

A woman with blonde hair, wearing a beige blazer over a striped shirt, is seated at a conference table. She is holding a white document and looking towards the camera with a slight smile. In the background, two other people are seated at the table, working on documents. The setting appears to be a modern office or conference room with large windows.

## USoft Product Information What's new in USoft 8

## Introduction

This document provides you with an overview of the main enhancements in USoft 8. The focus of attention in this version has been on manageability, scalability and openness, and the tool interface. Many of the enhancements included in USoft 8 are the result of customer requests and feedback. Improvements and new features in this version are outlined in this fact sheet. Note that some of these enhancements were introduced during ongoing development of version

## Modular Development

If you have developed one large USoft application, you may incidentally run into problems with managing and delivering new projects. Because of the dependencies between projects, small projects may have to wait for larger projects. In some organizations, this results in an undesirably low frequency of delivery of application versions, for example once or twice a year. As a solution, you may serialize these projects and assign time frames, with the disadvantage that if one of the projects does not deliver in time, all following projects have to wait.

Splitting up your application into several modules which can be developed and maintained separately can make your development more flexible. It has many advantages:

- You can create independent releases for the modules.
- Faster delivery of smaller projects.
- New teams only need to build up knowledge of one (small) module.

This is exactly where modular development and internal interfaces come in. USoft applications can then be developed in separate modules and run as one application. It allows developing independent modules. The end user application allows for enterprise rules and an integrated GUI. Migrating to such an environment is regarded a smaller step and can probably be done more gradually.

Most of the time, modules will not be completely independent of each other. Some tables for example might be used by more than one module. To solve this problem, USoft Definer supports the use of Internal Interfaces on the following objects:

- Database Tables, Logical Views and Component Tables.
- Domains
- Batch Jobs
- Decisions

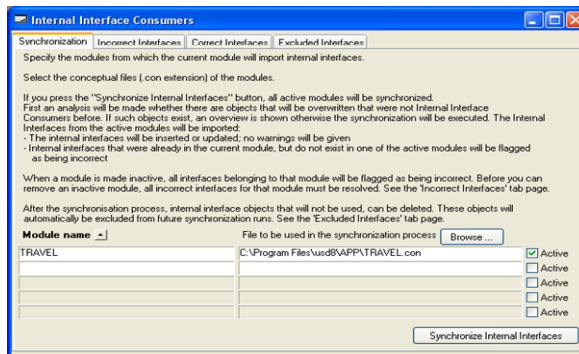
## Defining Interface Tables

In a Definer, you can define a table as being available for use by another module as an Interface Table, including one or more of its columns. Once the distributable definition files, also known as flat files, are created for this (provider) module, this table definition can then be used from another, consumer module.

This table definition can be seen as a shared table between modules. The table is owned by the provider and used by the consumer. The table columns used in the interface should not be changed as there are other modules that use it and depend on it. If the table is changed, the other modules have to be changed and synchronized too. In the table, it is possible for each column to specify if it is part of the interface or not. Columns that are not part of the interface can be freely deleted or added to the table without affecting the interface. Other modules then will not have to be changed and synchronized.

When running the application, the definitions of all modules are read and merged into one application. Rules are also merged. So, all constraints will fire when necessary, no matter in which module they were defined.

In the Internal Interface Consumers window, you can specify modules and synchronize internal interfaces. After synchronizing, you can view the correct, incorrect, and excluded interfaces on the appropriate tab pages. The help texts on the tab pages provide all information you need:



## Modules

A USoft application consists of one or more modules. They are developed in separate repositories. For each module, flat files have to be generated before the interfaces defined in it can be used from other modules. At runtime, the definitions of all modules are gathered into one application.

One module will be deployed as the application, this module has the same name as the application. This main module contains the specification of the other modules used in the application

## Web Designer and AJAX Technology

USoft web applications use AJAX technology. In general, this results in a reduced network and server load since lots of data will be stored on the client. This results in a better user experience and better performance compared with previous USoft versions.

AJAX (Asynchronous JavaScript and XML), is a group of interrelated web development techniques used to create interactive web applications. Using Ajax, USoft web applications retrieve data from the server asynchronously in the background, without the need to refresh a web page, and without interfering with the display and behavior of the existing web page.

All data in Web Designer pages is linked to data sources. The advantage is that each control that extracts its data from a data source automatically benefits from the AJAX technology.

Web pages only retrieve data parts, all server actions (except the executeSQLStatement action) are executed asynchronously using AJAX technology - there is no longer a need to retrieve entire web pages. This results in an improved look and feel for your web application.

## Tree Control in Web Designer

A TreeControl object is now available for use in the Web Designer. This makes it possible to include tree views in web applications.

## Log File Location

For security and convenience reasons all USoft log and temporary files are now stored in one central location, which you can specify during USoft setup. A central location makes it easier to find log files, and is useful for security, as it is important to know where USoft needs write permission, especially in a Web environment.

The folder that you specify during setup will have two subfolders: usoft\_logs and usoft\_temp. Within the usoft\_logs folder are several subfolders, to contain logs from USoft Batch, the Create Tables utility, the Rules Service, and so on. All temporary files generated by USoft are placed in the usoft\_temp folder.

### Debugging Web Service Components

For debugging purposes, you can make a BenchMark profile when calling a web service component. This profile then contains the soap messages that are sent to and received from a web service. The profiler is very useful when developing a web service component table. Using the Profiler, you can debug the input and output transformations.

When a web service provider is created, four files are generated: two .asp, one .inc and one .xml file. The .inc file contains a DEBUG parameter. The default value of this parameter is No. You can use the DEBUG parameter to suppress or display messages regarding the setup of the web service provider (e.g. the rules service is not started or is using a different port number, the user/password are not valid, the rules service is not started on a specific application, the MSXML component is not installed, and so on)

### Web Service Error Handling

You can improve the default error message behavior by using customized XML error messages or USoft XML error messages (or a combination). Both types of XML error messages apply an XSL transformation to the default error message.

Customized XML error messages are defined per web service provider method, and generate a wsdl:fault element in the WSDL schema. This makes the structure of the error message available to the web service client.

### Integration of Batch Definer and Definer

All USoft Batch Definer functionality has been integrated into the USoft Definer. The repositories for both applications have also been merged. All references to USoft Batch as a separate application have been removed from the toolset. A new "Batch" tab page has been added to the Definer catalog for easy access to the USoft Batch related features within the integrated Definer. Other changes you will notice in the Definer due to this integration include:

- You can now create Batch (.job) files from the Create Flat Files dialog.
- Menus have been adapted to include Batch features.
- Batch jobs are included in the list of business objects under the Teamwork tab of the catalog.

### Calling Batch Jobs from Constraints

It is now possible to call batch jobs from a constraint. The following restrictions apply:

- The job (or sub-jobs of the job) cannot have action tasks.
- The job cannot be an interface job.
- Currently, the job can only be used in an 'invocative' constraint, that is: only as the TOPMOST component in an invoke, for example:

```
invoke batchrunner.mybatch with select col1, col2 from t1 where pk=73
```

- The job can only be called using the method call syntax, NOT using the StartJob / RunJob methods.
- Any (pre)commit fired from the job will NOT be executed. This is also the case for record-level commits for import/SQL tasks.

### Batch Cross-Referencing

USoft 8 introduces cross-referencing for batch jobs. Cross-reference on batch job items is now available for SQL statements defined in the following locations::

- External sets
- SQL Statements within a SQL Task
- Import SQL Statements within an Import Task
- Decisions

These categories of SQL statements now have associated 'Correct' flags. There are now also constraints that flag SQL statements as being incorrect when a table- or column name changes, or when an external set (element) name is modified.

### Off the Shelf Components

Off the Shelf Components have been made more accessible. You can now open the Off the Shelf Component from the catalog on the right mouse menu of the J2EE Components tree node.

The 'Select for import' checkbox in the Off the Shelf Components window has been replaced by an Import button. This button imports the Off the shelf component.

### Flat Files

The number of flat files has been reduced by transferring all information that was stored in the .EXT files to the .ESI files. Flat files are now:

- <application>.JOB containing batch job definitions
- <application>.CON containing conceptual definitions
- <application>.ESI containing external/design definitions

.JOB and .CON files are generated from within the Definer. .ESI files are generated from the Windows Designer. The size of the generated .CON and .ESI files has been considerable reduced (by a factor of between 4 and 10), and the time taken to generate a .CON file has improved dramatically (10 times faster).

In addition, there are two message files that can be generated from the Definer for translated messages

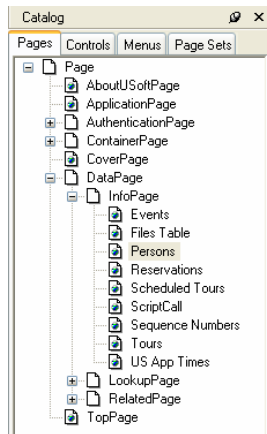
- <application>.SMG for system messages
- <application>.UMG for application strings

## New Catalogs

Catalogs for the Definer, Windows Designer and Web Designer have been adapted to take account of changes in the applications:

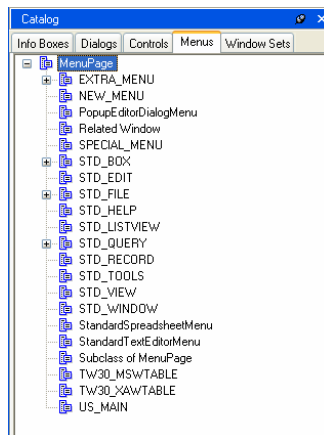
### *Web Designer:*

A Menu tab page has been added to the catalog, to reflect the move of the menu definition functionality from the Definer.



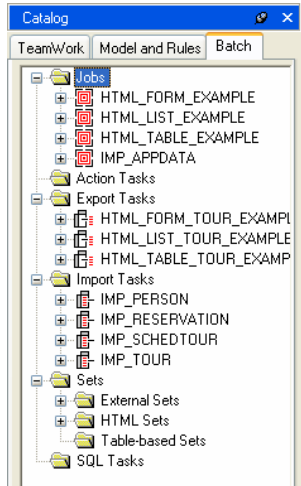
### *Windows Designer:*

A Menu tab page has been added to the catalog, to reflect the move of the menu definition functionality from the Definer. A Window Sets tab page has also been added. This is equivalent to the Page Sets tab page in the Web Designer.



**Definer:**

A Batch Tab has been added due to the integration of the USoft Batch Definer and the USoft Definer.

**Menu Definition in Windows/Web Designer**

All menu definition tasks have been moved from the Definer to the Windows Designer and Web Designer. All menu definitions are now stored in the .ESI flat file. This move simplifies the workflow required when defining menus, and it is now possible to make subclasses of defined menus.

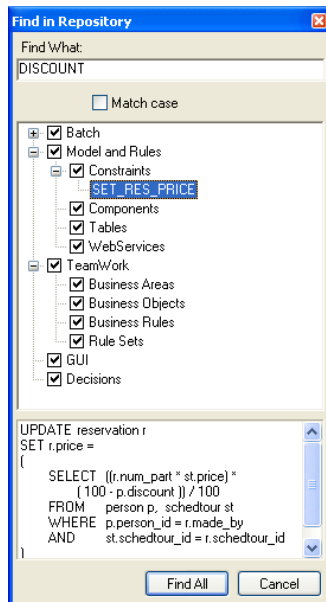
The Windows Designer and Web Designer now have a "Menus" tab page in their catalogs. This provides easy access to defined menus and the functions for defining menus.

**New Search/Find Option in Tree Views**

A new feature has been added to search/find items in tree views. Both Windows Designer and Web Designer are now using this feature which is accessible from the right-mouse menu in the catalog trees. This makes it possible to search through all classes listed in the catalog. The feature is generic and can also be used in client applications by using the "Search" action under the Tree View.

## Find in Repository Dialog

The Find in Repository dialog, accessible from the Definer's Edit menu, allows you to search for an occurrence of a string in several areas of the repository.



The Find in Repository dialog contains:

- An area where you can type the string that is to be searched for.
- A tree view that allows you to restrict the search areas, and also shows the results of the search.
- A text area that shows the text in the selected item in the tree view containing the searched string.

If you double-click an item in the tree view, the definition window for that item will be displayed. In the example illustrated above, double-clicking on SET\_RES\_PRICE opens the constraint definition window for the SET\_RES\_PRICE constraint.

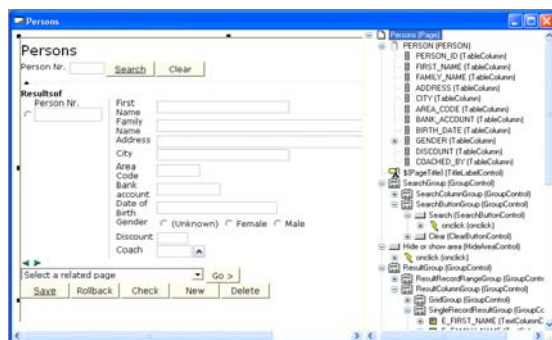
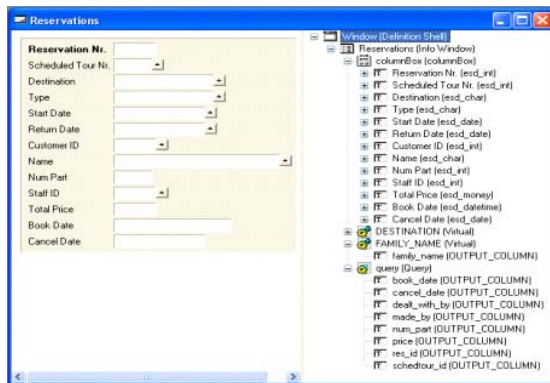
You can restrict the search area by clearing checkboxes in the tree view before pressing the Find All button. Only those areas for which the checkbox is checked will be searched. You can toggle the setting of the checkboxes using the mouse, or the keyboard space bar.

## Context Sensitive Editor

An improved popup text editor is now available. This new editor includes a keyword highlighting feature, for SQL, for example, search/replace options and a customizable font.

## Design Views and Object Trees

The new Windows Designer and Web Designer Design Views are the primary development environment for your end user GUI interface. They are a close approximation of what the end user will see and use. The left pane of the design window shows a view of the class that is being worked on, and the right pane contains the Object tree. The Object tree enables you to quickly navigate to parent or child objects of the currently selected object. It provides access to visible objects such as controls, and to non-visible objects such as queries.



In a design view, you can for example:

- Select any visible control contained in the window simply by pointing at it.
- Navigate to parent controls by clicking whilst holding the ALT key down, or by using the Object tree.
- Switch the Selection Filter to Off to select low-level child controls.
- Open the Property Inspector for a selected object, and set or change object properties.
- Add or remove classes in the window.
- Use options from the menus.

## Property Inspector

The Property Inspectors in the Windows Designer and Web Designer have been redesigned to simplify access to properties and their associated values.



## Docking Windows

Several windows within the Windows Designer and Web Designer are dockable. This includes the Catalog and Property Inspector, for example. Dockable windows can be:

- Docked to the sides of the main Windows or Web Designer window.
- Floated above your application in separate windows.
- You can also drag the window outside the perimeter of the main window, where it can be positioned on the Desktop, for example.
- Hidden, or collapsed in the left margin of the main window, only appearing when the user moves their cursor over the window's caption in the left margin.
- Docked to other Docked Windows.

### **Repository Manager**

The new USoft Repository Manager provides an XML-oriented approach to USoft Version Management that enables you to:

- Export a USoft Definer, Authorizer, or BenchMark repository, or parts of it, in XML format.
- Compare two XML repositories.
- Analyze this comparison and the differences between two repositories.
- Use this comparison to synchronize two repositories with each other by “importing” differences partially or completely.
- Define releases and patches.
- Export and compare parts of an application.
- Define task sets that let you perform these tasks outside USoft Repository Manager, on an ad-hoc or scheduled (daily) basis using external scheduler software.

### **dotNet (.NET) Components**

With the popular dotNet(.NET) common language runtime (CLR), a vast amount of functionality has become available in the form of components in standard libraries and code stub on the internet in an easy to use form. This functionality is available in USoft using the RDML interface for Microsoft's .NET, and the C Sharp programming language.