Migrating 3.0 Applications to 3.1

Every effort has been made to provide backward compatibility from General Interface 3.1 to 3.0. In most cases, applications will work with no changes.

This document contains a complete list of tasks necessary for migrating a 3.0 application to 3.1. This document does not describe features and APIs from 3.0 that have been deprecated in 3.1 but are nonetheless completely backward compatible.

Version 1.0. 04-27-2006

http://www.tibco.com
Global Headquarters
3303 Hillview Avenue
Palo Alto, CA 94304
Tel: +1 650-846-1000
Toll Free: 1 800-420-8450
Fax: +1 650-846-1005

Table of Contents

Data Mapping Rules Files

The format of data mapping rules files in 3.1 has changed from 3.0. Files from 3.0 will not run in 3.1. General Interface Builder includes logic for converting 3.0 rules files to 3.1. Simply open each rules file in 3.1 Builder and save it. The rules file will be updated automatically to the 3.1 format.

Class Hierarchy Related to jsx3.gui.BlockX

In General Interface 3.0, `jsx3.List`, `jsx3.Select`, `jsx3.Tree`, and `jsx3.chart.ChartComponent` all extended `jsx3.BlockX`. These classes inherited the methods for storing XML and XSL data in the application cache from `BlockX`.

In 3.1, these methods have been moved to the mixin interface `jsx3.xml.Cacheable`. Therefore, jsx3.gui.List, `jsx3.gui.Select`, `jsx3.gui.Tree`, and `jsx3.chart.ChartComponent` no longer extend `jsx3.gui.BlockX`. Any code that relied on this specific 3.0 class hierarchy will not work in 3.1. For example, the following code will **not** work:

```
function alertIfCDF(objControl) {
    if (objControl.isInstanceOf("jsx3.gui.BlockX"))
        objControl.getServer().alert("Alert", objControl.getName() + " is a CDF control");
}
```

It should be changed to:

```
function alertIfCDF(objControl) {
    if (objControl.instanceOf(jsx3.xml.CDF))
        objControl.getServer().alert("Alert", objControl.getName() + " is a CDF control");
}
```

The method `isSubclassOf()` is similarly affected.

Checking for Equality against Model.getInstanceOf()

The method `jsx3.Model.getInstanceOf()` has been deprecated in 3.1. (The Class Inheritance and Introspection document describes the preferred methods for determining whether an object is an instance of a class or interface.) Because of the package reorganization in 3.1, `getInstanceOf()` does not always return the same value as in 3.1. For example, if `getInstanceOf()` returned `jsx3.Block` in 3.0, it will return `jsx3.gui.Block` in 3.1. Therefore any code that checks the return value of `getInstanceOf()` for equality will likely break in 3.1. For example:

```
// this will break in 3.1
if (objBlock.getInstanceOf() == "jsx3.Block")
    objBlock.getServer().alert("Alert", "It's a block!");
```

Model.findDescendants() Results Order

General Interface 3.0 included a bug in the method `jsx3.Model.findDescendants()` that caused the results to be returned in reverse order. This was in violation of the method contract that said the results would be returned in either depth-first or breadth-first order. This bug is fixed in 3.1. However, any code that relied on the order of the results from this method may break in 3.1.

The following methods that call `findDescendants()` are also affected: `Model.getDescendantOfName()`, `Model.getFirstChildOfType()`, and `Model.getDescendantsOfType()`. The method `isSubclassOf()` would be similarly affected.

List.selectRecord() and Model Events

The contract of the method `jsx3.gui.List.selectRecord()` has changed. In 3.0 it caused the SELECT model event to fire. In 3.1 it never causes the model event to fire. This change is related to the new 3.1 model event protocol detailed in Model Events document. For an application running under the 3.0 model event protocol to continue to function properly in 3.1, `List.selectRecord()` should be replaced with `List.doSelect()`.Otherwise, the application must be upgraded to the 3.1 model event protocol. All other methods affected by the new event protocol are backwards compatible in 3.1.