
Beyond the Outer Limits with User Extensions

Douglas Scherer
Core Paradigm

Peter Koletzke
Millennia Vision Corporation

While Oracle Designer provides lots of support for the activities you perform when creating a system, there are always things you want to do that Oracle Designer does not do by default. User Extensibility is one of the features that Oracle Designer provides to allow you to fulfill many non-defaults needs. This feature gives you the ability to create and modify the basic structures used to store your data in the repository.

This presentation explains how User Extensibility works; what you need to know about the repository meta-model to be most effective; the benefits and drawbacks of extensions; the mechanics of getting data into the extended elements; and importing and exporting the extensions between repository instances. It steps through the extension process and also mentions some of the potentially dangerous side-effects with their workarounds.

What Are User Extensions?

User Extensions are additional properties or element types that you add to an existing repository. You use the Repository Administration Utility (RAU) to add, view, export, and import these extensions. This allows you to customize the repository to hold any elements you really need in a project that are not included by default.

The example used in this paper is that of the Requirement element and its links to the Function element. You can use this extension to track all the requirements gathered in the Strategy and Analysis phase and create mappings to functions and entities that you enter as usual in the Analysis phase. You can also map Requirements to Modules and Tables in the Design phase so you can cross-check that the design elements handle the stated requirements of the system.

To be effective with User Extensions, you have to understand two related concepts. You need to be familiar with the high-level architecture of the repository so you can prepare to make changes to it, and you have to learn to use the RAU User Extensibility features.

Elements, Associations, and Text Types

You will find it useful to start by reviewing the repository architecture. These discussions refer to the meta-model, or the "model of the system model," and how to relate it to elements and associations in RON. Keep in mind that the discussion pertains to element and association types, not instances. For example, it refers to the Entity element type, which RON represents as the Entity node in its hierarchy. This type can have many instances, which RON represents as the entities themselves: for example, PERSON, ORGANIZATION, and PHONE. Also keep in mind that the discussion does not pertain to the tables or views where Oracle Designer stores or presents these items, but only to the concepts about and structures for the data.

All structures for the data in the repository can be divided into three main conceptual types. They are listed here with the symbol that RAU uses to represent them.



Element types, which include Functions, Modules, Entities, Tables, and Columns. These are both the high-level nodes (for example, Tables) and some lower-level nodes (for example, Columns for those tables) you see in RON. Each element type consists of a set of properties, discrete facts about the element type. For example, the table *Name* and *Display Title* are properties of the Table element type. RON represents properties as the rows that appear in the properties window.



Association types, which include Function Entity Data Usage, Module Function Usage, Table Entity Usage, and Tables Implemented in Schemas. These relationships are defined between particular element types and are represented in RON's *Usages* nodes. Association types, like element types, are composed of a set of properties, which appear as rows in the RON property sheet.



Text types, which include Description, Notes, Select Text, and Where/Validation Condition. Unlike element and association types, text types do not include discrete properties, but consist of one area which can contain free-format text. Text types appear in RON as properties of an element or association, but they are conceptually different--and separate--types.

Each repository type can be defined as a combination of properties and text type usage. Text type usage indicates the kind of text you are storing for an element or association--such as *Description*, *Notes*, or *Where/Validation Condition*. RON also shows the text type usages as rows in the properties window even though they are stored apart from the element instance. Text types use text type usages to indicate which element types use them. Figure 1 shows this system of types and properties.

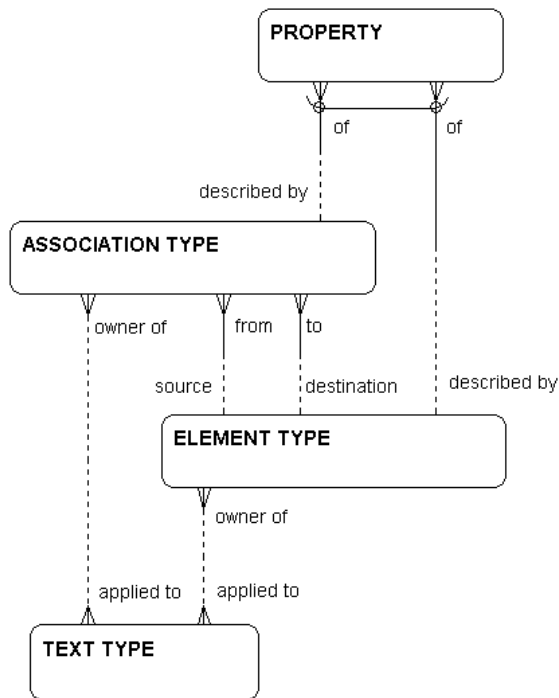


Figure 1: The Repository Types Meta-Model

This figure represents the type classes and property class which serve as structural definitions of the actual repository type and property. For example, the element type entity on this diagram represents the class or category of all repository element types. This class is instantiated in the repository as objects like Entities, Business

Functions, Table Definitions, View Definitions, and Module Definitions. All of these are members of the same class: the element type.

Similarly, the association type serves as a class for all relationships between element types. For example, the association type on this diagram represents the class or category of all repository association types. The association type class is instantiated in the repository as objects like Function Entity Usages, Module Network, and Table Entity Usage. All of these are members of the same class: association type. Since there are multiple instantiations of the association type, implicitly there are multiple instantiations of the relationship lines that link element type to association type.

The text type on this diagram represents the class or category of all repository text types. Instantiations of this class are repository objects like Description, Notes, PL/SQL Block, and User/Help Text. As with the association type relationships, the relationships from the text type are instantiated in the repository when a text type links to an association or element type.

Finally, the Property entity on this diagram represents the class of all repository properties. Instantiations of this class are properties of repository types like *Name* for the Function element type, *Short Name* for a Business Function element type, and *Prompt* for a Column element type. Since there are many properties for each element type, the single relationship lines on this diagram that link the Property class to the Element and Association classes represent multiple instantiations.

Creating User Extensions

The task of creating user extensions has a deceptively easy learning curve. If you follow the steps described in this paper in conjunction with reviewing the Oracle Designer help topics you will be able to extend your repository and support your additional repository needs with very little effort. Heed all the cautions, tips, and notes. Some of the actions you take in extending the repository cannot be easily reversed.

What You Can Extend

Using this system of repository types and their properties, Oracle Designer provides all the nodes in the hierarchy that you see in RON. You can customize the repository with RAU's User Extensibility feature to add up to 500 entirely new element and association types or an unlimited number of text types and usages; you can also add up to 20 new properties for each type. Adding element and association types uses one of the predefined but unassigned types named E0 to E499 and A0 to A499, for elements and associations, respectively. Adding text types defines a new type with any unused name. Adding properties is just a matter of defining a new name and datatype for one or more of the 20 unassigned Usrx properties (called User-Defined Property 0 through User-Defined Property 19) already attached to each element type but not shown in RON.

Caution

Be careful when defining properties of existing elements. If you make a user-extended property mandatory, you will be unable to insert the definition through a property dialog or other tool that does not display the extended property. For example, assume that you add a user-extended property called Table Usage and make it mandatory. If you try to draw a new table in the Server Model Diagram, the action will fail because the new mandatory property has no value and there is no way to fill it in using the property dialog.

If you decide to define extra properties for an existing element or association, remember that the properties you see in RAU are only the extensible ones (and *Name* and *Comments*), so be sure a property whose name matches the one you want to create does not already exist.

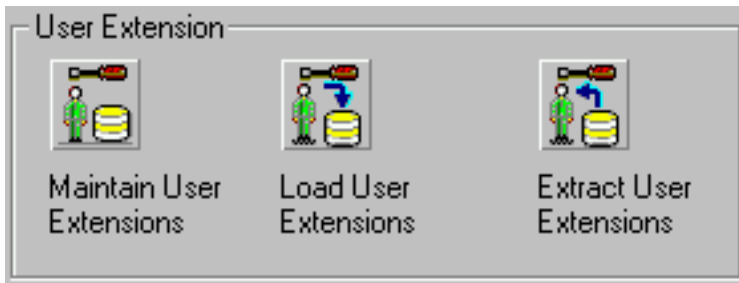
Tip

The repository stores audit information on each definition. However, audit-related properties are not displayed by default. In RON, choose **Options→Customize Properties Palette** from the menu. Then click the check box for *Show Audit Properties*. Once that is checked, you will be able to see the name of the person who created and last changed any properties of the element type and when these actions occurred.

How to Define User Extensions in RAU

When reviewing the User Extensibility window you will see that each repository element type can contain definitions that are either System or User Defined. System-Defined types are created as part of the Oracle Designer base product configuration. User-Defined types are types that you create as extensions to the repository.

RAU's User Extension area shown below, provides three tools to accomplish the following:



- *Maintain User Extensions* for creating, changing, and viewing user extensions
- *Load User Extensions* for bringing user extensions into the repository from an export file created with the next utility
- *Extract User Extensions* for creating the export file that stores the user extensions

The last two tools are used only for importing and exporting the extended definitions and are well documented in the help system. You use the first tool to perform the main work of creating and modifying user extensions.

Steps for Creating a User Extension

There are four major steps to creating a User Extension property or type.

1. Make a plan.
2. Define the new objects.
3. Publish the extensions.
4. Reconcile grants.

Remember that you can press the Help button to get more information on the process and details on particular steps as you work.

1. Make a Plan

Make a plan of the steps you will take and, if desired, draw an ERD of the new items like the one in Figure 2. If you have elements that have a relationship or link, you'll need to define an association type (for example, REQUIREMENT FOR FUNCTION).

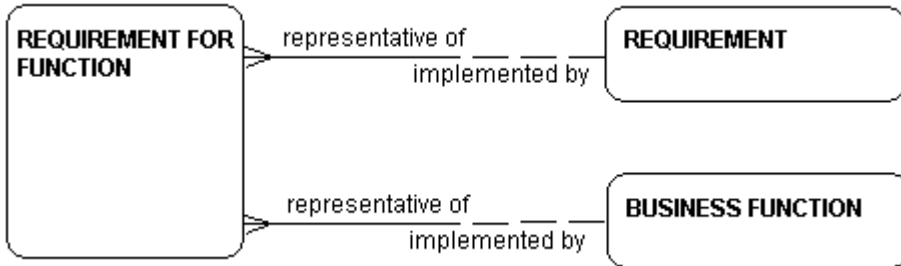


Figure 2: ERD of proposed element and association

Caution

When you publish a new type or property, you will change all application systems in the repository.

Tip

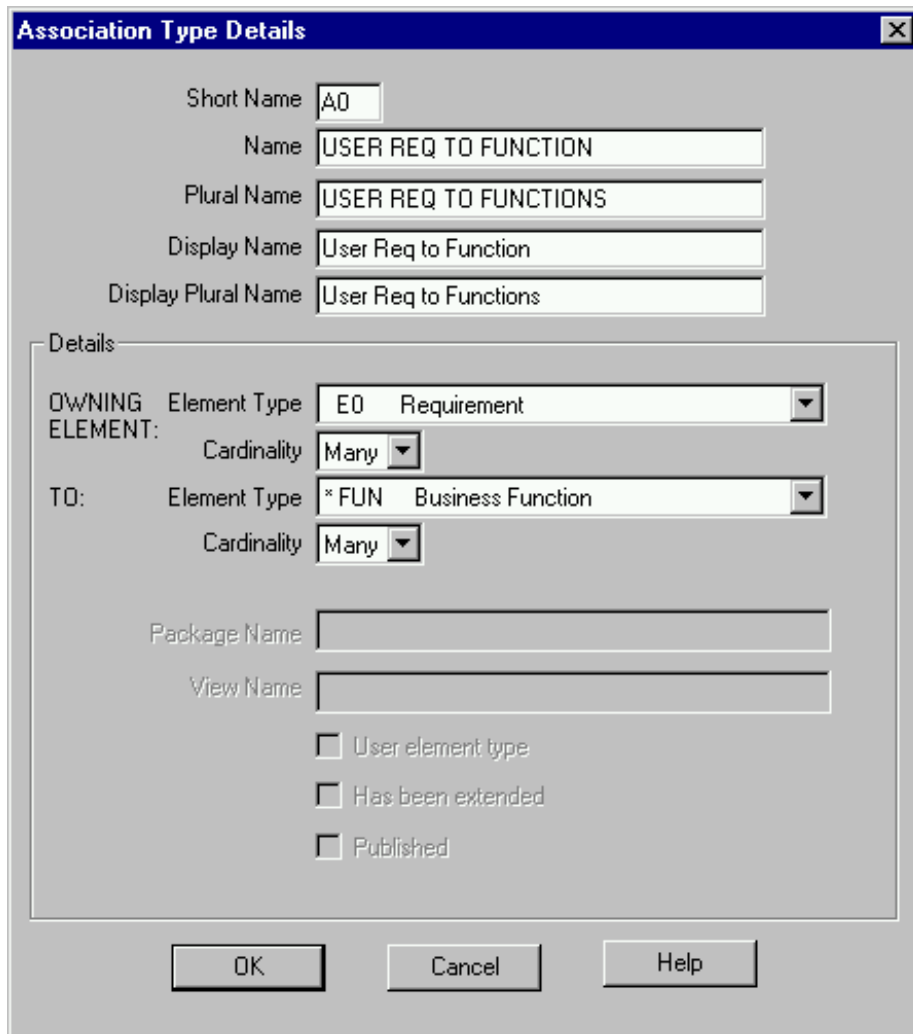
Perform a full-user repository export to create a .DMP file via RAU *before* you start working on the extensions. And try to work on the extensions when no one else is using the repository. If you make a serious mistake, you can delete the repository and import the .DMP file to restore the original state of the repository (all RAU tasks) or you can Unpublish the extension if it does not contain data.

2. Define the New Objects

Click the Maintain User Extensions button in RAU. If you want to create a new type (element, association, or text), expand the hierarchy so that the User Defined node is available. Choose that node. Click the New button (or choose New from the right-click menu) to display the Details dialog for the selected type and fill in all items there. Click OK. Note that the new type's symbol on the User Extensibility window is red, indicating that it is unpublished. Do not publish yet.

The element and association types are automatically named with a letter and number (like E7). You should not change these names; they will help you clearly identify these items as user extensions when you look at the list. This reference name will not be visible anywhere other than in this utility (and in the base table, of course). Elsewhere, you will refer to the new item by the name you assign. Be sure to define properties for the new element. For the Requirement example, you also need to define a new association type for User Req to Function. The Association Type Details dialog requires you to specify the cardinality of the link between the associated elements.

Specify the cardinality carefully--you usually want many-to-many associations so the link will appear under both elements. Figure 3 shows the Details dialog for the new User Req to Function association type. Note that the Business Function (FUN) has an asterisk to its left. This indicates that FUN is a published element. Conversely, REQ, the extended element, does not have an asterisk to its left, indicating that the REQ extension has not yet



been published.

Figure 3: Definition of the User Req to Function association type

After defining the element and association types, you can assign names to the extensible Usrx properties. Expand the new element and its Properties subnode. Then, select the Usrx property you want to use and choose Edit. Fill in the items on the Property Details dialog and click OK. Do not publish yet.

If you want to create a new text type usage, expand the node of the type you want to add it to until you see the Extended Text Type Usages node. Choose that node, click the New button (or select New from the right-click menu), and fill in the Details dialog that appears. Click OK and do not publish yet. For the Requirement example, we will not define any additional text types.

3. Publishing the Extension

Even though you can unpublish later it is a good idea to back up the repository (as mentioned above) so you can restore it if needed after the extension process. Click the Publish button for each type after you have checked your

work. If you are defining a new type, you will be prompted to indicate whether you also want to publish unpublished properties. This means you are confirming that you want the Urx properties for the new extension as well as the new extension; normally you would click OK because you do want properties as well as the extension. Then you will be prompted as to whether you want to run the data definition language (DDL) script to generate the views and packages. This is not an optional step, but you can perform it later with the Submit DDL button if you wish.

Note

Running the DDL is not necessary if the only change you have made is to add properties to an existing element type, but you do have to perform the Full Reconcile described in step 4 below.

4. Reconcile Grants

The last step in the process is to update the grants and synonyms for repository users so they can see and use the new extensions. Click the Recreate button in the RAU Repository Maintenance area. Choose Full Recompile, and RAU will run scripts to update user access so users can use the new elements or properties.

Unpublishing the User Extension

If you want to delete or edit the user extension, you need to unpublish it first. You can unpublish the extension only if you have not used it to add element definitions to the repository. In other words, if you start using the extension by defining elements, you will not be able to unpublish until you remove those element definitions. In addition, if there are other element or association types that reference the one you want to remove or change, you must detach them before unpublishing. Once those conditions are met, you click on the element, association, or text type you wish to unpublish and click the Unpublish button in the User Extensibility dialog. If there are user-defined properties, a dialog will appear where you will make a choice whether to unpublish those properties. If you want to edit the extension, you can do so at this time by clicking the Edit button. If you want to delete the extension, click the Delete button. If you get an error message, you may need to break the links to whatever element type is linked. If the delete is successful it will remove the supporting API view and package.

How to Access the User Extension

Your repository will now have new elements and properties. You will be able to view all of the new extensions in RON, while only some extensions are available in the Matrix Diagrammer and the Design Editor. The extensions are not available in any of the other Oracle Designer tools.

In RON be sure you specify that you want to see user extensions in the Hierarchy window by selecting the User Extensions check box in the **View→Include Navigator Groups** dialog before you open the application system.

When you open a new diagram in the Matrix Diagrammer you will see your element types on either side of your extended associations. Of course, if you did not add an extended element into an association you will not be able to see it the Matrix Diagrammer.

After extending the repository you will want to view the extensions in RON, Design Editor, and the Matrix Diagrammer. If during the publishing process the tool you are going use to check your extension was open, you should close it and reopen it before proceeding so that it picks up the repository changes. Start by checking RON for the new node or property in an element. If it is an extended property it will appear in RON if you have set the View as described above. The property Palette will show the new Property.

If you also defined an association, you will not see that node until you create an element of one of the types. Figure 4 shows the new Requirement element with an association node under one of its instances.

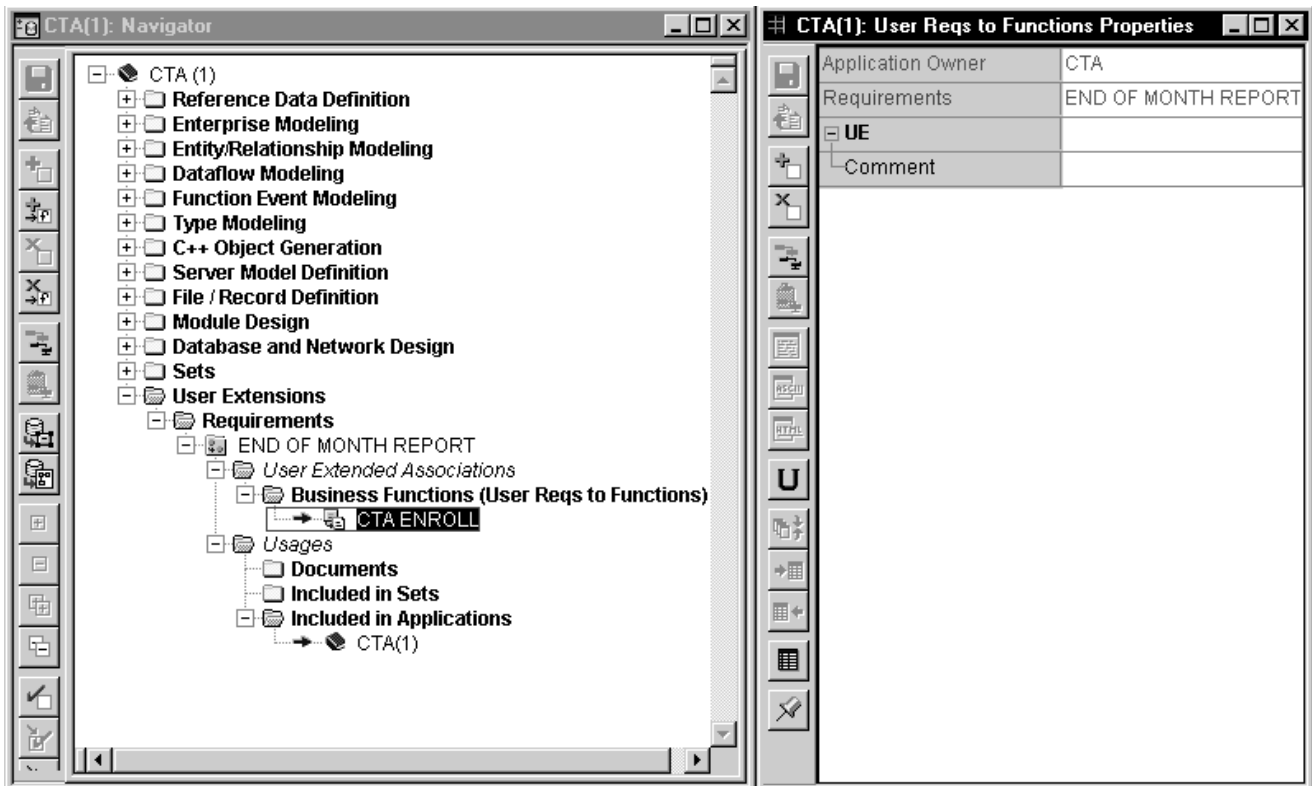


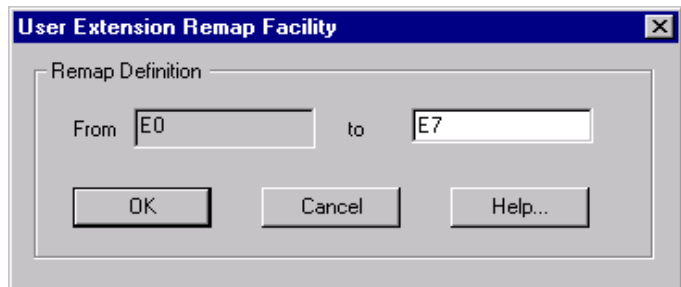
Figure 4: New Requirement element

Import/Export

If you need to export an application system from an extended repository into an archive file (through **Application→Archive** in RON), you have to handle the user extensions carefully. If the user extensions you created in the source repository do not exist in the target repository, you have to export them from the source repository using the RAU function *Extract User Extensions*. This creates a .DMP file that you can load into the target repository with the *Load User Extensions* RAU function. You can then proceed with the Archive and Restore as usual and your extended elements will be loaded.

Remapping the User Extension

A conflict will occur when the target repository already contains the same named extension (say E0) as the source repository but these elements are used for different purposes. The same type of problem applies to extended properties that exist in both repositories but are used differently. The rule-of-thumb here is to be sure the



repository you are importing into has exactly the same extensions as the repository from which you are exporting.

If you wish to rename the Short Name of an extension, you can use the Remap facility. To perform the Remap, choose the item that you want to change and click the Remap button. A dialog will appear similar to the one shown to the left. When you have entered

the new name click the OK button. The change is now complete. There is no need to publish or Submit DDL.

About the Authors

Douglas Scherer (Certified Oracle Database Administrator under Chauncy and Oracle exams) is President of Core Paradigm, a firm providing database and IT solutions through consulting, training and seminars focusing primarily on Oracle environments. He is a frequent speaker at Oracle conferences and user group meetings and recently appeared in a PBS series examining technology and educational trends into the 21st century. He is the Contributing Editor to the *Oracle Designer Handbook, Second Edition*, published by Osborne McGraw-Hill and is in the process of co-authoring several books including *Oracle 8i Tips and Techniques*, also published by Osborne McGraw-Hill, and two books in the new *Oracle Interactive Workbook Series* published by Prentice Hall. He is also the Chair of the database track at Columbia University's Computer Technology and Applications (CTA) Program. www.coreparadigm.com

Peter Koletzke is a practitioner and self-proclaimed evangelist for Oracle Designer and Developer. He is a Consulting Manager and Principal Instructor for Millennia Vision Corporation, a strategy-through-implementation Oracle-solutions provider that concentrates on custom development, Oracle Applications, data warehousing, Oracle training, supply chain management, electronic commerce, finance strategy, and web applications in California's Silicon Valley. He is the Director of Web Initiatives for the IOUG-A, a frequent contributor to national and international newsletters and users group conferences, and co-author, with Dr. Paul Dorsey, of the *Oracle Designer Handbook, Second Edition*, published by Osborne McGraw-Hill, Oracle Press which was the source for part of this paper and is available also in Korean and Spanish. www.mvsn.com