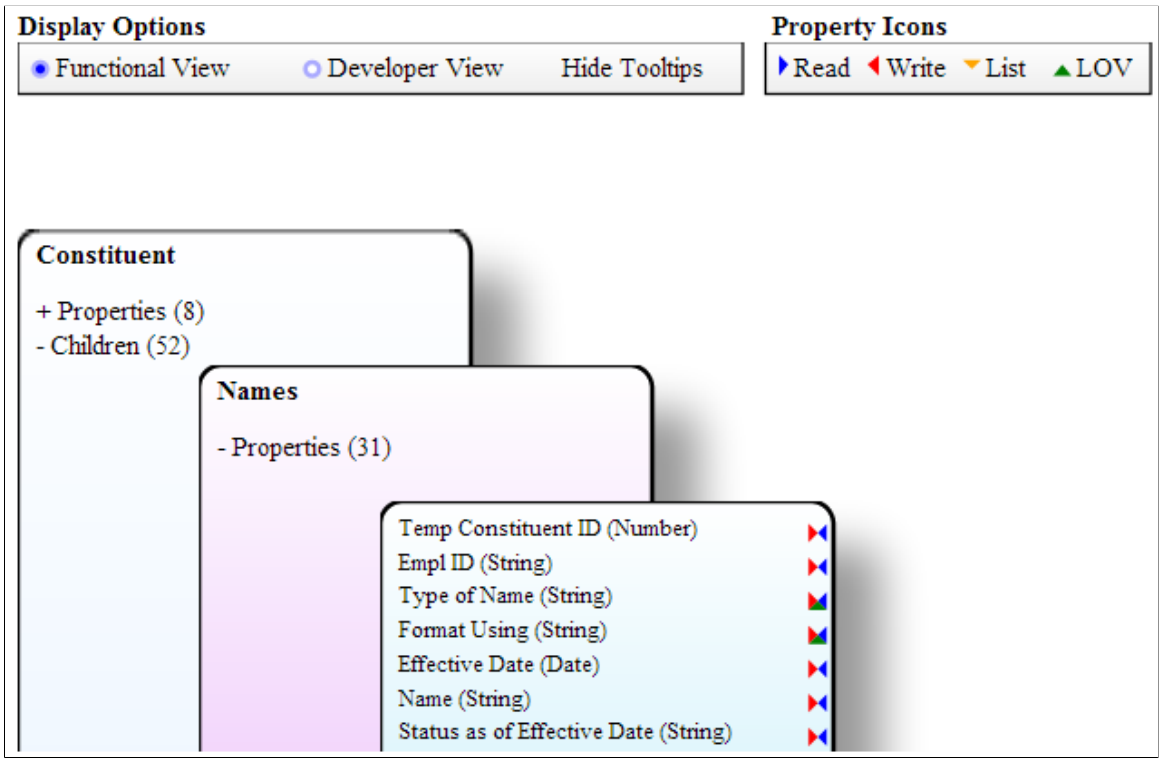
Note: Clicking this button is useful at the time of creating a new entity or if a change is made to an existing entity. For example, you could generate code when production or stage records are modified, or when properties are changed.

Viewing the Entity Hierarchy

To view the children hierarchy for an entity, select the View Hierarchy action on the Entity Registry page. Here is an example of an entity hierarchy:

Image: Entity Hierarchy example

This example illustrates the fields and controls on the Entity Hierarchy example. You can find definitions for the fields and controls later on this page.

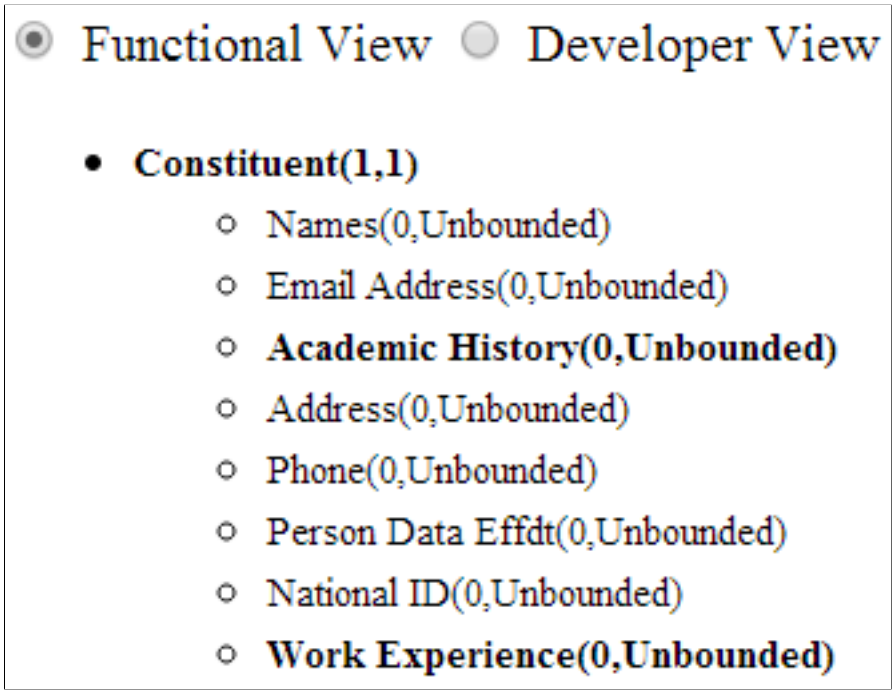


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The hierarchy is dynamically drawn using HTML5 SVG. You can download the SVG, but only in Chrome and Firefox (not in Internet Explorer). IE8 does not support HTML5, so it shows an alternative list view. All other browsers can also view this using the View List link:

Image: Entity Hierarchy example list view

This example illustrates the fields and controls on the Entity Hierarchy example list view. You can find definitions for the fields and controls later on this page.



Display Options

Functional View	A view targeted at functional users. Shows entity names and property labels.
Technical View	Targeted at developers. Adds Entity ID. Shows hidden properties and children (hidden children are transparent).
Hide Tooltips	Allow or disallow tooltips. Tooltips appear over every entity name, property name, child link and property link to give extra details. Only available in SVG mode.

SVG View

Click on properties(x) or children(x) to expand details.

List View

You can click on entities that appear in bold and these entities have children.

Whether child entities appear as expanded or collapsed is retained when switching back and forth between SVG and List View.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Setting Up Entity Profiles

Entity Profiles allow entities and views to be combined to create groupings of entities and views.

This section discusses how to configure entity profiles.

Page Used to Set Up Entity Profiles

Page Name	*Definition Name*	*Navigation*	*Usage*
Entity Profile	SCC_ENTITY_PRFL	Set Up SACR, System Administration, Entity, Entity Profile	Combine entities and views to create groupings of entities and views.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Configuring Entity Profiles

Access the Entity Profile page (Set Up SACR, System Administration, Entity, Entity Profile).

Image: Entity Profile page (1 of 2)

This example illustrates the fields and controls on the Entity Profile page (1 of 2). You can find definitions for the fields and controls later on this page.

Entity Profile ID: SCC_EPRFL_20121206025640

Entity Profile Name: Mobile

Profile Description Profile for Mobile

***Profile Type** Web Service

Parent Profile

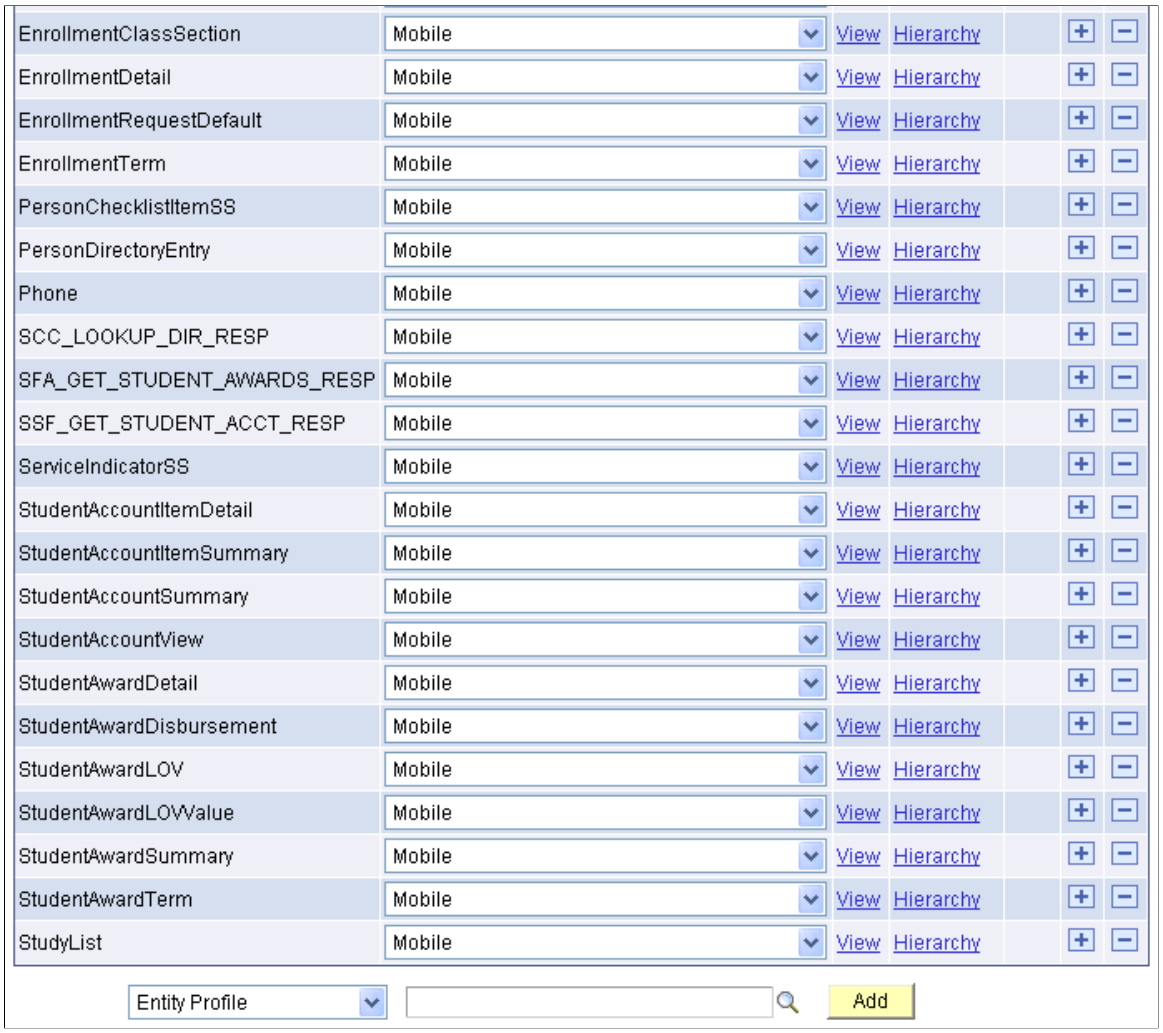
Personalize | Find | First 1-44 of 44 Last

Entity Name	*Entity View	View	Hierarchy
Address	Mobile	View	Hierarchy
ClassComponent	Mobile	View	Hierarchy
ClassSearchResult	Mobile	View	Hierarchy
ClassSearchSubject	Mobile	View	Hierarchy
ClassSearchSummary	Mobile	View	Hierarchy
ClassSection	Mobile	View	Hierarchy
ClassSectionCombined	Mobile	View	Hierarchy
ClassSectionComponent	Mobile	View	Hierarchy
ClassSectionEnrlInfo	Mobile	View	Hierarchy
ClassSectionInstructor	Mobile	View	Hierarchy
ClassSectionMeetingPattern	Mobile	View	Hierarchy
ClassSectionOptions	Mobile	View	Hierarchy
ClassSectionResult	Mobile	View	Hierarchy
ClassSectionTextbook	Mobile	View	Hierarchy
ClassSectionTextbookDetails	Mobile	View	Hierarchy
CourseAttributesDetail	Mobile	View	Hierarchy
CourseOfferingDerived	Mobile	View	Hierarchy
CourseOfferingDetail	Mobile	View	Hierarchy
CourseOfferingDetailResult	Mobile	View	Hierarchy
CourseOfferingTerm	Mobile	View	Hierarchy
CourseShoppingCart	Mobile	View	Hierarchy
Email Address	Mobile	View	Hierarchy

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Image: Entity Profile page (2 of 2)

This example illustrates the fields and controls on the Entity Profile page (2 of 2). You can find definitions for the fields and controls later on this page.



Entity Profile ID Auto-generated unique ID for the profile.

Entity Profile Name and **Entity Profile Description** Enter a name and description for the profile.

Profile Type

- *Category:* This type will be used to make it easier to search for entities.

  For category there is no entity view, since it is not used to specify display.

- *Data Set Category:* This profile type is similar to Category, but is restricted to entities of the type "Rules Engine Data Set". Data set categories are created here but entities can be added in the Define Data Sets component of the Rules Engine setup.

  See Understanding the Rules Engine.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- *Rules Engine:* This profile type allows you to restrict what properties and entity children are available based on the entity view. The available base entities for a rule category are also set here. Any entity that is not specified as a base entity is not selectable as such when generating a rule in a rule category tied to this entity profile. Any entity that is not specified here is still accessible from the Rules Engine, using the default profile, if it is accessible from the entity tree.
- *Web Service* entity profiles are designed for use with Web Services. They allow simple restriction of each entity to a particular view. Any entity not specified here can still be accessed using its default profile. To remove an entity, remove it from the list of children on the parent view.

Parent Profile

Each profile may have a parent. A child inherits entities, and their settings, from the parent profile.

Only one level of inheritance is allowed, so a profile can be a parent or a child, but not both. This avoids the potential for confusion and maintenance difficulty.

Additional entities may be added on children, and entity settings may be overridden.

Entity Name

Select the name of the entity to be associated with this profile, from the list of entities defined in the registry.

Entity View

Select the entity view to be used for this entity in the context of this profile.

View

Click this link to see details of access based on the entity view.

Hierarchy

Click this link to see the complete hierarchy, starting from the specified entity, based on the current profile.

At the end of the page, if you select *Entity Profile,* this allows the addition of all entities from another profile. Because this is a one time copy it can be done as often as you want.

If you select *Entity Tree,* you can add entities based on a parent entity being specified, tracing down the entire tree and adding data. This is a one time add, so it does not do maintenance.

Synchronizing Entity Properties

This section discusses how to run the Entity Property Sync process.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Page Used to Synchronize Entity Properties

Page Name	*Definition Name*	*Navigation*	*Usage*
Entity Property Sync	SCC_ENTITY_SYNC	Set Up SACR, System Administration, Entity, Entity Property Sync	Simultaneously synchronize all the entities configured on the Entity Registry page.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Running the Entity Property Sync Process

Access the Entity Property Sync page (Set Up SACR, System Administration, Entity, Entity Property Sync).

Image: Entity Property Sync page (1 of 2)

This example illustrates the fields and controls on the Entity Property Sync page (1 of 2). You can find definitions for the fields and controls later on this page.

Entity Property Sync

Press the "Sync All Entity Properties" to make sure all the properties on every entity get updated to properly reflect the record objects. The updates you manually have made to the properties will not be reset by this process.

Sync Log

Entity Name	Last Synced	Log
Test Score	10/05/13 11:02:52AM	Log
Constituent	20/05/13 11:57:50PM	Log
Prospect Recruiting Category	09/05/13 4:32:37PM	Log
Prospect Career	10/05/13 10:54:00AM	Log
Disability	09/05/13 4:12:11PM	Log
Academic Program (NLD)	21/05/13 11:22:11PM	Log
Student Prior Education (NLD)	09/05/13 4:11:45PM	Log
Student Nationalities (NLD)	09/05/13 4:12:02PM	Log
Higher Ed Student Data (NLD)	09/05/13 4:12:04PM	Log
ClassSearchRequest	29/05/13 2:49:39PM	Log
ClassSectionMeetingPattern	14/05/13 11:45:47PM	Log
CourseOfferingDetail	21/05/13 11:33:54PM	Log
StudyListDeadLines	09/05/13 4:11:10PM	Log
Research Supervisor	09/05/13 4:11:58PM	Log
SCC_ST_ClassSearchRequest	09/05/13 4:11:16PM	Log
EnrollmentDetail	21/05/13 9:27:28AM	Log
APT Attempt	09/05/13 4:11:20PM	Log
Rule Statement	09/05/13 4:11:36PM	Log
ACR Main	22/05/13 4:16:21PM	Log

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Image: Entity Property Sync page (2 of 2)

This example illustrates the fields and controls on the Entity Property Sync page (2 of 2). You can find definitions for the fields and controls later on this page.

ACR Attendance Inclusion	09/05/13 4:11:29PM	Log
DA Proxy	09/05/13 4:11:26PM	Log
DA Proxy Transactions	09/05/13 4:11:26PM	Log
ACM Attendance Inclusion	09/05/13 4:11:32PM	Log
ACM Main	09/05/13 4:11:33PM	Log
Course Group Header	09/05/13 4:11:21PM	Log
IAM Main	22/05/13 4:14:53PM	Log
IAM Result	22/05/13 4:11:44PM	Log
APT Header Attribute	09/05/13 4:12:21PM	Log
Entity View Property	09/05/13 4:11:37PM	Log
AIR Rule	09/05/13 4:11:40PM	Log
ACM Dates	21/05/13 11:21:47PM	Log
IAM Dates	22/05/13 4:06:32PM	Log
Test Date	09/05/13 12:31:56PM	Log
Test CAF	09/05/13 12:32:56PM	Log
Prospect CAF	09/05/13 12:31:03PM	Log
Test ID Graduate Record Exam1	10/05/13 1:55:00PM	Log
Test ID International English	10/05/13 1:56:50PM	Log
Test ID Pearson Test of Englis	10/05/13 2:01:59PM	Log
Test ID ACT Assessment	21/05/13 11:21:51PM	Log
Prospect/Admissions Data Load	23/05/13 11:14:38AM	Log

Sync All Entity Properties

Entity Sync Status Complete

Entity Sync Datetime 22/05/13 4:47:31AM

Wipe Entity Cache

Sync All Entity Properties and **Wipe Entity Cache**

Click the Sync All Entity Properties button to start a Process Scheduler that performs the sync. The Sync All Entity Properties button is then replaced with a Refresh button so that you can check the status until the process is complete.

Running this process ensures that all the properties on every entity get updated to properly reflect the underlying record objects (stage record when it exists and production record). It does not reset the properties for which you may have manually changed the default setup.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The sync process keeps all the entity data up to date. The sync process:

- Updates properties to match underlying records.
- Keeps entity views up to date with changes to entity properties and children.
- Keeps entity properties/children synced with the Common Attribute Framework.

After the sync process is complete, click the Wipe Entity Cache button. The cache wipe process wipes the entity registry cache on the appserver. You should perform a cache wipe each time that you run the sync process – after the process has completed successfully. This cache uses rowset caches to store entity registry meta-data. This avoids unnecessary trips to the databases, bringing significant increases to performance for all entity registry operations.

You should run a sync and wipe in these events:

- Setting up a new database. This will make sure all your properties match the fields, and all your views match their underlying entity.
- Importing data via dat/dms. Since dat/dms bypasses all processing logic, this fills in any holes.
- If changes are made to records that have entities built on top of them via appdesigner. The sync process is the only way to update the entities to match the records. This can also be done on the entity registry component in a one-off manner.

Note: At the time of delivering this documentation, a similar overall process does not exist to “reset” all properties at once.

Entity Sync Status

Indicates whether the sync process is Requested, In progress or Complete. For further details on the status of the sync process look at the PeopleTools Process Monitor.

Sync Log

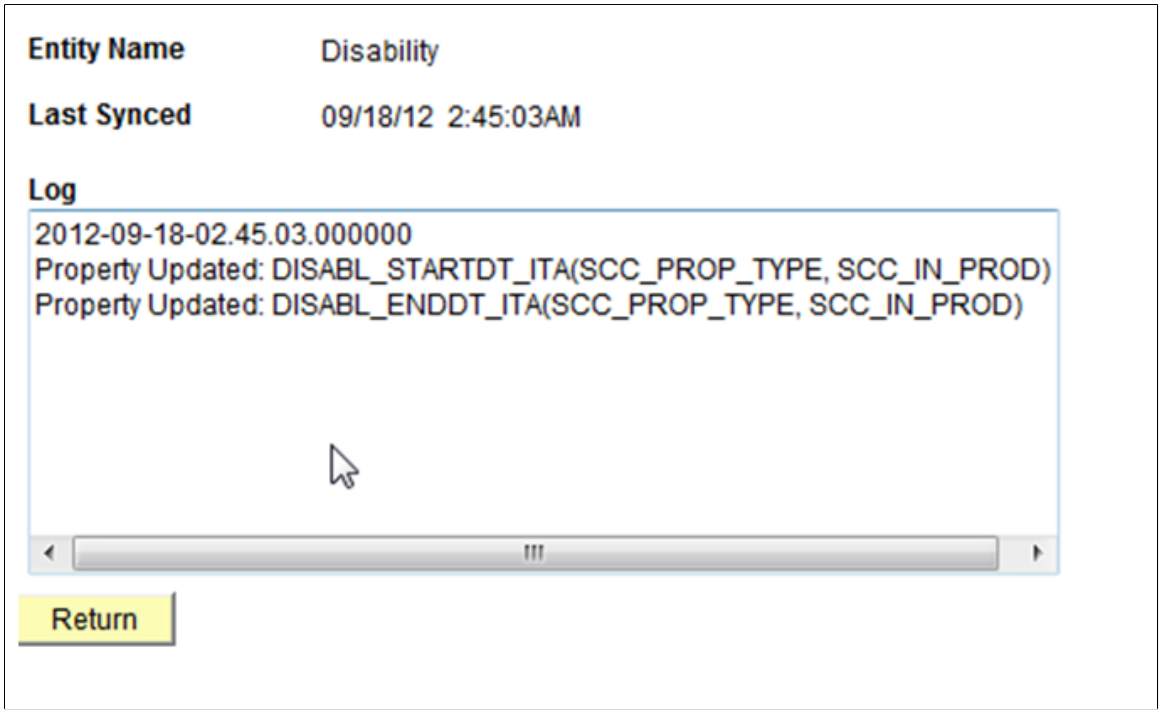
The full sync log grid is always shown on this page, showing all the entities that have been synced. If the sync was performed from the entity, as opposed to the Entity Property Sync page, the changes are also reflected here.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The grid also displays a Log link that enables you to view details of what was changed. Click the Log link to access the Sync Log page that provides a description of the fields that were added, removed, or changed:

Image: Sync Log page

This example illustrates the fields and controls on the Sync Log page. You can find definitions for the fields and controls later on this page.



The log grows over time, showing timestamped sections specifying the changes made.

Executing Unit Tests for Entities

This section discusses how to:

- Create an entity unit test.
- Run entity unit test cases.

Pages Used to Execute Unit Tests for Entities

Page Name	*Definition Name*	*Navigation*	*Usage*
Entity Unit Tests (Setup)	SCC_ENTITY_UT	Select *Unit Test* in the Action field on the Entity Registry page.	Attach the unit test application classes to an entity and run the unit tests.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Page Name	Definition Name	Navigation	Usage
Entity Unit Tester	SCC_ENT_MASS_UT	Set Up SACR, System Administration, Entity, Entity Unit Tester	Run all the unit tests associated with a specified entity. Unit tests are defined on the Entity Unit Tests (Setup) page.
Entity Unit Tests	SCC_ENTITY_UT	Click the View Unit Tests link on the Entity Unit Tester page.	View the unit tests tied to an entity name, the application class used and the latest run status for a specific unit test.

Creating an Entity Unit Test

Unit tests are small focused tests that academic institutions can use to ensure that each unit of code in an entity works properly. This is done to find errors in the smallest segments of code during development and allow them to be fixed early, resulting in higher quality code.

Because the Entity Registry component is highly extensible, the system integrates unit testing with the Entity Registry, by delivering a unit test framework.

Coding a Unit Test

Each unit test must extend the class TestBase.

There are three methods that the unit test can override in TestBase:

1. Setup(): Is the first method that your unit test must call. This method should be used to set up the environment for the test. If environment setup is not required, the unit test does not have to implement this method.

2. Run(): Is the main testing method. A unit test must run all the code relevant to executing the test in this method. If the test fails, the property status should equal the property FAILED. If the test runs successfully, the property status should be set to PASSED. This method must be overridden.

3. Teardown(): Is the method that does the cleanup. In theory, this method should reset all the data, as if the test had never been run, therefore, if required the test can be run again successfully. This method is optional, depending on your testing requirements.

The full class reference for TestBase is presented in the SCC_COMMON:Testing:TestBase application class. This application class is described later in this section.

The following code shows an example where a unit test retrieves a constituent that is based on TestBase:

```
import SCC_COMMON:Testing:*;
import SCC_COMMON:ENTITY:IEntity;
import SCC_SL_TRANSACTION:INTFC:Constituent;
class GetConstituent extends SCC_COMMON:Testing:TestBase
method GetConstituent();
method setup();
method teardown();
method run();
end-class;

method GetConstituent
%Super = create SCC_COMMON:Testing:TestBase("GetConstituent");
end-method;
```

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

```
method run
/+ Extends/implements SCC_COMMON:Testing:TestBase.Run +/
Local SCC_SL_TRANSACTION:INTFC:Constituent &constituent = create
SCC_SL_TRANSACTION:INTFC:Constituent( Null);
/*Retrieves the constituent*/
&constituent.setStageMode( True);
&constituent.EMPLID = "KU0004";
&constituent.fromProduction();

/*Prints in the message area the complete consitutent xml
for the constituent received*/
Local XmlDoc &responseDoc = CreateXmlDoc("");
Local XmlNode &rootNode =
&responseDoc.CreateDocumentElement("CONSTITUENT");
&constituent.toXmlNode(&rootNode);
%This.Msg(&responseDoc.GenFormattedXmlString());
end-method;

method setup
/+ Extends/implements SCC_COMMON:Testing:TestBase.Setup +/
end-method;

method teardown
/+ Extends/implements SCC_COMMON:Testing:TestBase.Teardown +/
end-method;
```

Attaching a Unit Test

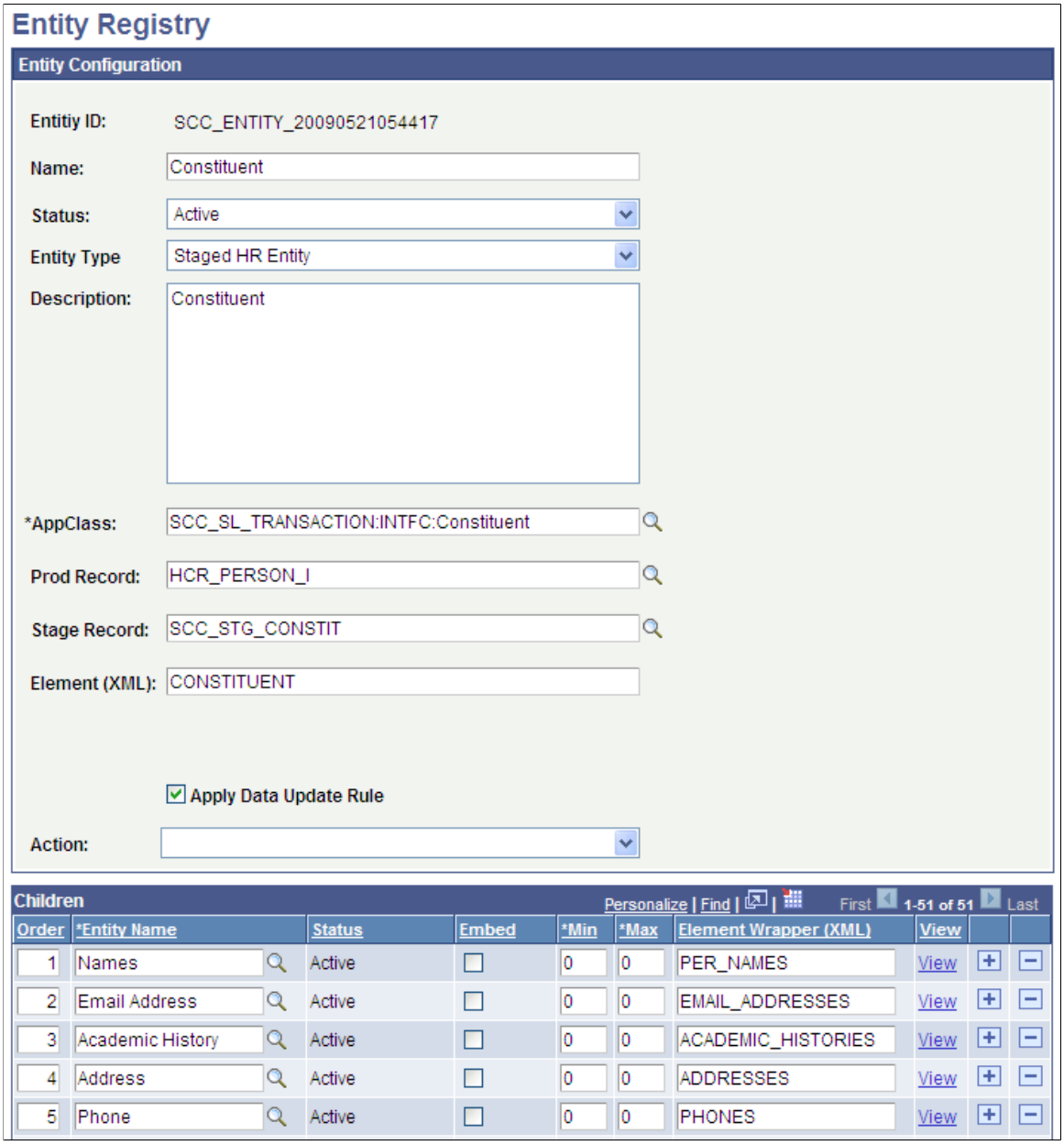
You attach a unit test to a specific entity. To attach a unit test to an entity:

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

1. Navigate to the Entity Registry page for the entity to which you want to attach a unit test (Set Up SACR, System Administration, Entity, Entity Registry). Continuing with the previous code example, the following shows that the Constituent entity is selected.

   Image: Example of the Entity Registry page for the Constituent entity

   This example illustrates the fields and controls on the Example of the Entity Registry page for the Constituent entity. You can find definitions for the fields and controls later on this page.

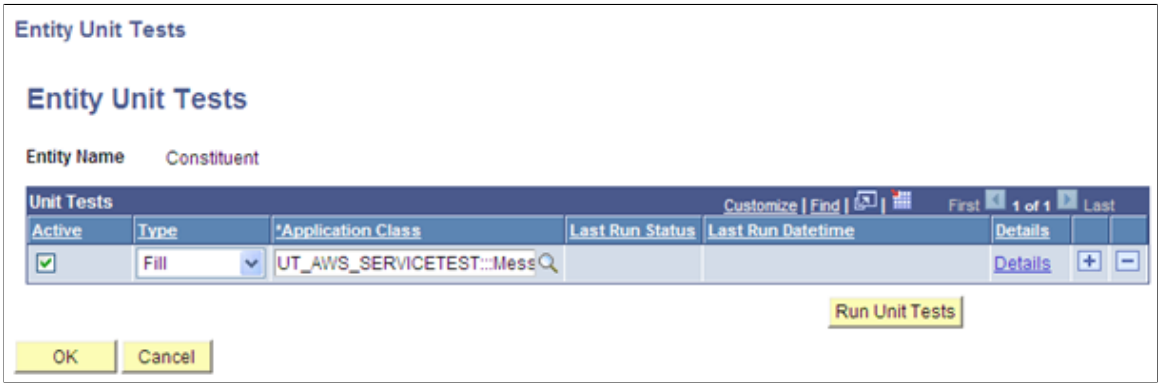


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

2. Select *Unit Test* in the Action field to access the Entity Unit Tests (setup) page.

   Image: Entity Unit Tests (setup) page (for the Constituent entity)

   This example illustrates the fields and controls on the Entity Unit Tests (setup) page (for the Constituent entity). You can find definitions for the fields and controls later on this page.



Active	Select to allow the system to run the unit test. If you clear this check box, the system ignores the unit test.
Type	Select a categorical classification of the unit test, this is mostly for description.
Application Class	Select the class that you have created for the unit test. In the previous exhibit example — Entity Unit Tests (setup) page for the Constituent entity — the MessageGenerator class is selected for the Constituent entity.
Run Unit Tests	Click to run the unit test. You can add any number of unit tests on this page and click this button once to run all the tests simultaneously.
OK	Click to save this unit test setup.

The other fields on this page are similar to the Entity Unit Tests (view-only) page that is described in the next section “Running Entity Unit Test Cases”.

class SCC_COMMON:Testing:TestBase

This is the base class which developers must extend in order to code their tests. The methods to be overridden are Setup(), Run(), and Teardown(). The system always calls these methods in that order for each test class. In the final phase of code execution, the system attempts to call Teardown(), even when there are errors.

In addition, the system delivers helper methods for asserting various invariant conditions. The syntax is: *Assert (&condition_which_must_be_true, "The expected condition was not met!");* In other words, the first argument to Assert() is a condition which you expect to be true. The second argument is an error message to display if the condition is not met. The other Assert*() methods are more of syntactic sugar.

Summary:

- *Property Summary:*

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- public string FAILED
- public string NOT_RUN
- public string PASSED
- public string status
- public string UnitTestName

- *Constructor Summary:*

Method Modifier and Return Type	***Method Name and Parameters***	***Method Description***
public void	TestBase(string UnitTestName_in)	This is the constructor. Developers should pass in a test name as the sole argument. For now, the system does not use this value.

- *Method Summary:*

Method Modifier and Return Type	***Method Name and Parameters***	***Method Description***
public void	Assert (boolean isTrue, string onFail)	The first argument must be a Boolean expression, which should evaluate to true. If false, the system throws an exception to indicate that the test failed. The system catches the exception, marks the test as having failed, and continues on to the next test.
public void	AssertNumbersDiffer(number num1, number num2, string onFail)	NA
public void	AssertNumbersEqual(number num1, number num2, string onFail)	NA
public void	AssertStringsDiffer(string str1, string str2, string onFail)	NA
public void	AssertStringsEqual(string str1, string str2, string onFail)	NA
public void	Msg(string msg)	NA

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Method Modifier and Return Type	***Method Name and Parameters***	***Method Description***
public void	Run()	This method must be overridden to implement the actual test that developers want to write. ***Warning!*** If you do not override this method, the system throws an error.
public void	Setup()	The first method of the test class to be called during a run.
public void	Teardown()	The last method of the test class to be called during a run. The framework tries to call this method even after an error. Due to inconsistent exception handling by the PeopleCode runtime, a PeopleCode "catch" statement does not catch all exceptions, and so the system cannot guarantee this method will be run after an error. Examples of untrappable errors include calls to the Error() builtin, SQL errors, and PeopleCode compile errors.

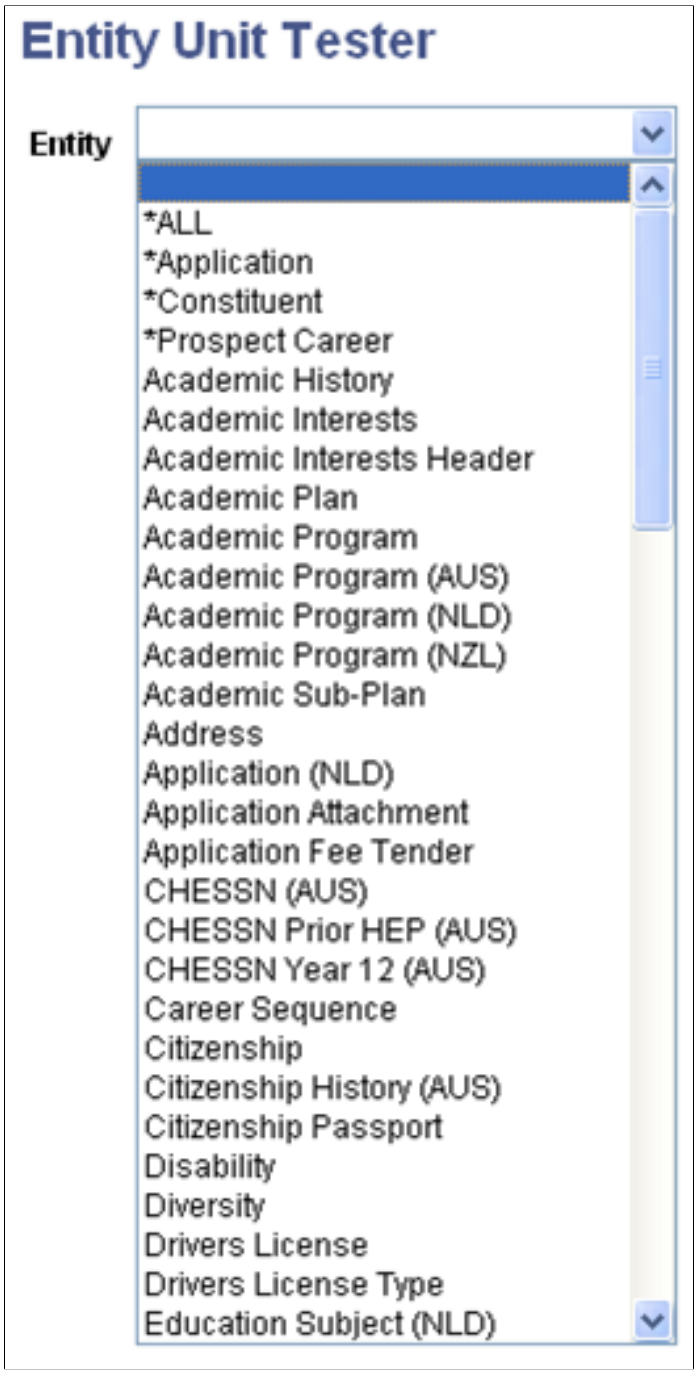
 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Running Entity Unit Test Cases

Access the Entity Unit Tester page (Set Up SACR, System Administration, Entity, Entity Unit Tester).

Image: Entity Unit Tester page

This example illustrates the fields and controls on the Entity Unit Tester page. You can find definitions for the fields and controls later on this page.



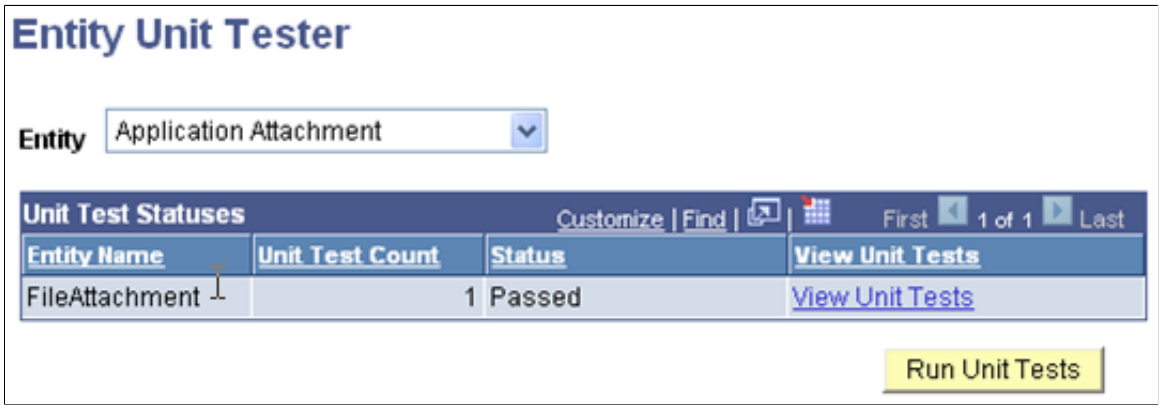
This component is purely for running unit tests that you associate with entities on the Entity Unit Tests (setup) page. The unit tests themselves are based on the Campus Solutions delivered unit test framework.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

First, select an entity. This loads the unit tests specified in the Entity Registry for that entity and all child entities in the grid below it. Except for the entity called ‘All’, the prompt dynamically marks with an * the entities that have no parent entity. Those can be considered top level entities. ALL is a special case as it runs unit tests for all entities in the registry.

Image: Example of the Entity Unit Tester page for a selected entity

This example illustrates the fields and controls on the Example of the Entity Unit Tester page for a selected entity. You can find definitions for the fields and controls later on this page.



Once an entity is selected, any unit test for that entity or any children show up. If an entity has no unit tests, it does not appear in the grid.

Clicking the Run Unit Tests button starts running unit tests entity by entity. Each time all the unit tests for an entity are completed the status is returned and the next entity is set for running. This is all done automatically.

The statuses are set to either *Passed* or *Failed* - *Passed* telling you all the unit tests passed and *Failed* meaning at least one unit test failed.

Warning! Once the unit tests start, wait until statuses are set for all unit tests. Changing pages and other such actions interrupt the unit test process.

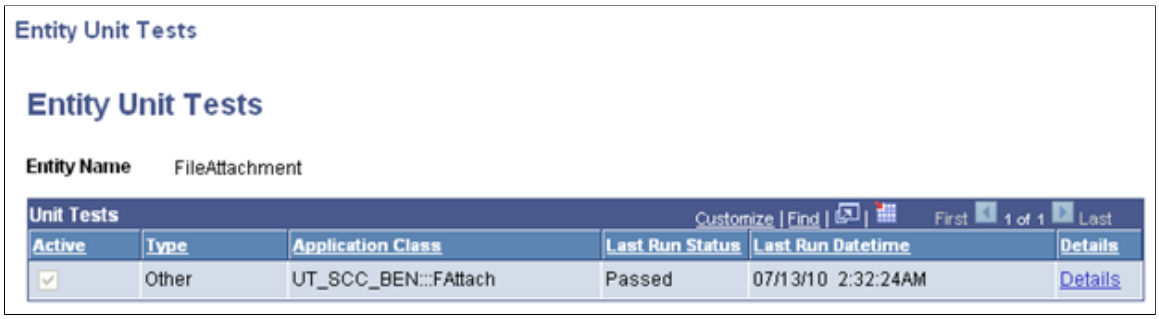
Note: When unit tests are run from the Entity Registry page directly, they perform a similar process. However, instead of running tests on a per entity basis, tests are run on a per unit test basis.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Click the View Unit Tests link on the Entity Unit Tester page to access the Entity Unit Tests page. The Entity Unit Tests page displays the individual unit tests for an entity.

Image: Entity Unit Tests page

This example illustrates the fields and controls on the Entity Unit Tests page. You can find definitions for the fields and controls later on this page.



Note: The fields on the Entity Unit Tests page are display only. Their setup is defined by selecting *Unit Test* in the Action field on the Entity Registry page for the entity.

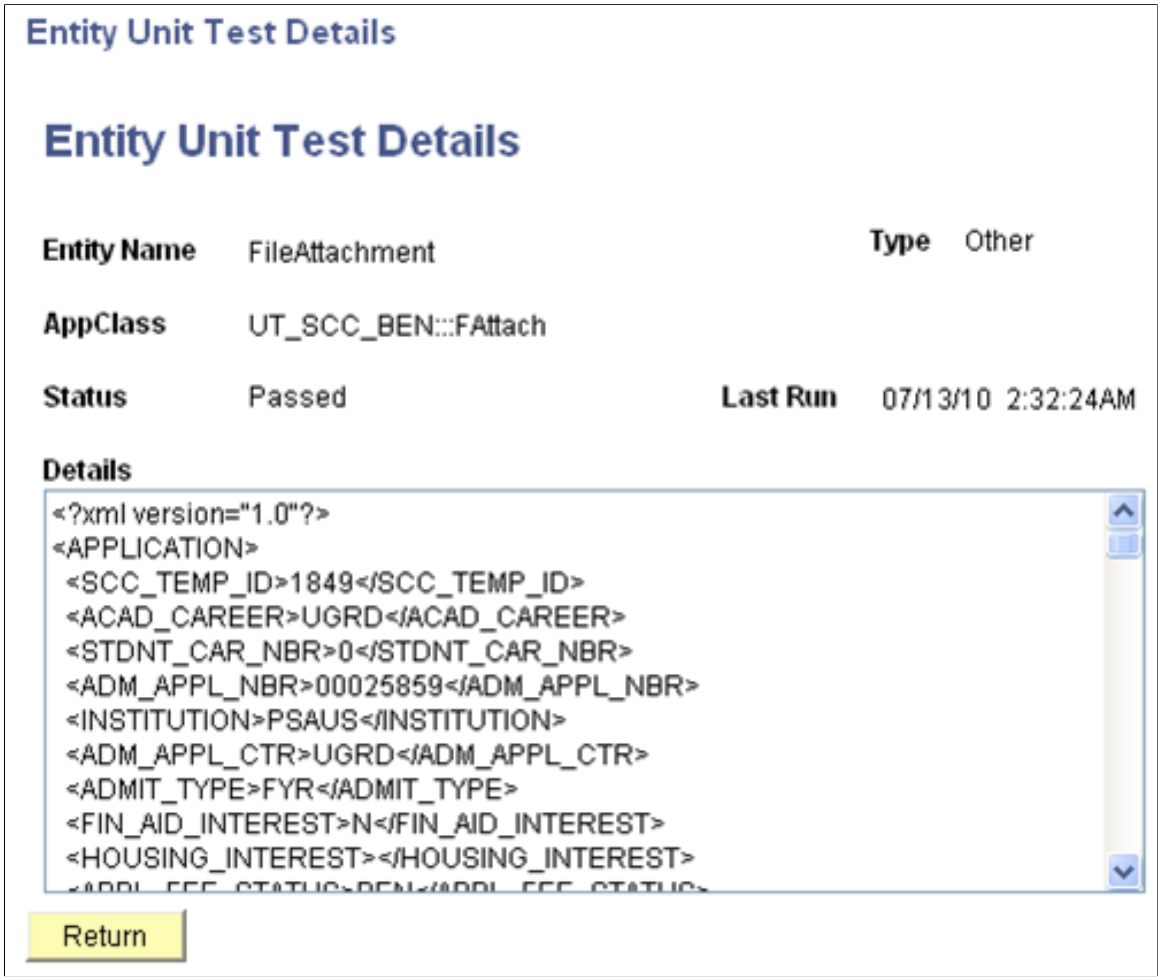
Active	Specifies if the unit test is set to active and should be run. Any unit test set to not active is not run.
Type	A categorical classification of the unit test, this is mostly for description.
Application Class	The application class of the unit test you created for this entity.
Last Run Status	The status the last time this unit test was run (*Passed* or *Failed*).
Last Run Datetime	The last time this unit test was run.
Details	Click to access the Entity Unit Test Details page. The page displays the details of the unit test, including all messages output. If a unit test fails, the information provided could help you identify the reasons of the failure.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The following is an example of the Entity Unit Test Details page:

Image: Entity Unit Test Details page

This example illustrates the fields and controls on the Entity Unit Test Details page. You can find definitions for the fields and controls later on this page.



Creating a New Entity

The way that you create a new entity varies based on the nature of the transaction that uses the entity. For example, if the entity is used as part of a Constituent Transaction Management transaction, the entity you create needs to be set with an entity type that uses stage records and production records. Those records need to be created prior to creating the entity. While these specificities are described in the features' related documentation, this section describes the generic steps that need to be taken independently from the transactions that use the entity.

This section discusses how to:

- Create or extend stage records.
- Create an entity application class.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- Create an entity.
- Generate XSD schemas.
- Set up or verify the Campus Solutions SOA framework.
- Set up logging.

Creating or Extending Stage Records

If the entity you are creating requires staging the information into a temporary record (stage record) prior to posting to a true production record, you first need to create the stage records prior to creating the corresponding entities. You can also extend any of the delivered stage records to meet your requirements. The *Staging* table holds the staged data before it gets promoted to its matching production table.

The newly created staging record along with its matching production record are later linked together through an entity defined in the Entity Registry component (see the *Creating an Entity* subsection in this section). In the Entity Registry component, you see that the entities have a parent/child relationship. This relationship is mostly based on the record structure of the production and the stage records.

Note: When creating the entity for the stage and production records, if you select one of the following two delivered entity types: *Staged Entity* or *Staged HR Entity,* the system expects the staging record to be keyed by the SCC_TEMP_ID field, and the production record keyed by EMPLID. If instead you create your own Entity Type that also uses staging records, then you can key the stage and the production records however you like.

The high level key used for your staging records (in most cases you use SCC_TEMP_ID) is a unique identifier that replaces the ID used in the equivalent production table. The rest of the keys for a staging record are similar to the equivalent production record.

A staging record is not effective-dated (it does not include the EFFDT and the EFF_STATUS fields). This is true even if the matching production record is effective-dated. This is because the data entered in staging can be modified many times prior to being promoted to production. Only at that time, the data is current, and therefore the production record is set with the date the data is posted.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Below is an example of a delivered staging record with its matching production record that uses effective-dating:

Image: Production record definition that stores Addresses (HCR_PER_ADDR_I)

This example illustrates the fields and controls on the Production record definition that stores Addresses (HCR_PER_ADDR_I). You can find definitions for the fields and controls later on this page.

HCR_PER_ADDR_I (Record)

Record Fields | Record Type

Num	Field Name	Type	Key	Ordr	Dir	CurC	Srch	List	Sys	Audt	InAu	EnAuto	Default
1	EMPLID	Char	Key	1	Asc		Yes	Yes	No		No	No	
2	ADDRESS_TYPE	Char	Key	2	Asc		Yes	Yes	No		No	No	
3	EFFDT	Date	Key	3	Desc		No	No	No		No	No	%date
4	EFF_STATUS	Char					No	No	No		No	No	'A'
5	ADDRESS_NPC_SBR	SRec					No	No	No		No	No	
6	ADDRESS1_AC	Char					No	No	No		No	No	
7	ADDRESS2_AC	Char					No	No	No		No	No	
8	ADDRESS3_AC	Char					No	No	No		No	No	
9	CITY_AC	Char					No	No	No		No	No	
10	REG_REGION	Char					No	No	No		No	No	

The equivalent staging record definition is defined as follows:

Image: Matching Staging record definition that stores Addresses (SCC_STG_ADDR)

This example illustrates the fields and controls on the Matching Staging record definition that stores Addresses (SCC_STG_ADDR). You can find definitions for the fields and controls later on this page.

SCC_STG_ADDR (Record)

Record Fields | Record Type

Num	Field Name	Type	Key	Ordr	Dir	CurC	Srch	List	Sys	Audt	InAu	EnAuto	Default
1	SCC_TEMP_ID	Nbr	Key	1	Asc		No	No	No		No	No	
2	ADDRESS_TYPE	Char	Key	2	Asc		Yes	Yes	No		No	No	
3	SCC_ADDRESSAREA	Char					No	No	No		No	No	
4	ADDR_TYPE_DESCR	Char	Alt		Asc		No	Yes	No		No	No	
5	ADDRESS_SBR	SRec					No	No	No		No	No	
6	ADDRESS1_AC	Char					No	No	No		No	No	
7	ADDRESS2_AC	Char					No	No	No		No	No	
8	ADDRESS3_AC	Char					No	No	No		No	No	
9	CITY_AC	Char					No	No	No		No	No	
10	REG_REGION	Char					No	No	No		No	No	
11	SCC_AUDIT_SBR	SRec					No	No	No		No	No	

The delivered staging tables can be extended or modified to meet your needs. For instance, if you customized the production record that stores addresses (HCR_PER_ADDR_I record) by adding new fields, you may want to make sure those extra fields are also added to the its corresponding staging record (SCC_STG_ADDR record).

Note: If you create or extend a stage record that is later used as part of the Constituent Transaction Management framework, review the information in Developer Reference to Deploy New User Registration, “Step 1: Creating or Extending Staging Tables.”

Creating an Entity Application Class

When creating a new entity, you must select an entity type. The entity type selected is set up with a base application class. The new entity uses this base application class by default. However, you can create a

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

new application class that is specific to the entity name. This *entity* application class extends the base application class.

To create a new entity application class, follow the below steps:

1. In the Application Designer, create a new application class under a new application package or reuse an existing one. For example, the application classes setup with delivered entities were created under the following application packages:

Delivered Transactions	***Application Package***
AAWS Admission Transactions	SAD_ADM_APPL
Academic Item Registry (AIR)	SSR_AIR
Academic Progress Tracker (APT)	SSR_APT
Activity Management	SSR_ACTIVITY
Constituent Transaction Management (CTM)	SCC_SL_TRANSACTION
Class Search	SSR_CLASS
Course Shopping Cart	SSR_COURSE
Delegated Access	SCC_DA
Entity API	SCC_COMMON:ENTITY:API
Evaluation Management	SCC_GENERIC_EVALUATIONS
Notification Framework	SCC_COMMON:NOTIFICATION
Research	SSR_RS_RESEARCH
Rules Engine	SCC_RULES_ENGINE
StudyList	SSR_STUDYLIST

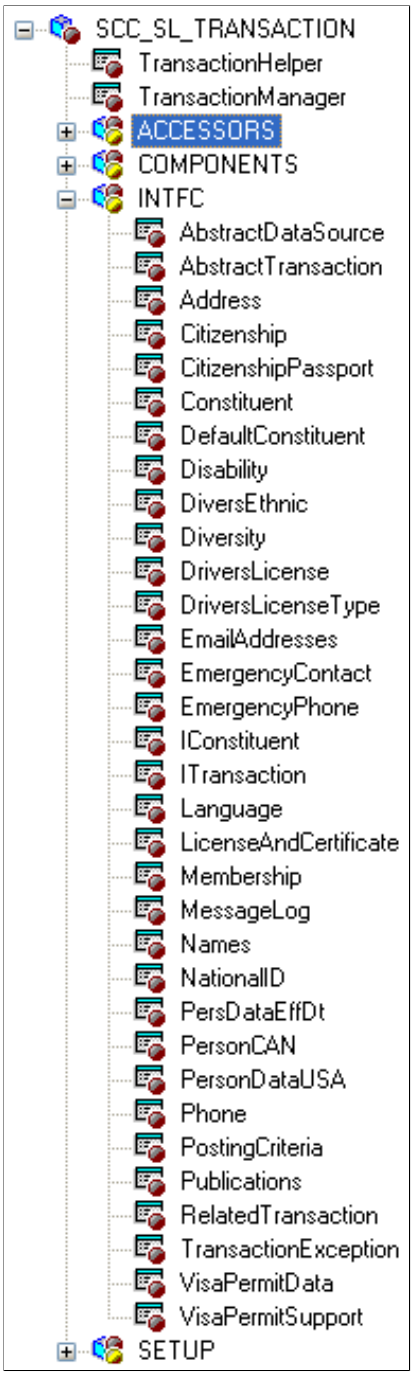
The above table shows the application packages used by the delivered transactions that use entity registry.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Below is an example of the SCC_SL_TRANSACTION:INTFC application package that comprises application classes used by Constituent Transaction Management:

Image: Example of application classes used by CTM

This example illustrates the fields and controls on the Example of application classes used by CTM. You can find definitions for the fields and controls later on this page.



 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Warning! It is recommended that you do not update delivered application packages to insert the entity application class. Instead, create a separate application package. This is because any modifications made to the delivered application package is considered a customization, and might be overridden when a software updates are delivered.

2. Save the new application class without inserting any Peoplecode logic. Use the entity registry to generate the Peoplecode. This task is defined in the next steps.

Creating an Entity

This section discusses how to create an entity for the application class you created in the previous section.

The new entity can be a parent entity or a child of an existing parent. For instance, delivered with your system, the *Constituent* and the *Application* entities are parent entities of many children.

Creation considerations:

1. Name the new entity in relation with the underlying record(s) (that is, the stage and/or the production records you are creating the entity for). For example, the entity name for the address underlying records is *Address.*

2. Select a delivered entity type that corresponds to the need of the transactions that use the entity. For example, if you create an entity for a transaction that requires temporarily staging the information to a stage record, make sure you select an entity type that use stage records. If the delivered entity types do not match your needs, create a new one.

3. Enter the application class name you created in the previous step. The application class you insert here extends the Base Application Class defined in the Entity Type selected. For example, if you create an entity using entity type Staged Entity, the application class you insert extends the base application class *SCC_COMMON:ENTITY:StagedEntity.* If you used entity type Basic Entity, the application class given inside the entity setup extends the base application class *SCC_COMMON:ENTITY:BasicEntity.*

4. Enter the other required information and save. After the first save, the Properties button is displayed. Use the Entity Properties page to add any extra needed properties. Save.

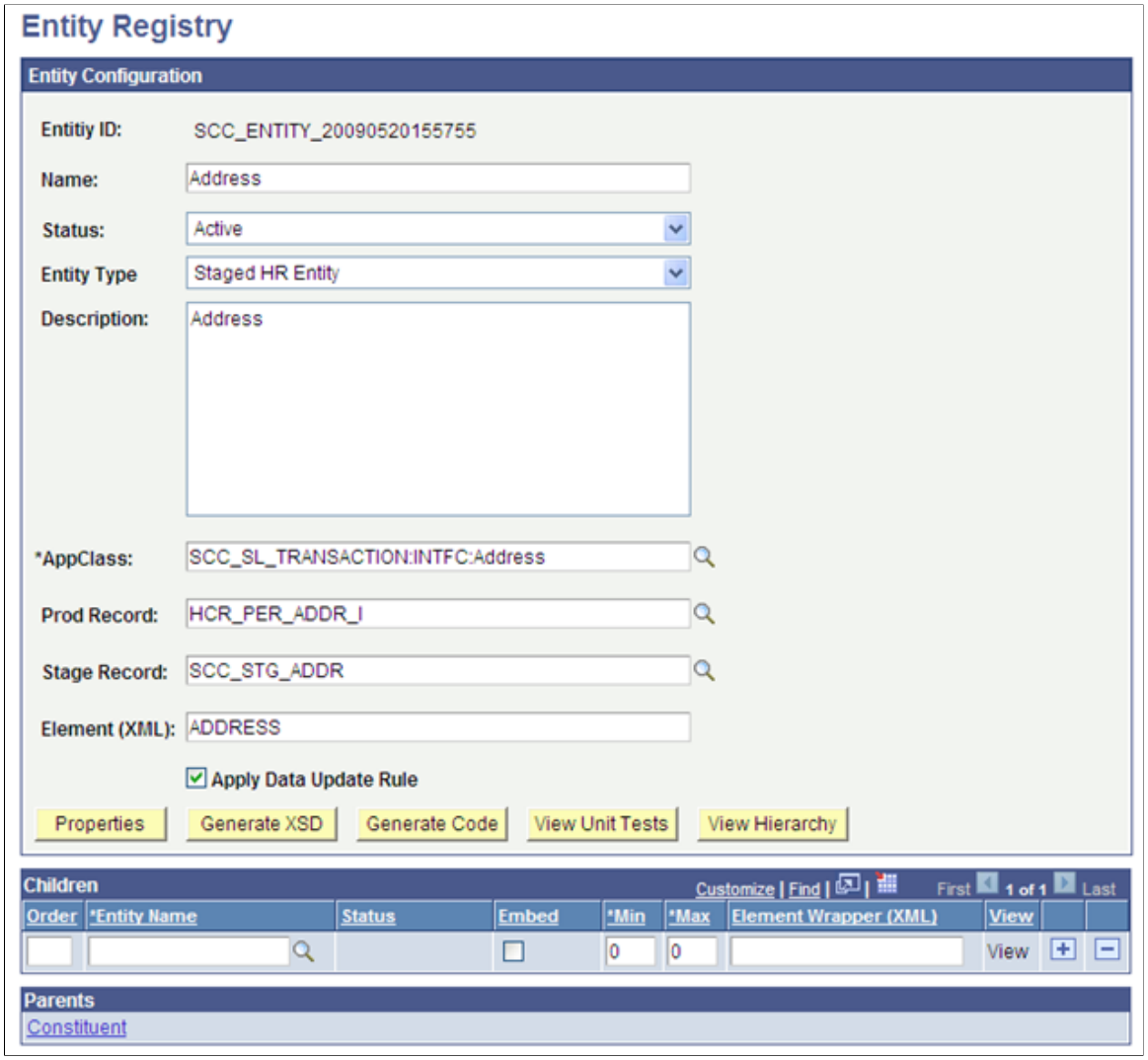
5. If the entity you created was to be a child of an existing entity (like *Address* is a child to *Constituent* entity), you can now access the parent entity and list the newly created entity in the Children grid of the Entity Registry page. If the entity you created was to be a parent, you can then attach children to it.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Warning! At this stage, if you added a child entity, the message schema does not reflect this new entity. However, if the entity is added to an incoming message and no schema validation is occurring, the incoming message containing this data is processed and this entity is handled. As an example, the Address entity is registered as a child entity to the Constituent entity.

Image: Example of the entity created for Address

This example illustrates the fields and controls on the Example of the entity created for Address. You can find definitions for the fields and controls later on this page.



Generate the Application Class Peoplecode

When creating the entity application class, you do not enter peoplecode logic inside the application class. After creating the new entity and entering the newly created application class, use the Entity Registry to write the application class peoplecode for you.

From the Entity Registry component for the entity you created above, select *Generate Code* in the Action field. In the Entity Code Generation page, select the desired options to include in the peoplecode. The inclusions later facilitate the work of customizing the peoplecode (Custom Presave Code, Custom Validation Code, Custom SetDefault Code and Extend Fill). Click the Generate Code button. The

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

AppClass Code box is shown. Select all the code and paste it in the peoplecode of the application class you created above in Application Designer.

In Application Designer, modify the peoplecode as needed:

- Any new class that you create must declare all its own immutable keys and mutable properties:

  Image: Examples of properties listed inside the SCC_SL_TRANSACTION:INTFC.Address application class

  This example illustrates the fields and controls on the Examples of properties listed inside the SCC_SL_TRANSACTION:INTFC.Address application class. You can find definitions for the fields and controls later on this page.

```
property string ADDRESS_TYPE get set;
property date EFFDT get set;

property string ADDRESS1 get set;
property string ADDRESS2 get set;
property string ADDRESS3 get set;
property string ADDRESS4 get set;
property string POSTAL get set;
property string CITY get set;
property string STATE get set;
property string COUNTRY get set;
property boolean PRIMARY;

property string ADDRESS1_AC get set;
property string ADDRESS2_AC get set;
property string ADDRESS3_AC get set;
property string CITY_AC get set;
property string REG_REGION get set;

/*other fields from ADDRESS_SBR*/
property string NUM1 get set;
property string NUM2 get set;
property string HOUSE_TYPE get set;
property string ADDR_FIELD1 get set;
property string ADDR_FIELD2 get set;
property string ADDR_FIELD3 get set;
property string COUNTY get set;
property string GEO_CODE get set;
property string IN_CITY_LIMIT get set;
```

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- Any new class that you create may choose to override the validate() method in order to validate data prior to saving.

  Image: Examples of validation done inside the SCC_SL_TRANSACTION:INTFC.Address application class

  This example illustrates the fields and controls on the Examples of validation done inside the SCC_SL_TRANSACTION:INTFC.Address application class. You can find definitions for the fields and controls later on this page.

```
method validate
   /+ &p_messageLog as SCC_COMMON:ENTITY:LOG:MessageLogBase out +/
   /+ Extends/implements SCC_COMMON:ENTITY:AbstractEntity.validate +/

   Local string &add1, &add2, &add3, &add4;
   Local SCC_COMMON:ENTITY:LOG:MessageEntry &msgEntry;

   %Super.validate(&p_messageLog);

   If Not %This.entityDelete Then

      /* An Address record must contain one of the ADDRESS1,2,3 or 4 fields.  */
      &add1 = %This.ADDRESS1;
      &add2 = %This.ADDRESS2;
      &add3 = %This.ADDRESS3;
      &add4 = %This.ADDRESS4;

      If None(&add1, &add2, &add3, &add4) Then
         /*2030817000 */
         rem     &p_messageLog.write(1000, 542, 2, "", "", "", "", "");
         &msgEntry = create SCC_COMMON:ENTITY:LOG:MessageEntry();
         &msgEntry.DataPopulateV1(1000, 542, &msgEntry.Severity_Error, Null, "", %
         &p_messageLog.writeEntry(&msgEntry);

         /*2030817000 */
      End-If;
      End-If;
end-method;
```

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- Any new class that you create may choose to override the preSave() method in order to default fields before the user saves the application.

  Image: Examples of presave logic done inside the SCC_SL_TRANSACTION:INTFC.Names application class

  This example illustrates the fields and controls on the Examples of presave logic done inside the SCC_SL_TRANSACTION:INTFC.Names application class. You can find definitions for the fields and controls later on this page.

```
method preSave
   /+ Extends/implements SCC_COMMON:ENTITY:AbstractEntity.preSave +/

   /*2003758000*/
   Local string &srch;
   /*2003758000*/
   If None(%This.data.FIRST_NAME_SRCH.Value) Then
      /*2003758000*/
      REM  %This.data.FIRST_NAME_SRCH.Value = Upper(%This.data.FIRST_NAME.Value);
      RemoveAccent(%This.data.FIRST_NAME.Value, &srch);
      %This.data.FIRST_NAME_SRCH.Value = &srch;
      /*2003758000*/
   End-If;

   If None(%This.data.LAST_NAME_SRCH.Value) Then
      /*2003758000*/
      REM  %This.data.LAST_NAME_SRCH.Value = Upper(%This.data.LAST_NAME.Value);
      &srch = "";
      RemoveAccent(%This.data.LAST_NAME.Value, &srch);
      %This.data.LAST_NAME_SRCH.Value = &srch;
      /*2003758000*/
   End-If;


end-method;
```

- Any new class that you create may choose to override the setDefault() in order default values.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- Any new class may declare get/set methods for its properties.

  Image: Examples of get and set inside the SCC_SL_TRANSACTION:INTFC.Address application class

  This example illustrates the fields and controls on the Examples of get and set inside the SCC_SL_TRANSACTION:INTFC.Address application class. You can find definitions for the fields and controls later on this page.

```
get ADDRESS1
   /+ Returns String +/
   Return %This.data.ADDRESS1.Value;
end-get;

set ADDRESS1
   /+ &NewValue as String +/
   %This.data.ADDRESS1.Value = &NewValue;
end-set;

get ADDRESS2
   /+ Returns String +/
   Return %This.data.ADDRESS2.Value;
end-get;

set ADDRESS2
   /+ &NewValue as String +/
   %This.data.ADDRESS2.Value = &NewValue;
end-set;
```

Generating XSD Schemas

The features using Entity Registry can use web services. Depending on what entities are referenced by the messages on the service operation, select an action of Generate XSD for the entity name that is considered the top level used. Selecting the Generate XSD action creates the message schemas of the API for that entity.

To do so, perform the following steps:

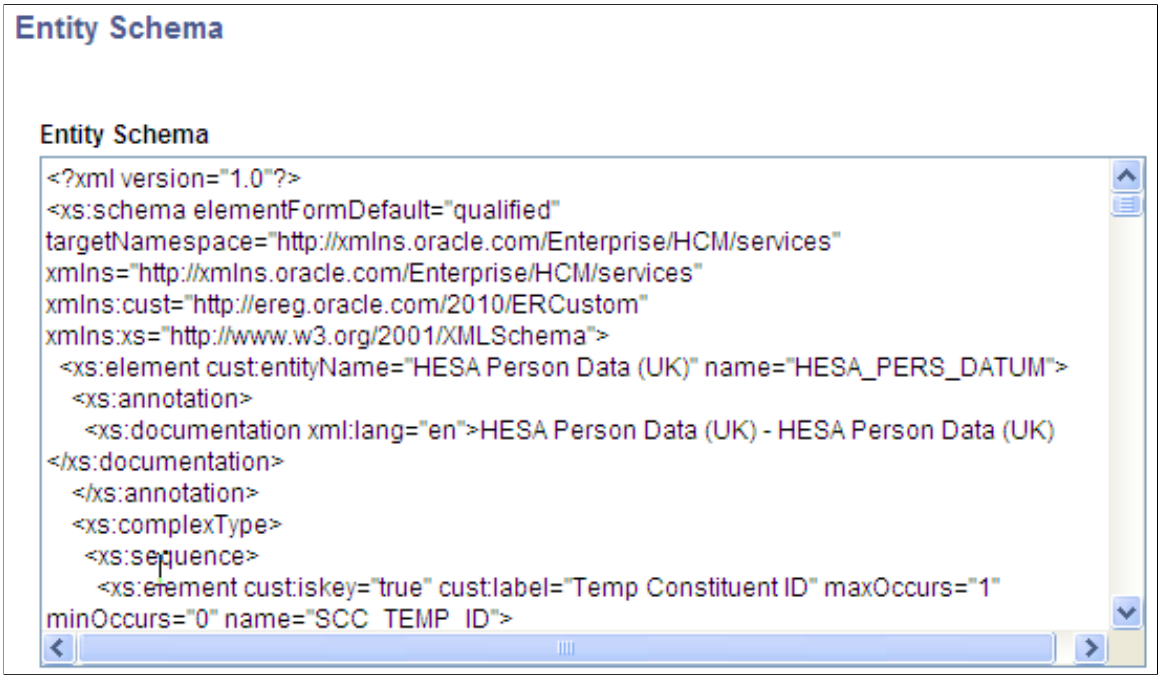
1. Access the top level entity name referred by the messages in the service operation and select the Generate XSD action (this may not necessarily be the parent entity name. It depends on what entities are referenced by the messages on the service operation).

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

2. After selecting the Generate XSD action, the entity schema is displayed:

   Image: Entity schema generated for the Constituent entity (parent entity)

   This example illustrates the fields and controls on the Entity schema generated for the Constituent entity (parent entity). You can find definitions for the fields and controls later on this page.



3. Select the entire generated schema and paste it into the appropriate messages to modify the overall schema.

4. Navigate to PeopleTools, Integration Broker, Integration Setup, Messages. Search for the relevant message. In the Schema tab, click the Edit Schema button and paste in the generated schema. Save the component.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

For example, suppose you generated the schema for the *Constituent* entity. Navigate to the message Schema page. Search for the message called SCC_ENTITY_CONSTITUENT. Click the Edit Schema button and paste the generated schema as shown in screen shot below.

Image: Constituent message partial schema

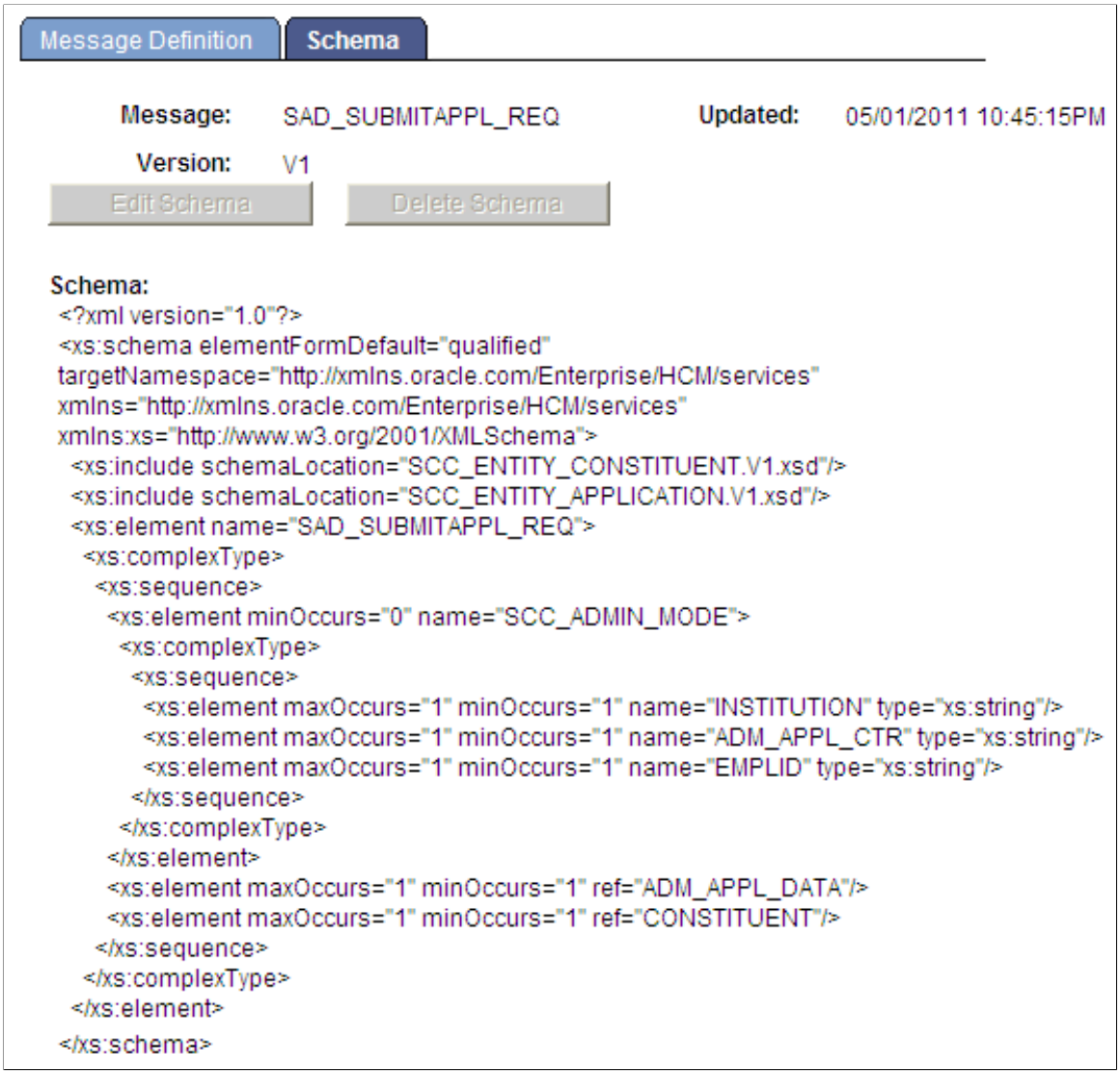
This example illustrates the fields and controls on the Constituent message partial schema. You can find definitions for the fields and controls later on this page.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Note: The messages where you paste the schema are *entity* messages. These messages must be created with a message type of *Part NonRowset* based. This means they can be used by any Nonrowset based message. The web services created for your transaction can also refer to delivered entity messages. For example, a message with AAWS admission transaction, SAD_SUBMITAPPL_REQ message, calls both SCC_ENTITY_CONSTITUENT and SCC_ENTITY_APPLICANT message schema in its schema.
It is easy to maintain the entity messages as their schema can be generated from the Entity Registry page. Entity messages can be reused in any request or response message as per your requirement.

Image: Example of a web service message calling the schema of a Part NonRowset message

This example illustrates the fields and controls on the Example of a web service message calling the schema of a Part NonRowset message. You can find definitions for the fields and controls later on this page.



5. Once the schemas have been updated, the service WSDL needs to be republished in order to have all the new or updated entities recognized. In order to do that delete the existing WSDL under

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

PeopleTools, Integration Broker, Service Utilities, Service Administration, WSDL. Select the appropriate service WSDL and press delete.

Republish the service WSDL under PeopleTools, Integration Broker, Web Services, Provide Web Service.

Setting Up or Verifying the Campus Solutions SOA Framework

During installation, the system automatically inserts configuration data into the Campus Solutions SOA Framework setup tables.

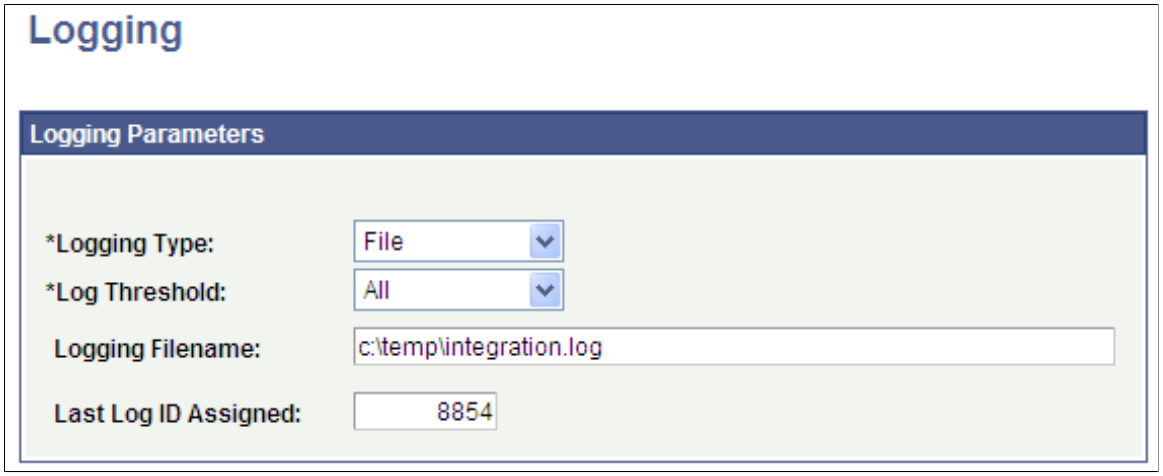
Access the Request Handlers component for the service operations that you created or are planning to use under Set Up SACR, System Administration, Integrations, Request Handlers. For each of the service operations you plan to use, ensure the Request Handler page contains the proper information. The application class path should be properly entered.

Setting Up Logging

Access the Logging component to enable and configure the logging feature under Set Up SACR, System Administration, Integrations, Logging.

Image: Example of how the Logging page can be set up

This example illustrates the fields and controls on the Example of how the Logging page can be set up. You can find definitions for the fields and controls later on this page.



Note: You should not enable logging in a production environment. SOA framework logging is only appropriate for a testing, demo or development environment or when performing critical troubleshooting activities.

See *PeopleTools: Integration Broker*.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Entity Application Class Reference

This section discusses the delivered application classes and interfaces that you can use for creating or modifying entities.

Bundle 48. New application classes:

SCC_COMMON:AUDIT:AuditRecord – to log data to audit records from entity registry

SCC_COMMON_UTIL:BitArray – to parse through bit data

interface SCC_COMMON:ENTITY:IEntity

Implementing Classes: SCC_COMMON:ENTITY:FileAttachment, SCC_COMMON:ENTITY:AbstractEntity

See Also: BasicEntity, StagedEntity, StagedHREntity

Summary

Property Summary:

public Array of string	baseProps: A list of properties that are controlled on the base entities, so the property name cannot be changed.
public Array of ChildEntity	ChildEntityArray: An array of all the ChildEntities of the current entity.
public Record	data: This entities a backing record.
public boolean	entityDelete: Marks this entity for deletion, and prevents save.
public string	entityID: ID of the entity from the registry.
public string	entityName: The name of the entity as defined in the registry.
public Array of DataKey	keyCollection: A collection of key values to be used when population of this entity occurs.
public IEntity	parent: parent Entity, null if it is at the top level of tree.
public ChildEntity	parentCE: The child entity that encompasses all the entities of a particular type under the parent.
public string	SCC_ENTITY_INST_ID: Entity Instance Id.
public boolean	selfServiceMode: Self Service mode flag.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public boolean	STAGE_MODE: Specifies if this entity is currently in Stage Mode (True) and working with the stage record, or in Production Mode (False) and working with the production record.
public string	UPDATE_RULE: Specifies the update rule to use for the purposes of Data Update Rule.
public string	USER_CONTEXT: The user that this is being run on behalf of, since adds and updates may be done from administrative mode on behalf of a given user.
public Array of string	XSDFieldList: A string list of all the fields for the current entity for the XSD. This is used in conjunction with buildXSD to determine if key fields are required, based on if their parent has the same key.

Method Summary:

public void	buildXSD(XmlNode xmlnode): Builds the xml schema section pertaining to this entity.
public void	clearDataPreImage(): Clears the DataPreImage for this entity and all its children.
public void	delete(boolean p_entityDelete): Sets entityDelete property for this entity and all its children.
public void	demote(number p_tempID): Moves data from production records to staging record.
public void	destroy(): Perform cleanup activity on this entity.
public void	fill(Record p_inRec out, boolean p_clearState): Fill (Populate) this entity using the given record, and populates all children based on the key structure.
public void	fillFromKeys(boolean p_clearState): Performs a select by key based on the fields on the underlying record. This should be run on the top level entity to start a fill.
public void	fromXmlNode(XmlNode parentNode): Converts data from XML to the entity model.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public Rowset	generateRowset(): This method generates a rowset of data for the underlying record of this entity based on the keys of the parent. It is used as part of the fill process for building out the entity tree. The method is called as if it were a static method, so it is not associated with any particular entity data. The base implementation dynamically matches keys from the parent to the child and builds the query for the rowset based on that.
public string	generateXSD(): Begins generation of an xml schema, based on the registry using this entity as the root.
public ChildEntity	getChildEntity(string p_entityID): Retrieves the ChildEntity Object based on the entity ID.
public string	getLOVDescr(string propname): Returns the LOV Description based on the associated property name when parsed from XML.
public number	getMaxSeqNbrSQL(): Retrieves the maximum sequence number from the database based on the key structure of the current entity.
public any	getProperty(string propname)
public void	hardDelete(MessageLogBase p_messageLog out): Performs the actual delete of all objects marked with the entityDelete flag.
public boolean	hasEffdt(): Checks if this entity has an effective date.
public boolean	hasEffseq(): Checks if this entity has an effective sequence.
public boolean	hasKey(string p_keyname): Checks if the current entity has a particular field as a key.
public boolean	isEqualKeys(IEntity p_entity): Determine Entity Key Equality.
public boolean	isEqualKeysRecord(Record p_rec): Allows the given record to be compared with this entity for equality. Compares for equality based on the underlying entity record content. RECOMMENDATION: Override this method if you need to do custom equality checking.
public void	prePublish(Rowset p_changesRS out): Uses the workflowPreimage and post images of the entity to build a PSCAMA Rowset in preparation for publish. The parameter Rowset may be Null. Data should be appended to this rowset if it is NON-Null.
public void	preSave(): Perform pre-Save activities on this entity.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	preSaveChildren()
public void	PreSaveFirst(): Pre-Save code that only runs on the first entity of a child array.
public void	promote(string emplID, MessageLogBase p_log out): Moves Data from Staging to Production Records (does not validate or save).
public void	publish(Rowset p_changesRS, boolean p_onlineMode): Takes a PSCAMA based rowset of information to be published during workflow Uses that information to construct a Message and according to the onlineMode publishes the Message data in realtime or to the batch delay table.
public void	save(MessageLogBase p_messageLog out): Save this entity and all its children.
public void	setChildren(): Builds out the structure of the ChildEntityArray object based on the registry.
public void	setDefault(): Sets default values for record fields.
public void	setProperty(string propname, any value)
public void	setStageMode(boolean p_stageInd): Changes the Stage_Mode property, as well as any related records. This should not be used once data has been set, to convert data use promote or demote.
public void	setUserContext(string p_userContext): Sets the value of the UserContext property.
public void	toXmlNode(XmlNode parentNode): Converts the data from the entity structure to XML.
public void	updateFromRecord(Record rec): Updates by matching fields in the record to fields in the underlying record on the entity. Values are only set if the value has changed. Values are set using properties, so any additional get-set logic is handled.
public void	updateKeyFields(): Updates the keyfields of all children based on the current entity. This allows key changes to roll down properly.
public void	validate(MessageLogBase p_messageLog out): Validate this entities' contents (and all its children).
public void	validateFirst(MessageLogBase p_messageLog out): Validation code that only runs on the first entity of a child array.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	workflow(boolean p_onlineMode): Orchestrates the prepublish and publish methods for this entity and all children.

Details

Property Details:

baseProps	public Array of string
ChildEntityArray	public Array of ChildEntity
data	public Record
entityDelete	public boolean
entityID	public string
entityName	public string
keyCollection	public Array of DataKey
parent	public IEntity
parentCE	public ChildEntity
SCC_ENTITY_INST_ID	public string
selfServiceMode	public boolean
STAGE_MODE	public boolean
UPDATE_RULE	public string
USER_CONTEXT	public string
XSDFieldList	public Array of string

Method Details:

buildXSD	Parameters: XmlNode xmlnode: The Tools XmlNode object of the parent that should be continued by this entity.
clearDataPreImage	N/A

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

delete	Parameters: boolean p_entityDelete: The value to be set for the entityDelete property.
demote	Parameters: number p_tempID: The tempid to replace the emplid with.
destroy	N/A
fill	Parameters: Record p_inRec(out): This record populates the record underlying this entity. boolean p_clearState: If the clearstate is true the datapreimage is populated as well.
fillFromKeys	Parameters: boolean p_clearState: Whether or not to RESET this entities pre-image.
fromXmlNode	Parameters: XmlNode parentNode: The XML node to pull from.
generateRowset	Return: Rowset - a rowset populated based on the parent keys, or null if no rows are found.
generateXSD	Return: string - the XSD as a string.
getChildEntity	Parameters: string p_entityID: The entity ID, based on the registry, of the ChildEntity object to retrieve. Return: ChildEntity

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

getLOVDescr	Parameters: string propname: The property name to get a LOV Description based on. Return: string
getMaxSeqNbrSQL	Return: number - the maximum sequence number found.
getProperty	Parameters: string propname Return: any
hardDelete	Parameters: MessageLogBase p_messageLog(out): The MessageLog object to log all errors/warnings to.
hasEffdt	Return: boolean - true if it has an effective date false otherwise.
hasEffseq	Return: boolean - true if it has an effective sequence false otherwise.
hasKey	Parameters: string p_keyname: The name of the field to check. Return: boolean - true if the field is a key, false if the field is not a key or the entity does not include it.
isEqualKeys	Parameters: IEntity p_entity: The entity to compare the current entity to. Return: boolean - true if the entity keys are equal false otherwise.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

isEqualKeysRecord	Parameters: Record p_rec Return: boolean
prePublish	Parameters: Rowset p_changesRS(out)
preSave	N/A
preSaveChildren	N/A
PreSaveFirst	N/A
promote	Parameters: string emplID: The emplid for production to replace the tempid from staging. MessageLogBase p_log(out): The MessageLog object to log all errors/warnings to.
publish	Parameters: Rowset p_changesRS boolean p_onlineMode
save	Parameters: MessageLogBase p_messageLog(out): The messagelog object to log errors and warnings to.
setChildren	N/A
setDefault	N/A
setProperty	Parameters: string propname any value
setStageMode	Parameters: boolean p_stageInd: The value to set the stage mode to, if the value passed matches the current stage mode no changes are made.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

setUserContext	Parameters: string p_userContext: The value set for the property UserContext.
toXmlNode	Parameters: XmlNode parentNode: The XML parent node to push the xml to.
updateFromRecord	Parameters: Record rec: The record object to copy data from.
updateKeyFields	N/A
validate	Parameters: MessageLogBase p_messageLog(out): The messagelog object to log errors and warnings to.
validateFirst	Parameters: MessageLogBase p_messageLog(out): The MessageLog object to log all errors/warnings to.
workflow	Parameters: boolean p_onlineMode

abstract class SCC_COMMON:ENTITY:AbstractEntity

Implemented Interfaces: SCC_COMMON:ENTITY:IEntity

Direct Known Subclasses: SCC_COMMON:ENTITY:StagedEntity, SCC_COMMON:ENTITY:StagedHREntity, SCC_COMMON:ENTITY:BasicEntity, SCC_COMMON:ENTITY:WorkEntity

See Also: BasicEntity, StagedEntity, StagedHREntity

Summary

Property Summary

public Array of string	baseProps: A list of properties that are controlled on the base entities, so the property name cannot be changed.
public Array of ChildEntity	ChildEntityArray: An array of all the ChildEntities of the current entity. The childEntities are determined by the entity registry, and populated when the entity is instansiated.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public Record	data: The backing record for this entity.
public Record	dataPreimage: The preimage of the backing record for this entity. The entity is compare against the preimage prior to save.
public boolean	DO_SAVE: Specifies if the entity is allowed to save.
protected AnyHashMap	dynamicProperties
public boolean	entityDelete: Marks this entity for deletion, and prevents save.
public string	entityID: ID of the entity from the registry.
protected Record	EntityMetaData
public string	entityName: The name of the entity as defined in the registry.
protected boolean	hasAuditRowAddDttm
protected boolean	hasAuditRowAddOprid: A value, set by testForAudit, that speficies if this entity has audit fields. It is used during save time to determine if the audit fields need to be updated.
protected boolean	hasAuditRowUpdDttm
protected boolean	hasAuditRowUpdOprid
protected Array of string	ignoreFields: DEPRECATED. ignoreFields is no longer used; if it is found it is used by EntityPropBuilder during the upgrade to the new property system. The array of names of fields to ignore for the purposes of moving data to and from xml, pushing a new field onto this array adds it to the list.
public Array of DataKey	keyCollection
public IEntity	parent: Parent Entity, null if it is at the top level of tree.
public ChildEntity	parentCE: The child entity that encompasses all the entities of a particular type under the parent.
public string	prodRecordName: The name of the production record is populated here based on the registry.
protected Rowset	PropertyMetaData
protected string	RECORD_NAME: The name of the current record underlying this entity.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public string	SCC_ENTITY_INST_ID: A unique identifier for the specific row of data.
public datetime	SCC_ROW_ADD_DTTM: The Datetime the row of data was added, this is part of the who columns. This property corresponds to this field, if it exists on the record, if not it does nothing. RECOMMENDATION: Do not override get & set.
public string	SCC_ROW_ADD_OPRID: The Person who added the row of data, this is part of the who columns. This property corresponds to this field, if it exists on the record, if not it does nothing. RECOMMENDATION: Do not override get & set.
public datetime	SCC_ROW_UPD_DTTM: The Datetime the row of data was update, this is part of the who columns. This property corresponds to this field, if it exists on the record, if not it does nothing. RECOMMENDATION: Do not override get & set.
public string	SCC_ROW_UPD_OPRID The Person who updated the row of data, this is part of the who columns. This property corresponds to this field, if it exists on the record, if not it does nothing. RECOMMENDATION: Do not override get & set.
public boolean	selfServiceMode: If the entity is in selfServiceMode.
public boolean	STAGE_MODE: Specifies if this entity is currently in Stage Mode (True) and working with the stage record, or in Production Mode (False) and working with the production record.
public string	UPDATE_RULE: Specifies the update rule to use for the purposes of Data Update Rule.
public string	USER_CONTEXT: The user that this is being run on behalf of, since adds and updates may be done from administrative mode on behalf of a given user, or %UserId.
public Record	workflowPreimage: The pre-image object to use for workflow.
protected string	XSDCustomNS: The namespace to use for custom attributes in the schema, found and set during generateXSD.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public Array of string	XSDFieldList: A string list of all the fields for the current entity for the XSD. This is used in conjunction with buildXSD to determine if key fields are required, based on if their parent has the same key.
protected string	XSDNS: The namespace to use for the schema, found and set during generateXSD.

Constructor Summary:

public void	AbstractEntity(IEntity p_parent): The Constructor.

Method Summary:

protected string	applyDataKey(string p_queryString)
protected string	buildCSVString(array of any p_arrValues, string p_metaTag)
protected void	buildPropertyMetaDataCache()
public void	buildXSD(XmlNode xmlnode): Builds the xml schema section pertaining to this entity. The implementation builds and populates the schema based on the underlying record, ignoring any fields in the ignore array. Keys that roll down from the parent are not marked required on the child. RECOMMENDATION: Only override if the entity requires a custom schema.
protected void	buildXSDChildren(XmlNode xmlnode): Iterates through the ChildEntityArray and issues a buildXSD against each ChildEntity, which in turn creates a temporary entity and calls buildXSD. Base implementation uses the entity registry. RECOMMENDATION: Do not override.
protected void	changeRecord(string p_recname): Changes the record to the record name specified. RECOMMENDATION: Do not override.
public void	clearDataPreImage(): Clears the Data preimage and workflow preImage. RECOMMENDATION: Do not override.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

protected void	clearDataPreImageChildren(): Iterates through the ChildEntityArray and issues a clearDataPreImage against each ChildEntity, which in turn calls clearDataPreImage on all of the entities it contains. RECOMMENDATION: Do not override.
protected void	commonValidate(MessageLogBase p_messageLog out, number p_validateArgs): Runs the record based validations and outputs results to the log. Also invokes validateChildren. validateChildren is called before record based validations. Validation Arguments<table> <tr><td>%Edit_DateRange</td><td>Reasonable Date Range (Is the date contained within the specified reasonable date range?)</td></tr> <tr><td>%Edit_OneZero</td><td>1/0 (Do all 1/0 fields contain only a 1 or 0?)</td></tr> <tr><td> %Edit_PromptTable</td><td>Prompt Table (Is field data contained in the specified prompt table?)</td></tr> <tr><td> %Edit_Required</td><td>Required Field (Do all required fields contain data? For numeric or signed fields, it checks that they do not contain NULL or 0 values.)</td></tr> <tr><td> %Edit_TranslateTable</td><td>Translate Table (Is field data contained in the specified translate table?)</td></tr> <tr><td> %Edit_YesNo</td><td>Yes/No (Do all yes/no fields only contain only yes or no data?)</td></tr> </table>
protected void	defaultChildren(): Iterates through the ChildEntityArray and issues a setDefault against each ChildEntity, which in turn calls setDefault on all of the entities it contains. Base implementation uses the entity registry. RECOMMENDATION: Do not override.
public void	delete(boolean p_entityDelete): Marks this entity and all its children for deletion at save time - soft delete. RECOMMENDATION: Do not override this method.
protected void	deleteChildren(boolean p_entityDelete): Iterates through the ChildEntityArray and issues a delete against each ChildEntity, which in turn calls delete on all of the entities it contains. Base implementation uses the entity registry. RECOMMENDATION: Do not override.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	demote(number p_tempID): Moves data from production records to staging record. RECOMMENDATION: This must be implemented, but is only used if the entity is staged. In the implementation make sure to call demoteChildren to cause demote to work with the registry.
protected void	demoteChildren(number p_tempID): Iterates through the ChildEntityArray and issues a demote against each ChildEntity, which in turn calls demote on all of the entities it contains. Base implementation uses the entity registry. RECOMMENDATION: Do not override.
public void	destroy(): Perform cleanup activity on this entity.
public boolean	entityFieldUpdateNeeded(Rowset p_DataUpdateRule, string p_fieldName, any oldValue) RECOMMENDATION: Do not override
public boolean	entityTypeUpdateNeeded(Rowset p_DataUpdateRule, any p_entityType, boolean p_newFlag) RECOMMENDATION: Do not override.
public boolean	entityTypeUpdateRequested(Rowset p_DataUpdateRule, any p_entityType) RECOMMENDATION: Do not override
public boolean	entityUpdateRequested(Rowset p_DataUpdateRule) RECOMMENDATION: Do not override.
public void	fill(Record p_inRec out, boolean p_clearState): Populates (fills) this entity from the given record. RECOMMENDATION: Do not override.
protected void	fillChildren(boolean p_clearState): Iterates through the ChildEntityArray and issues a fill against each ChildEntity, which in turn calls fill on all of the entities it contains. Base implementation uses the entity registry. RECOMMENDATION : Do not override.
public void	fillFromKeys(boolean p_clearState): Performs a select by key based on the fields on the underlying record. This should be run on the top level entity to start a fill.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

protected void	fillLOVDescrs(): Re-initializes and Populates the LOV Description Hash Map shmLOVDescr based on the current property values.
public void	fromXmlNode(XmlNode parentNode): Retrieves data to properties based on the fields in the record. Fields marked ignore are not Processed. Entity Element Names, wrapping of the ChildEntities or embedding is based on the entity registry. RECOMMENDATION: Only override if the xml for this entity has to be custom.
protected void	fromXmlNodeChildren(XmlNode parentNode): Iterates through the ChildEntityArray and issues a fromXMLNode against each ChildEntity, which in turn calls fromXMLNode on all of the entities it contains. Base implementation uses the entity registry. RECOMMENDATION: Do not override.
public Rowset	generateRowset(): This method generates a rowset of data for the underlying record of this entity based on the keys of the parent. It is used as part of the fill process for building out the entity tree. The method is called as if it were a static method, so it is not associated with any particular entity data. The base implementation dynamically matches keys from the parent to the child and builds the query for the rowset based on that. RECOMMENDATION: If the key structure between the parent and child does not match up, or the rowset needs to be built using another method (i.e. a service) override this method.
public string	generateXSD(): Generates the xml schema with this node as the root. RECOMMENDATION: Do not override.
public ChildEntity	getChildEntity(string p_entityID): Retrives the ChildEntity Object based on the entity ID. RECOMMENDATION: Do not override.
protected CSUserDefaults	getCSUserDefaults(): Retrieves the CSUserDefaults. RECOMMENDATION: Do not override.
public DataKey	getDataKey(): Retrieves the dataKey for this entity. RECOMMENDATION: Do not override.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public Record	getEntPropsByPropertyName(string propname): Returns the EntityProperty Record (SCC_ENT_PROPS) based on the property name.
protected string	getFullEntityPathName()
public array of string	getIgnoreFields(): A method to get the current ignorefields, used for upgrade.
protected InstallationHCM	getInstallationHCM(): Retrieves the HCM Installation Defaults. RECOMMENDATION: Do not override.
public string	getLOVDescr(string propname): Returns the LOV Description based on the associated property name when parsed from XML.
protected string	getLOVValue(Record p_recProp, string p_code)
public number	getMaxSeqNbrSQL(): Retrieves the maximum sequence number from the database based on the keystructure of the current entity. This method is irrelevant if there are no sequence numbers for the entity. RECOMMENDATION: Override if there is a sequence number and it is not related to all the keys.
protected IEntity	getParent(): Retrieves the parent Entity. RECOMMENDATION: Do not override.
public any	getProperty(string propname)
protected any	getPropertyByMetaRec(Record propMetaRec)
protected string	getRecordName(): Retrieves the name of the current record on this entity. RECOMMENDATION: Do not override.
public string	getTagByFieldName(string fieldname): Returns the xml tag based on the field name. If metadata exists that is used, otherwise it returns the field name. This method should no longer be overriden! The xml tag is now based on the property meta data.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	hardDelete(MessageLogBase p_messageLog out): Deletes the entity object and all its child entities from the database, based on the entityDelete flag. RECOMMENDATION: Do not override this method.
protected void	hardDeleteChildren(MessageLogBase p_messageLog out): Iterates through the ChildEntityArray and issues a hardDelete against each ChildEntity, which in turn calls hardDelete on all of the entities it contains. Base implementation uses the entity registry. RECOMMENDATION: Do not override.
public boolean	hasEffdt()
public boolean	hasEffseq()
public boolean	hasKey(string p_keyname): Check if the entity has a particular field as a key. RECOMMENDATION: Do not override.
public boolean	isEqualKeys(IEntity p_entity): Allows the given entity to be compared with this entity for equality. The Base implementation compares for equality based on the underlying entity record contents. RECOMMENDATION: Override this method if you need to do custom equality checking.
public boolean	isEqualKeysRecord(Record p_rec): Allows the given record to be compared with this entity for equality. Compares for equality based on the underlying entity record content. RECOMMENDATION: Override this method if you need to do custom equality checking.
public void	populateAudit(): Populates the who column data, this method is invoked prior to save. This allows handling of the who columns to be more or less dynamic. If there are no who columns this method does nothing . RECOMMENDATION: Do not override.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	prePublish(Rowset p_changesRS out): Uses the workflowPreimage and post images of the entity to build a PSCAMA Rowset in preparation for publish. The parameter Rowset may be Null. Data should be appended to this rowset if it is NON-Null. RECOMMENDATION: Override if pre-publish logic is required.
public void	preSave(): Allows the opportunity for an entity to perform any pre Save activities. The Base implementation does nothing. Presave is automatically called from save, from the top level entity, and runs the presave logic before invoking preSave children. RECOMMENDATION: Override this method if the entity needs to do last minute preSave processing.
public void	preSaveChildren(): Iterates through the ChildEntityArray and issues a preSave against each ChildEntity, which in turn calls preSave on all of the entities it contains. Base implementation uses the entity registry. Presave is automatically called from save, from the top level entity, and runs the presave logic before invoking preSave children. RECOMMENDATION: Do not override.
public void	preSaveFirst(): Similar to presave, however, this method is called once per childEntity, prior to any other presave activity to allow for presave activity that may act across all entities in the childEntity. The Base implementation does nothing. RECOMMENDATION: Override this method if the entity needs to do last minute preSave processing crossing entity boundaries.
public void	promote(string p_emplID, MessageLogBase p_log out): Moves Data from Staging to Production Records (does not validate or save). RECOMMENDATION: This must be implemented, but is only used if the entity is staged. In the implementation make sure to call promoteChildren to cause promote to work with the registry.
protected void	promoteChildren(string p_emplID, MessageLogBase p_log out): Iterates through the ChildEntityArray and issues a promote against each ChildEntity, which in turn calls promote on all of the entities it contains. Base implementation uses the entity registry. RECOMMENDATION: Do not override.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	publish(Rowset p_changesRS, boolean p_onlineMode): Takes a PSCAMA based rowset of information to be published during workflow Uses that information to construct a Message and according to the onlineMode publishes the Message data in realtime or to the batch delay table. RECOMMENDATION: Override if publish logic is required.
public Rowset	retrieveDataRule(): Uses the entityID and Update Rule Name to read Data Update Rule information. RECOMMENDATION: Do not override.
public void	save(MessageLogBase p_messageLog out): Base implementation, calls presave and issues save for all children then a save for this entity. RECOMMENDATION: Do not override.
protected void	saveChildren(MessageLogBase p_messageLog out): Iterates through the ChildEntityArray and issues a Save against each ChildEntity, which in turn calls Save on all of the entities it contains. Base implementation uses the entity registry. RECOMMENDATION: Do not override.
public void	setChildren(): Builds out the ChildEntityArray based on the entity registry. RECOMMENDATION: Do not override.
protected void	setData(Record p_dataRec): Sets the Data record. RECOMMENDATION: Do not override.
protected void	setDataPreImage(Record p_dataPreImageRec): Sets the Data preimage. RECOMMENDATION: Do not override.
public void	setDefault(): Base implementation sets the default value for each field using the Field setDefault method if the field has not already been marked changed. RECOMMENDATION: Override if required
protected void	setEntityID(string p_entityID): Sets the entity ID, and retrieves the related information from the registry. RECOMMENDATION: Do not override.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

protected void	setKeyList(): Sets the list of key fields to private instance variable keyList. RECOMMENDATION: Do not override.
public void	setProperty(string propname, any value)
protected void	setPropertyByMetaRec(Record propMetaRec, any value)
public void	setStageMode(boolean p_stageInd): Sets the stage mode and changes the underlying record to stage/production for this entity and all children. This implementation does nothing since stage mode is not required. RECOMMENDATION: Override if the entity is Staged.
public void	setUpdateRule(string p_UpdateRule): Sets a specific update rule. RECOMMENDATION: Do not override.
public void	setUpdateRuleByTransName(string p_transName): Sets the update rule based on the transaction name in the setup table. RECOMMENDATION: Do not override.
public void	setUserContext(string p_userContext): Sets the user context for this entity. RECOMMENDATION: Do not override.
protected void	testForAudit(): Checks if the underlying record has the audit fields and sets the property hasAudit. RECOMMENDATION: Do not override.
public void	toXmlNode(XmlNode parentNode): Pushes data to the XML structure from the entity. Fields marked ignore are not Processed. Entity Element Names, wrapping of the ChildEntities or embedding is based on the entity registry. The value of properties corresponding to the fields are pushed. RECOMMENDATION: Only override if the xml for this entity has to be custom.
protected void	toXmlNodeChildren(XmlNode parentNode): Iterates through the ChildEntityArray and issues a toXMLNode against each ChildEntity, which in turn calls toXMLNode on all of the entities it contains. Base implementation uses the entity registry. RECOMMENDATION: Do not override.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	updateChildFields(string propname, array of ChildEntity cea): Sets the value of a specific property on all ChildEntites to match the current entity. RECOMMENDATION: Do not override.
public void	updateFromRecord(Record rec): Updates by matching fields in the record to fields in the underlying record on the entity. Values are only set if the value has changed. Values are set using properties, so any additional get-set logic is handled.
protected void	updateKeyFields(): Updates all the key fields of the child entities that match the key fields on this entity. RECOMMENDATION: Do not override.
public void	validate(MessageLogBase p_messageLog out): Validates the entity object if entity is not marked for deletion. NOTE: commonValidate invokes validateChildren. validateChildren is called before record based validations. RECOMMENDATION: Override this method if the entity requires custom validation. Use commonValidate to call the system edits for control over exactly which edits run, or %Super. Validate if you want them all.-- This method does the following standard system edits:- Required fields are present- Validates all 1/0 fields contain only a 1 or a 0- Validates all translate fields have a valid value- Validates all YesNo fields contain a Y or an N- Validates all prompt edit fields have a valid value.
protected void	validateChildren(MessageLogBase p_messageLog out): Iterates through the ChildEntityArray and issues a validate against each ChildEntity, which in turn calls validate on all of the entities it contains. Base implementation uses the entity registry. RECOMMENDATION: Do not override.
public void	validateFirst(MessageLogBase p_messageLog out): Allows validations to be performed on all entities under a childEntity; this is valuable in doing validation against other entities under the same childEntity. RECOMMENDATION: Override this method if the entity requires custom validation crossing entity boundaries.
public void	workflow(boolean p_onlineMode): Orchestrates the prepublish and publish methods for this entity and all children. RECOMMENDATION: Do not override.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Details

Property Details:

baseProps	public Array of string
ChildEntityArray	public Array of ChildEntity
data	public Record
dataPreimage	public Record
DO_SAVE	public boolean
dynamicProperties	protected AnyHashMap
entityDelete	public boolean
entityID	public string
EntityMetaData	protected Record
entityName	public string
hasAuditRowAddDttm	protected boolean
hasAuditRowAddOprid	protected boolean
hasAuditRowUpdDttm	protected boolean
hasAuditRowUpdOprid	protected boolean
ignoreFields	protected Array of string
keyCollection	public Array of DataKey
parent	public IEntity
parentCE	public ChildEntity
prodRecordName	public string
PropertyMetaData	protected Rowset
RECORD_NAME	protected string
SCC_ENTITY_INST_ID	public string
SCC_ROW_ADD_DTTM	public datetime

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

SCC_ROW_ADD_OPRID	public string
SCC_ROW_UPD_DTTM	public datetime
SCC_ROW_UPD_OPRID	public string
selfServiceMode	public boolean
STAGE_MODE	public boolean
UPDATE_RULE	public string
USER_CONTEXT	public string
workflowPreimage	public Record
XSDCustomNS	protected string
XSDFieldList	public Array of string
XSDNS	protected string

Constructor Details:

AbstractEntity	Parameters: IEntity p_parent: The parent of this entity, or null if this is the top of the entity tree.

Method Details:

applyDataKey	Parameters: string p_queryString Return: string
buildCSVString	Parameters: array of any p_arrValues string p_metaTag Return: string
buildPropertyMetaDataCache	N/A

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

buildXSD	Parameters: XmlNode xmlnode: The Tools XmlNode object of the parent that should be continued by this entity.
buildXSDChildren	Parameters: XmlNode xmlnode: The node of the current entity that all children see as the parent.
changeRecord	Parameters: string p_recname: The name of the record to have this entity use.
clearDataPreImage	N/A
clearDataPreImageChildren	N/A
commonValidate	Parameters: MessageLogBase p_messageLog(out): The message log to log any errors to. number p_validateArgs: Specify the validations to be run in a format like (%Edit_DateRange + %Edit_OneZero +Edit_Required).
defaultChildren	N/A
delete	Parameters: boolean p_entityDelete: True to delete this entity and False to "undelete".
deleteChildren	Parameters: boolean p_entityDelete: True to delete this entity and False to "undelete".
demote	Parameters: number p_tempID: The tempid to replace the emplid with.
demoteChildren	Parameters: number p_tempID: The Tempid to associated with all demoted data.
destroy	N/A

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

entityFieldUpdateNeeded	Parameters: Rowset p_DataUpdateRule string p_fieldName any oldValue Return: boolean
entityTypeUpdateNeeded	Parameters: Rowset p_DataUpdateRule any p_entityType boolean p_newFlag Return: boolean – True if the Data Update Rule specifies the entity type should be updated.
entityTypeUpdateRequested	Parameters: Rowset p_DataUpdateRule: The data update rule. any p_entityType: The entity type value (for example, For the Address entity, type values include: DORM, HOME, MAIL, and so on.) Return: boolean - True if any row of the Data Update Rule Detail (for this entity AND type) has an action not equal to "Do Not Update" (N).
entityUpdateRequested	Parameters: Rowset p_DataUpdateRule: The data update rule. Return: boolean - True if any row of the Data Update Rule Detail (for this entity) has an action not equal to "Do Not Update" (N).
fill	Parameters: Record p_inRec(out): The record structure that is used to populate this entity. boolean p_clearState: Whether or not to RESET this entities pre-image.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

fillChildren	Parameters: boolean p_clearState
fillFromKeys	Parameters: boolean p_clearState: Whether or not to RESET this entities pre-image.
fillLOVDescrs	N/A
fromXmlNode	Parameters: XmlNode parentNode: The parent node to add this node to.
fromXmlNodeChildren	Parameters: XmlNode parentNode: The node of the current entity that all children see as the parent.
generateRowset	Return: Rowset - a rowset populated based on the parent keys, or null if no rows are found.
generateXSD	Return: string - the XML Schema as a string.
getChildEntity	Parameters: string p_entityID: The entity ID, based on the registry, of the ChildEntity object to retrieve. Return: ChildEntity
getCSUserDefaults	Return: CSUserDefaults
getDataKey	Return: DataKey
getEntPropsByPropertyName	Parameters: string propname: The property name. Return: Record

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

getFullEntityPathName	Return: string
getIgnoreFields	Return: array of string
getInstallationHCM	Return: InstallationHCM
getLOVDescr	Parameters: string propname: The property name to get a LOV Description based on. Return: string
getLOVValue	Parameters: Record p_recProp string p_code Return: string
getMaxSeqNbrSQL	Return: number - the maximum sequence number found.
getParent	Return: IEntity
getProperty	Parameters: string propname Return: any
getPropertyByMetaRec	Parameters: Record propMetaRec Return: any

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

getRecordName	Return: string - The record name
getTagByFieldName	Parameters: string fieldname: The fieldname to search on. Return: string
hardDelete	Parameters: MessageLogBase p_messageLog(out): the message log
hardDeleteChildren	Parameters: MessageLogBase p_messageLog(out): the message log
hasEffdt	Return: boolean
hasEffseq	Return: boolean
hasKey	Parameters: string p_keyname: The name of the field to check. Return: boolean - true, the field exists and is a key, false, the field is not a key, or does not exist.
isEqualKeys	Parameters: IEntity p_entity Return: boolean
isEqualKeysRecord	Parameters: Record p_rec Return: boolean
populateAudit	

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

prePublish	Parameters: Rowset p_changesRS(out)
preSave	N/A
preSaveChildren	N/A
preSaveFirst	N/A
promote	Parameters: string p_emplID MessageLogBase p_log(out): The MessageLog object to log all errors/warnings to.
promoteChildren	Parameters: string p_emplID: The EMPLID key to associate with the promoted data. MessageLogBase p_log(out)
publish	Parameters: Rowset p_changesRS boolean p_onlineMode
retrieveDataRule	Return: Rowset - the Data Update Rule information for the current entity in a rowset (Level 0 - SCC_DUR_HDR, Level 1 - SCC _DUR_DTL).
save	Parameters: MessageLogBase p_messageLog(out): the message log
saveChildren	MessageLogBase p_messageLog(out): the message log
setChildren	N/A
setData	Parameters: Record p_dataRec: The record to set data to.
setDataPreImage	Parameters: Record p_dataPreImageRec: The preimage record to set.
setDefault	N/A

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

setEntityID	Parameters: string p_entityID: The ID of the Entity.
setKeyList	N/A
setProperty	Parameters: string propname any value
setPropertyByMetaRec	Parameters: Record propMetaRec any value
setStageMode	Parameters: boolean p_stageInd: Set to stage (true) or production (false).
setUpdateRule	Parameters: string p_UpdateRule: The specific update rule.
setUpdateRuleByTransName	Parameters: string p_transName: The transaction name.
setUserContext	Parameters: string p_userContext: The user context to set. An empty string sets the context to %UserId.
testForAudit	N/A
toXmlNode	Parameters: XmlNode parentNode: The parent node of this node.
toXmlNodeChildren	Parameters: XmlNode parentNode: The node of the current entity that all children see as the parent.
updateChildFields	Parameters: string propname: The name of the property to update. array of ChildEntity cea: The array of child entities (and their children down the tree) to set the property on.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

updateFromRecord	Parameters: Record rec: The record object to copy data from.
updateKeyFields	N/A
validate	Parameters: MessageLogBase p_messageLog(out): The container for all the messages that are generated by this method.
validateChildren	Parameters: MessageLogBase p_messageLog(out): The message log.
validateFirst	Parameters: MessageLogBase p_messageLog(out): The container for all the messages that are generated by this method.
workflow	Parameters: boolean p_onlineMode

class SCC_COMMON:Audit

Bundle 48. New application class containing methods to save audit data.

Implementing Classes:

- SCC_COMMON:ENTITY:AbstractEntity
- SCC_COMMON:ENTITY:BasicEntity
- SCC_COMMON:ENTITY:StagedEntity
- SCC_COMMON:ENTITY:ATTRIBUTES:BaseStagedAttribute

class SCC_COMMON_UTIL:BitArray

Bundle 48. New application class for parsing bit data, the way auditing setup is stored in the tools metadata record.

class SCC_COMMON:ENTITY:BasicEntity

SCC_COMMON:ENTITY:AbstractEntity – – SCC_COMMON:ENTITY:BasicEntity

Direct Known Subclasses: SCC_COMMON:ENTITY:GENERICS:GenericBasicEntity, SCC_COMMON:ENTITY:API:EntityUnitTest, SCC_COMMON:ENTITY:API:EntityRegistry, SCC_COMMON:ENTITY:API:EntityType, SCC_COMMON:ENTITY:API:EntityChild,

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

SCC_COMMON:ENTITY:API:EntityProperty, SCC_COMMON:SHOPCART:ShoppingCartItem, SCC_COMMON:ENTITY:API:EntityTypeChild

Summary

Property Summary:

public boolean	STAGE_MODE
Properties Inherited from SCC_COMMON:ENTITY:AbstractEntity	baseProps, ChildEntityArray, data, dataPreimage, DO_SAVE, dynamicProperties, entityDelete, entityID, EntityMetaData, entityName, hasAuditRowAddDttm, hasAuditRowAddOprid, hasAuditRowUpdDttm, hasAuditRowUpdOprid, ignoreFields, keyCollection, parent, parentCE, prodRecordName, PropertyMetaData, RECORD_NAME, SCC_ENTITY_INST_ID, SCC_ROW_ADD_DTTM, SCC_ROW_ADD_OPRID, SCC_ROW_UPD_DTTM, SCC_ROW_UPD_OPRID, selfServiceMode, UPDATE_RULE, USER_CONTEXT, workflowPreimage, XSDCustomNS, XSDFieldList, XSDNS

Constructor Summary:

public void	BasicEntity(IEntity p_parent): Base Constructor

Method Summary:

public void	demote(number p_tempID): This method does nothing since this entity is not staged, but it has to be implemented.
public void	promote(string p_emplID, MessageLogBase p_log out): This method does nothing since this entity is not staged, but it has to be implemented.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Methods Inherited from SCC_COMMON:ENTITY:AbstractEntity	applyDataKey, buildCSVString, buildPropertyMetaDataCache, buildXSD, buildXSDChildren, changeRecord, clearDataPreImage, clearDataPreImageChildren, commonValidate, defaultChildren, delete, deleteChildren, demoteChildren, destroy, entityFieldUpdateNeeded, entityTypeUpdateNeeded, entityTypeUpdateRequested, entityUpdateRequested, fill, fillChildren, fillFromKeys, fillLOVDescrs, fromXmlNode, fromXmlNodeChildren, generateRowset, generateXSD, getChildEntity, getCSUserDefaults, getDataKey, getEntPropsByPropertyName, getFullEntityPathName, getIgnoreFields, getInstallationHCM, getLOVDescr, getLOVValue, getMaxSeqNbrSQL, getParent, getProperty, getPropertyByMetaRec, getRecordName, getTagByFieldName, hardDelete, hardDeleteChildren, hasEffdt, hasEffseq, hasKey, isEqualKeys, isEqualKeysRecord, populateAudit, prePublish, preSave, preSaveChildren, preSaveFirst, promoteChildren, publish, retrieveDataRule, save, saveChildren, setChildren, setData, setDataPreImage, setDefault, setEntityID, setKeyList, setProperty, setPropertyByMetaRec, setStageMode, setUpdateRule, setUpdateRuleByTransName, setUserContext, testForAudit, toXmlNode, toXmlNodeChildren, updateChildFields, updateFromRecord, updateKeyFields, validate, validateChildren, validateFirst, workflow

Details

Property Details:

STAGE_MODE	public boolean

Constructor Details:

Basic Entity	Parameters: IEntity p_parent: the parent of this entity, or null if this is the top of the entity tree.

Method Details:

demote	Parameters: number p_tempID

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

promote	Parameters: string p_emplID MessageLogBase p_log(out)

class SCC_COMMON:ENTITY:StagedEntity

SCC_COMMON:ENTITY:AbstractEntity – – SCC_COMMON:ENTITY:StagedEntity

Direct Known Subclasses: SCC_COMMON:ENTITY:GENERICS:GenericStagedEntity

Summary

Property Summary:

public boolean	DO_SAVE_CHILDREN: Specifies if the children should be saved.
public string	EMPLID: Builds the xml schema section pertaining to this entity. The implementation builds and populates the schema based on the underlying record, ignoring any fields in the ignore array. Keys that roll down from the parent are not marked required on the child. RECOMMENDATION: Only override if the entity requires a custom schema. @param xmlnode: The Tools XmlNode object of the parent that should be continued by this entity.
public string	entityID: The entity ID based on the entity registry.
public string	entityName: The entity Name as specified in the registry.
public string	PROD_RECORD_NAME: The Production Record name from the registry.
public number	SCC_TEMP_ID: The property for the tempid field on stage records.
public string	STAGE_RECORD_NAME: The Stage Record name from the registry.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Properties Inherited from SCC_COMMON:ENTITY:AbstractEntity	baseProps, ChildEntityArray, data, dataPreimage, DO_SAVE, dynamicProperties, entityDelete, EntityMetaData, hasAuditRowAddDttm, hasAuditRowAddOprid, hasAuditRowUpdDttm, hasAuditRowUpdOprid, ignoreFields, keyCollection, parent, parentCE, prodRecordName, PropertyMetaData, RECORD_NAME, SCC_ENTITY_INST_ID, SCC_ROW_ADD_DTTM, SCC_ROW_ADD_OPRID, SCC_ROW_UPD_DTTM, SCC_ROW_UPD_OPRID, selfServiceMode, STAGE_MODE, UPDATE_RULE, USER_CONTEXT, workflowPreimage, XSDCustomNS, XSDFieldList, XSDNS

Constructor Summary:

public void	StagedEntity(IEntity p_parent): The Constructor

Method Summary:

public void	comparePreviousEffdt(): Compares the current record underlying this entity with previous effective dated rows. If no data has changed between this entity and the current effective dated row it prevents a save from occurring. This only matters when we are dealing with effective dated entities. RECOMMENDATION: Do not override.
public void	demote(number p_tempID): Moves data from production records to staging record. RECOMMENDATION: Do not override.
public string	getID(): Retrieves the temp ID.
protected string	getProdRecordName(): Retrieves the production record name.
protected string	getStageRecordName(): Retrieves the stage record name.
public void	promote(string p_emplID, MessageLogBase p_log out): Moves Data from Staging to Production Records (does not validate or save). RECOMMENDATION: Do not override.
public void	save(MessageLogBase p_messageLog out): Calls presave and issues save for all children then a save for this entity. RECOMMENDATION: Do not override.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

protected void	setEntityID(string p_entityID): Sets the entity ID, and retrieves the related information from the registry. RECOMMENDATION: Do not override.
public void	setID(string p_ID): Sets the temp id for this entity. RECOMMENDATION: Do not override.
public void	setStageMode(boolean p_stageInd): Sets the stage mode and changes the underlying record to stage/production for this entity and all children. RECOMMENDATION: Do not override.
Methods Inherited from SCC_COMMON:ENTITY:AbstractEntity	applyDataKey, buildCSVString, buildPropertyMetaDataCache, buildXSD, buildXSDChildren, changeRecord, clearDataPreImage, clearDataPreImageChildren, commonValidate, defaultChildren, delete, deleteChildren, demoteChildren, destroy, entityFieldUpdateNeeded, entityTypeUpdateNeeded, entityTypeUpdateRequested, entityUpdateRequested, fill, fillChildren, fillFromKeys, fillLOVDescrs, fromXmlNode, fromXmlNodeChildren, generateRowset, generateXSD, getChildEntity, getCSUserDefaults, getDataKey, getEntPropsByPropertyName, getFullEntityPathName, getIgnoreFields, getInstallationHCM, getLOVDescr, getLOVValue, getMaxSeqNbrSQL, getParent, getProperty, getPropertyByMetaRec, getRecordName, getTagByFieldName, hardDelete, hardDeleteChildren, hasEffdt, hasEffseq, hasKey, isEqualKeys, isEqualKeysRecord, populateAudit, prePublish, preSave, preSaveChildren, preSaveFirst, promoteChildren, publish, retrieveDataRule, saveChildren, setChildren, setData, setDataPreImage, setDefault, setKeyList, setProperty, setPropertyByMetaRec, setUpdateRule, setUpdateRuleByTransName, setUserContext, testForAudit, toXmlNode, toXmlNodeChildren, updateChildFields, updateFromRecord, updateKeyFields, validate, validateChildren, validateFirst, workflow

Details

Property Details:

DO_SAVE_CHILDREN	public boolean
EMPLID	public string
entityID	public string

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

entityName	public string
PROD_RECORD_NAME	public string
SCC_TEMP_ID	public number
STAGE_RECORD_NAME	public string

Constructor Details:

StagedEntity	Parameters: IEntity p_parent: the parent of this entity, or null if this is the top of the entity tree.

Method Details:

comparePreviousEffdt	
demote	Parameters: number p_tempID: The tempid to replace the emplid with.
getID	Return: string - the tempID
getProdRecordName	Return: string
getStageRecordName	Return: string
promote	Parameters: string p_emplID MessageLogBase p_log(out): The MessageLog object to log all errors/warnings to.
save	Parameters: MessageLogBase p_messageLog(out): The message log.
setEntityID	Parameters: string p_entityID: The ID of the Entity.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

setID	Parameters: string p_ID: The temporary ID value to set.
setStageMode	Parameters: boolean p_stageInd: Set to stage (true) or production (false).

abstract class SCC_COMMON:ENTITY:StagedHREntity

SCC_COMMON:ENTITY:AbstractEntity – – SCC_COMMON:ENTITY:StagedHREntity

Summary

Property Summary:

public string	EMPLID: Builds the xml schema section pertaining to this entity. The implementation builds and populates the schema based on the underlying record, ignoring any fields in the ignore array. Keys that roll down from the parent are not marked required on the child. RECOMMENDATION: Only override if the entity requires a custom schema. @param xmlnode: The Tools XmlNode object of the parent that should be continued by this entity.
public string	entityID: The entity ID based on the entity registry.
public string	entityName: The entity Name as specified in the registry.
public string	PROD_RECORD_NAME: The Production Record name from the registry.
public number	SCC_TEMP_ID: The property for the tempid field on stage records.
public string	STAGE_RECORD_NAME: The Stage Record name from the registry.
Properties Inherited from SCC_COMMON:ENTITY:AbstractEntity	baseProps, ChildEntityArray, data, dataPreimage, DO_SAVE, dynamicProperties, entityDelete, EntityMetaData, hasAuditRowAddDttm, hasAuditRowAddOprid, hasAuditRowUpdDttm, hasAuditRowUpdOprid, ignoreFields, keyCollection, parent, parentCE, prodRecordName, PropertyMetaData, RECORD_NAME, SCC_ENTITY_INST_ID, SCC_ROW_ADD_DTTM, SCC_ROW_ADD_OPRID, SCC_ROW_UPD_DTTM, SCC_ROW_UPD_OPRID, selfServiceMode, STAGE_MODE, UPDATE_RULE, USER_CONTEXT, workflowPreimage, XSDCustomNS, XSDFieldList, XSDNS

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Constructor Summary:

public void	StagedHREntity(IEntity p_parent): The Constructor

Method Summary:

public void	demote(number p_tempID): Moves data from production records to staging record. RECOMMENDATION: Do not override.
public baseType	getHRType(string p_EMPLID): Builds the HCR_PERSON_TYPES object based on the data in the current entity using the emplid provided.
public string	getID(): Retrieves the temp ID.
protected ServiceManager	getServiceManager(): Returns the HCM ServiceManager object.
public void	hardDelete(MessageLogBase p_messageLog out): Deletes the entity object and all its child entities from the database, based on the entityDelete flag. RECOMMENDATION: Do not override this method
public void	promote(string p_emplID, MessageLogBase p_log out): Moves Data from Staging to Production Records (does not validate or save). RECOMMENDATION: Do not override.
public void	save(MessageLogBase p_messageLog out): Calls presave and issues save for all children then a save for this entity. On save for production it invokes updateDeleteHcm to call the HCM Web Service. RECOMMENDATION: Do not override.
protected void	setEntityID(string p_entityID): Sets the entity ID, and retrieves the related information from the registry. RECOMMENDATION: Do not override.
public void	setID(string p_ID): Sets the temp id for this entity. RECOMMENDATION: Do not override.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	setStageMode(boolean p_stageInd): Sets the stage mode and changes the underlying record to stage/production for this entity and all children. RECOMMENDATION: Do not override.
protected void	updateDeleteHCM(boolean p_deleteFlag): This method is called from save, it should perform the update for production data by calling the HCM service.
Methods Inherited from SCC_ COMMON:ENTITY:AbstractEntity	applyDataKey, buildCSVString, buildPropertyMetaDataCache, buildXSD, buildXSDChildren, changeRecord, clearDataPreImage, clearDataPreImageChildren, commonValidate, defaultChildren, delete, deleteChildren, demoteChildren, destroy, entityFieldUpdateNeeded, entityTypeUpdateNeeded, entityTypeUpdateRequested, entityUpdateRequested, fill, fillChildren, fillFromKeys, fillLOVDescrs, fromXmlNode, fromXmlNodeChildren, generateRowset, generateXSD, getChildEntity, getCSUserDefaults, getDataKey, getEntPropsByPropertyName, getFullEntityPathName, getIgnoreFields, getInstallationHCM, getLOVDescr, getLOVValue, getMaxSeqNbrSQL, getParent, getProperty, getPropertyByMetaRec, getRecordName, getTagByFieldName, hardDeleteChildren, hasEffdt, hasEffseq, hasKey, isEqualKeys, isEqualKeysRecord, populateAudit, prePublish, preSave, preSaveChildren, preSaveFirst, promoteChildren, publish, retrieveDataRule, saveChildren, setChildren, setData, setDataPreImage, setDefault, setKeyList, setProperty, setPropertyByMetaRec, setUpdateRule, setUpdateRuleByTransName, setUserContext, testForAudit, toXmlNode, toXmlNodeChildren, updateChildFields, updateFromRecord, updateKeyFields, validate, validateChildren, validateFirst, workflow

Details

Property Details:

EMPLID	public string
entityID	public string
entityName	public string
PROD_RECORD_NAME	public string
SCC_TEMP_ID	public number

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

STAGE_RECORD_NAME	public string

Constructor Details:

StagedHREntity	Parameters: IEntity p_parent: The parent of this entity, or null if this is the top of the entity tree.

Method Details:

demote	Parameters: number p_tempID: The tempid to replace the emplid with.
getHRType	Parameters: string p_EMPLID: The emplid to use for the PERSON_TYPE. Return: baseType
getID	Return: string - the tempID
getServiceManager	Return: ServiceManager
hardDelete	Parameters: MessageLogBase p_messageLog(out)the message log
promote	Parameters: string p_emplID MessageLogBase p_log(out): The MessageLog object to log all errors/warnings to.
save	Parameters: MessageLogBase p_messageLog(out): The message log.
setEntityID	Parameters: string p_entityID: The ID of the Entity.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

setID	Parameters: string p_ID: The temporary ID value to set.
setStageMode	Parameters: boolean p_stageInd: Set to stage (true) or production (false).
updateDeleteHCM	Parameters: boolean p_deleteFlag: Specifies if the entity should be deleted.

class SCC_COMMON:ENTITY:WorkEntity

SCC_COMMON:ENTITY:AbstractEntity – – SCC_COMMON:ENTITY:WorkEntity

Direct Known Subclasses: SCC_COMMON:SHOPCART:ShoppingCart, SCC_COMMON:ENTITY:GENERICS:GenericWorkEntity

Summary

Property Summary:

public boolean	STAGE_MODE
Properties Inherited from SCC_COMMON:ENTITY:AbstractEntity	baseProps, ChildEntityArray, data, dataPreimage, DO_SAVE, dynamicProperties, entityDelete, entityID, EntityMetaData, entityName, hasAuditRowAddDttm, hasAuditRowAddOprid, hasAuditRowUpdDttm, hasAuditRowUpdOprid, ignoreFields, keyCollection, parent, parentCE, prodRecordName, PropertyMetaData, RECORD_NAME, SCC_ENTITY_INST_ID, SCC_ROW_ADD_DTTM, SCC_ROW_ADD_OPRID, SCC_ROW_UPD_DTTM, SCC_ROW_UPD_OPRID, selfServiceMode, UPDATE_RULE, USER_CONTEXT, workflowPreimage, XSDCustomNS, XSDFieldList, XSDNS

Constructor Summary:

public void	WorkEntity(IEntity p_parent): Base Constructor

Method Summary:

public void	demote(number p_tempID): This method does nothing since this entity is not staged, but it has to be implemented.
public rowset	generateRowset()
public void	hardDelete(MessageLogBase p_messageLog out)

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	promote(string p_emplID, MessageLogBase p_log out): This method does nothing since this entity is not staged, but it has to be implemented.
public void	save(MessageLogBase p_messageLog out)
Methods Inherited from SCC_COMMON:ENTITY:AbstractEntity	applyDataKey, buildCSVString, buildPropertyMetaDataCache, buildXSD, buildXSDChildren, changeRecord, clearDataPreImage, clearDataPreImageChildren, commonValidate, defaultChildren, delete, deleteChildren, demoteChildren, destroy, entityFieldUpdateNeeded, entityTypeUpdateNeeded, entityTypeUpdateRequested, entityUpdateRequested, fill, fillChildren, fillFromKeys, fillLOVDescrs, fromXmlNode, fromXmlNodeChildren, generateXSD, getChildEntity, getCSUserDefaults, getDataKey, getEntPropsByPropertyName, getFullEntityPathName, getIgnoreFields, getInstallationHCM, getLOVDescr, getLOVValue, getMaxSeqNbrSQL, getParent, getProperty, getPropertyByMetaRec, getRecordName, getTagByFieldName, hardDeleteChildren, hasEffdt, hasEffseq, hasKey, isEqualKeys, isEqualKeysRecord, populateAudit, prePublish, preSave, preSaveChildren, preSaveFirst, promoteChildren, publish, retrieveDataRule, saveChildren, setChildren, setData, setDataPreImage, setDefault, setEntityID, setKeyList, setProperty, setPropertyByMetaRec, setStageMode, setUpdateRule, setUpdateRuleByTransName, setUserContext, testForAudit, toXmlNode, toXmlNodeChildren, updateChildFields, updateFromRecord, updateKeyFields, validate, validateChildren, validateFirst, workflow

Details

Property Details:

STAGE_MODE	public boolean

Constructor Details:

WorkEntity	Parameters: IEntity p_parent: The parent of this entity, or null if this is the top of the entity tree.

Method Details:

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

demote	Parameters: number p_tempID
generateRowset	Return: Rowset
hardDelete	Parameters: MessageLogBase p_messageLog(out)
promote	Parameters: string p_emplID MessageLogBase p_log(out)
save	Parameters: MessageLogBase p_messageLog(out)

class SCC_COMMON:ENTITY:ChildEntity

Summary and Details:

Summary

Property Summary:

public Array of IEntity	childEntities: An array of all child entities of a specific type.
public string	className: The name of the appclass the entities under this childEntity implement.
public string	elementName: The element name to use for each entity.
public boolean	Embed: Specifies if the Entities this child contains have been marked for embedding in the parent entity.
public string	Encapsulate: The name of the tag to use to encapsulate all entities in this childEntity object.
public string	EntityID: The Entity ID for the entities under this childEntity.
public string	EntityName: The Entity Name for the entities under this childEntity.
public string	MaxCount: The maximum number of entities of this type allowed based on the registry.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public number	MaxSeqNbr: The current maximum sequence number, if this entity has a key that is a sequence number.
public string	MinCount: The minimum number of entities of this type allowed based on the registry.
public IEntity	parent: The parent of this ChildEntity.
public string	PROD_RECORD: The name of the production record the entities under this childEntity implement.
public boolean	STAGE_MODE: The current stage state of the entities, true=staged, false=production.
public string	STAGE_RECORD: The name of the stage record the entities under this childEntity implement.

Constructor Summary:

public void	ChildEntity()

Method Summary:

public void	buildXSD(XmlNode xmlnode): The method to build the schema for this entity, works in conjunction with the method of the same name on the entity.
public void	checkForDupes(MessageLogBase p_messageLog out): Checks for entities with duplicate keys in the childEntities array.
public void	clearDataPreImage(): Invokes clearDataPreImage on all entities in childEntities.
public void	clearEntities(): Resets the childEntities array to empty.
public void	copyFromRowset(Rowset rs): Copies the record data from a rowset into childEntities. If the keys match it, an existing entity is updated. Otherwise, an entity is added. Data is copied field by field into the equivalent properties. This means any logic in get-set is executed. In the case of update-only fields where values have changed, they are updated. Properties are copied based on matching the underlying field names.
public Rowset	copyToRowset(): Copies all the records in childEntities to a rowset.
public IEntity	createEntity(): Create an instance of the entity based on the className.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	default(): Invokes setDefault on all entities in childEntities.
public void	delete(boolean p_entityDelete): Invokes delete on all entities in childEntities.
public void	demote(number p_tempID): Invokes demote on all entities in childEntities.
public void	destroy(): Perform cleanup activity on this entity.
public void	fill(boolean p_clearstate): Invokes fill on all entities in childEntities.
public IEntity	findKeyMatchingEntity(Record rec): Finds the entity with keys that match the given record.
public void	fromXMLNode(XmlNode parentnode): Invokes fromXMLNode on all entities in childEntities.
public array of IEntity	getEntitiesByProperties(array of PStruct pstructs): Returns all entities in the childEntities array where the properties provided match.
public array of IEntity	getEntitiesByProperty(PStruct pstruct): Returns all entities in the childEntities array where the property provided matches.
public IEntity	getEntityByProperties(array of PStruct pstructs): Returns a specific entity in the childEntities array where all properties provided match.
public IEntity	getEntityByPropertiesEffdt(array of PStruct pstructs, date asOfDate): Returns a specific entity in the childEntities array where all properties provided match and the entity is the current entity as of a specific effective date.
public IEntity	getEntityByProperty(PStruct pstruct): Returns a specific entity in the childEntities array where a property matches that requested.
public number	getNextSeqNbr(string fieldname, array of IEntity entities): Provides the next sequence number for a specified fieldname out of a specific array of entities. Setting the array allows the entities to be a subset of childEntities.
public Rowset	getRowset(): Builds an empty rowset of stage or production based on the STAGE_MODE.
public void	hardDelete(MessageLogBase p_messageLog out): Invokes hardDelete on all entities in childEntities.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	preSave(): Invokes presaveFirst on the first entity in childEntities and presave on all entities in childEntities.
public void	promote(string p_emplID, MessageLogBase p_messageLog out): Invokes promote on all entities in childEntities.
public void	save(MessageLogBase p_messageLog out): Invokes save on all entities in childEntities.
public void	toXMLNode(XmlNode parentnode): Invokes toXMLNode on all entities in childEntities.
public void	validate(MessageLogBase p_messageLog out): Invokes validate on all entities in childEntities.
public void	workflow(boolean p_onlineMode): Invokes all the workflow methods in childEntities.

Details

Property Details:

childEntities	public Array of IEntity
className	public string
elementName	public string
Embed	public boolean
Encapsulate	public string
EntityID	public string
EntityName	public string
MaxCount	public string
MaxSeqNbr	public number
MinCount	public string
parent	public IEntity
PROD_RECORD	public string
STAGE_MODE	public boolean
STAGE_RECORD	public string

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Constructor Details:

ChildEntity	N/A

Method Details:

buildXSD	Parameters: XmlNode xmlnode: The XmlNode object from the parent to add the schema elements to.
checkForDupes	Parameters: MessageLogBase p_messageLog(out)
clearDataPreImage	N/A
clearEntities	N/A
copyFromRowset	Parameters: Rowset rs: The rowset to copy data from.
copyToRowset	Return: Rowset
createEntity	Return: IEntity
default	N/A
delete	Parameters: boolean p_entityDelete: The delete state passed from the parent to the child.
demote	Parameters: number p_tempID: The tempID to set on the records when the data is demoted.
destroy	N/A
fill	Parameters: boolean p_clearstate: Specifies if the preImage should be set.

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

findKeyMatchingEntity	Parameters: Record rec: The record to use when searching for a match. Return: IEntity
fromXMLNode	Parameters: XmlNode parentnode: The xml node that entities in this childEntity should be parsed from.
getEntitiesByProperties	Parameters: array of PStruct pstructs Return: array of IEntity
getEntitiesByProperty	Parameters: PStruct pstruct: An object that provides the property name and value to search for. Return: array of IEntity
getEntityByProperties	Parameters: array of PStruct pstructs Return: IEntity
getEntityByPropertiesEffdt	Parameters: array of PStruct pstructs date asOfDate: The date to check against the effective date. Return: IEntity

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

getEntityByProperty	Parameters: PStruct pstruct: An object that provides the property name and value to search for. Return: IEntity
getNextSeqNbr	Parameters: string fieldname: The field name of the sequence number. array of IEntity entities: The array of entities to search through to figure out the max sequence number. Return: number - the next available sequence number.
getRowset	Return: Rowset - the rowset generated.
hardDelete	Parameters: MessageLogBase p_messageLog(out): The messageLog object passed from the parent to the child to log any errors or warnings.
preSave	N/A
promote	Parameters: string p_emplID: The employee ID to set on the records when the data is promoted. MessageLogBase p_messageLog(out): The messageLog object passed from the parent to the child to log any errors or warnings.
save	Parameters: MessageLogBase p_messageLog(out): The messageLog object passed from the parent to the child to log any errors or warnings.
toXMLNode	Parameters: XmlNode parentnode: The xml node that entities in this childEntity should be added to.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

validate	Parameters: MessageLogBase p_messageLog(out): The messageLog object passed from the parent to the child to log any errors or warnings.
workflow	Parameters: boolean p_onlineMode: The online mode.

class SCC_COMMON:ENTITY:EntityRegFactory

Summary and Details:

Summary

Method Summary:

public IEntity	createEntity(string entityID, IEntity parent)

Details

Method Details:

createEntity	Parameters: string entityID IEntity parent Return IEntity

interface SCC_COMMON:ENTITY:CODEGEN:EntityGeneratorInterface

Implementing Classes: SCC_COMMON:ENTITY:CODEGEN:EntityGeneratorBase

Summary

Property Summary:

public string	BaseAppClass: The base appclass, this is populated based on the entity type when the code generate button is pressed.
public string	ClassName: The class name, which is populated by the entity registry when the code generate button is pressed.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public string	EntityID: The entity id, which is populated by the entity registry when the code generate button is pressed.
public boolean	hasProduction: Specifies if this entity has a production record.
public boolean	hasStage: Specifies if this entity has a stage record.
public Array of string	ignoreFields: Specifies the names of properties implemented on the parent entities that should not be generate on the current entity.
public Rowset	rsProperties: Passes in the rowset of SCC_ENT_PROPS related to the entity.

Method Summary:

public void	BuildUI(Rowset rs): Builds the UI of check boxes in the given rowset, SCC_ER_CGEN_VW, this rowset has 2 fields, SCC_ER_CGEN_ARG, the check box to be populated by the user and SCC_ER_CGEN_LBL the label displayed relative to the check box. Using that this code can generate as many yes/no questions as needed for code generation.
public string	GenerateCode(): The process called to generate the code based on the provided properties and UI input.
public void	ParseUI(Rowset rs): Retrieves the data from the UI built in BuildUI, SCC_ER_CGEN_VW, this rowset has 2 fields, SCC_ER_CGEN_ARG, the check box to be populated by the user and SCC_ER_CGEN_LBL the label displayed relative to the check box. This code should retrieve the arguments.

Details

Property Details:

BaseAppClass	public string
ClassName	public string
EntityID	public string
hasProduction	public boolean
hasStage	public boolean
ignoreFields	public Array of string

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

rsProperties	public Rowset

Method Details:

BuildUI	Parameters: Rowset rs: The rowset for SCC_ER_CGEN_VW to be populated.
GenerateCode	Return: string - returns the generated code as a string for display.
ParseUI	Parameters: Rowset rs: The rowset for SCC_ER_CGEN_VW to be parsed.

interface SCC_COMMON:ENTITY:PROPERTY:PropertySyncInterface

Implementing Classes: SCC_COMMON:ENTITY:PROPERTY:PropertySyncBase

Summary

Property Summary:

public string	appclass: The application class for the specified entity, populated pre-sync.
public string	entityID: The entity ID for the specified entity, populated pre-sync.
public string	EntityType: The entity type ID for the specified entity, populated pre-sync.
public MessageLogBase	Log: All property changes should be written to the log, which is then displayed to the user when the sync is done using All Property Sync.
public string	ProdRecord: The production record as specified on the entity, populated pre-sync.
public rowset	rs: The rowset of current entity properties based on record SCC_ENT_PROPS, populated pre-sync.
public string	StageRecord: The staging record as specified on the entity, populated pre-sync.

Method Summary:

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public string	LogAsString(): Returns the Log as a string.
public void	save(): The method called to save the updated properties, this is called from the all property sync process, for the entity registry component this is not used.
public void	updateProperties(): The method called to update the properties in the rowset, based on the records.

Details

Property Details:

appclass	public string
entityID	public string
EntityType	public string
Log	public MessageLogBase
ProdRecord	public string
rs	public Rowset
StageRecord	public string

Method Details:

LogAsString	Return: string
save	N/A
updateProperties	N/A

abstract class SCC_COMMON:ENTITY:LOG:MessageLogBase

Direct Known Subclasses: SCC_COMMON:ENTITY:LOG:TempMessageLog

Summary

Property Summary:

public boolean	isError

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public boolean	isInformation
public boolean	isReconcileError
public boolean	isWarning
public string	messageContext
protected Array of MessageEntry	MessageEntries

Constructor Summary:

public void	MessageLogBase()

Method Summary:

public void	append(MessageLogBase p_messageLog)
public void	buildMsgSegment(XmlNode msgsnode)
public void	clear()
public Message	generateFaultMsg(string p_operationName)
public void	generateValidationMsg(XmlNode p_node)
public MessageEntry	getMessageEntry(number p_index): Returns the MessageEntry Object at the given index.
public number	length(): length Current length of the message log.
public string	read(number p_index): read - Read an entry from the Message Log. @Return String - The message entry at the given index or blank if index does not exist.
public void	save()
public void	writeEntry(MessageEntry p_entry)

Details

Property Details:

isError	public boolean

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

isInformation	public boolean
isReconcileError	public boolean
isWarning	public boolean
messageContext	public string
MessageEntries	protected Array of MessageEntry

Constructor Details:

MessageLogBase	N/A

Method Details:

append	Parameters: MessageLogBase p_messageLog
buildMsgSegment	Parameters: XmlNode msgsnode
clear	N/A
generateFaultMsg	Parameters: string p_operationName Return: Message
generateValidationMsg	Parameters: XmlNode p_node
getMessageEntry	Parameters: number p_index Return: MessageEntry
length	Return: number

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

read	Parameters: number p_index Return: string - The message entry at the given index or blank if index does not exist.
save	N/A
writeEntry	Parameters: MessageEntry p_entry

class SCC_COMMON:ENTITY:LOG:MessageEntry

Summary and Details:

Summary

Property Summary:

public string	Context
public number	MsgID
public number	MsgSet
public Array of string	ParmArray
public Array of MEProp	Properties
public number	Severity
public number	Severity_Error
public number	Severity_Info
public number	Severity_ReconcileError
public number	Severity_Warning

Constructor Summary:

public void	MessageEntry()

Method Summary:

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

public void	AddProperty(string p_UID, string p_Prop)
public void	DataPopulateV1(number p_MsgSet, number p_MsgID, number p_Severity, array of string p_ParmArray, string p_Context, string p_UID, string p_Prop) Quick population of a messageEntry.

Details

Property Details:

Context	public string
MsgID	public number
MsgSet	public number
ParmArray	public Array of string
Properties	public Array of MEProp
Severity	public number
Severity_Error	public number
Severity_Info	public number
Severity_ReconcileError	public number
Severity_Warning	public number

Constructor Details:

MessageEntry	N/A

Method Details:

AddProperty	Parameters: string p_UID string p_Prop

 Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

DataPopulateV1	Parameters: number p_MsgSet: The message set of the error. number p_MsgID: The Message Id, based on the message set for the error. number p_Severity: The severity of the error should be set using Severity_ReconcileError, Severity_Error, Severity_Warning, Severity_Info. array of string p_ParmArray: The array of parameters for the message specified, if no parameters are needed null can be passed. string p_Context: Specify the context of the MessageEntry. If no context is set, the MessageLog defaults it to writeEntry. string p_UID: The EntityInstanceId that this messageEntry should be attached to, if multiple EntityInstances are involved use AddProperty to add the additional properties. string p_Prop: A specific property (use the xml tag) under an instanceid that the messageEntry is related to, if multiple properties are involved use AddProperty after DataEntry to add additional properties.

class SCC_COMMON:ENTITY:LOG:MEProp

Summary and Details:

Summary

Property Summary:

public string	EntityInstanceId
public string	Property

Details

Property Details:

EntityInstanceId	public string
Property	public string

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.