

TIBCO General Interface™

Migration

*Professional Edition
Software Release 3.4
April 2007*

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN TIBCO GENERAL INTERFACE INSTALLATION). USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIB, TIBCO, TIBCO Adapter, Predictive Business, Information Bus, The Power of Now, TIBCO General Interface, TIBCO General Interface Framework, and TIBCO General Interface Builder are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README.TXT FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 2001-2007 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

Preface	v
Related Documentation	vi
TIBCO General Interface Documentation	vi
Further Information	vi
Typographical Conventions	viii
How to Contact TIBCO Customer Support	x
Chapter 1 Introduction	1
Chapter 2 Migrating Projects from 3.1.x to 3.4	3
Installation and Set Up	4
Required Steps	5
Optional Steps	6
Required Steps for Firefox	7
Class Loading	8
jsxlt Parameter	8
jsx3.require() Method	8
Project Settings	10
Paths	10
Auto Load Options	11
Internet Explorer Parameters	12
Custom Add-ins	12
Data Mapping	14
Updating Rules Files	14
Modifying the loadResource() Method Call	14
setOutboundStubURL() and setInboundURL() Methods	15
XSL Changes	16
Browsers and Layouts	17
Relative Paths	18
Migrating Projects for Firefox	19
GUI Components	19
Character Encoding	19
XPath and XSLT Requirements	20

Chapter 3 Migrating Projects from 3.2 or 3.3 to 3.4	21
Installation and Set Up	22
Migration Steps	23
Class Loading	23
Changes in Behavior in 3.4	24
Index	27

Preface

TIBCO General Interface consists of two components: TIBCO General Interface™ Builder and TIBCO General Interface™ Framework.

TIBCO General Interface Builder is a development environment for building rich Internet applications. The object modeling features of General Interface™ Builder enable developers to quickly create reusable GUI components and assemble them into full applications or application modules. Applications can be accessed in a web browser from a URL, existing HTML page, or portal environment.

General Interface™ Framework is the distributable runtime framework for running browser-based General Interface™ applications.

Topics

- *Related Documentation, page vi*
- *Typographical Conventions, page viii*
- *How to Contact TIBCO Customer Support, page x*

Related Documentation

This section lists documentation resources you may find useful.

TIBCO General Interface Documentation

The following documents form the TIBCO General Interface documentation set:

- *TIBCO General Interface Installation* Read this manual for General Interface installation instructions.
- *TIBCO General Interface Getting Started* Read this manual for overview information and a tutorial that explains how to build a TIBCO General Interface application.
- *TIBCO General Interface Developer Guide* Refer to this manual for in-depth information about developing and deploying TIBCO General Interface applications.
- *TIBCO General Interface Component Guide* Refer to this manual to learn more about components and the Charting add-in, as well as component properties and events.
- *TIBCO General Interface API Reference* Refer to this online help content for descriptions of the TIBCO General Interface JavaScript API functions and methods.
- *TIBCO General Interface Migration* Read this manual for instructions on migrating applications from previous General Interface releases.
- *TIBCO General Interface Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

To open the documentation in the product, choose **Help > Help Contents**. Documentation is also available online at <http://developer.tibco.com>.

Further Information

For further information on TIBCO General Interface, visit TIBCO Developer Network at <http://developer.tibco.com>, a comprehensive community that provides opportunities to increase your understanding of General Interface. Among the many resources at this web site you will find:

- Video tutorials
- Sample projects

- Product documentation
- Best practices documents
- Forums

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions




Convention	Use
code font	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use <code>MyCommand</code> to start the foo process.
bold code font	Bold code font is used in the following ways: <ul style="list-style-type: none"> • In procedures, to indicate what a user types. For example: Type admin. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, <code>MyCommand</code> is enabled: <code>MyCommand [enable disable]</code>
<i>italic font</i>	Italic font is used in the following ways: <ul style="list-style-type: none"> • To indicate a document title. For example: See <i>TIBCO BusinessWorks Concepts</i>. • To introduce new terms. For example: A portal page may contain several <i>portlets</i>. Portlets are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: <code>MyCommand <i>pathname</i></code>
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: <code>Ctrl+C</code> . Key names separated by a comma and space indicate keys pressed one after the other. For example: <code>Esc, Ctrl+Q</code> .
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 *Syntax Typographical Conventions*

Convention	Use
[]	<p>An optional item in a command or code syntax.</p> <p>For example:</p> <pre>MyCommand [optional_parameter] required_parameter</pre>
	<p>A logical 'OR' that separates multiple items of which only one may be chosen.</p> <p>For example, you can select only one of the following parameters:</p> <pre>MyCommand para1 param2 param3</pre>
{ }	<p>A logical group of items in a command. Other syntax notations may appear within each logical group.</p> <p>For example, the following command requires two parameters, which can be either the pair <code>param1</code> and <code>param2</code>, or the pair <code>param3</code> and <code>param4</code>.</p> <pre>MyCommand {param1 param2} {param3 param4}</pre> <p>In the next example, the command requires two parameters. The first parameter can be either <code>param1</code> or <code>param2</code> and the second can be either <code>param3</code> or <code>param4</code>:</p> <pre>MyCommand {param1 param2} {param3 param4}</pre> <p>In the next example, the command can accept either two or three parameters. The first parameter must be <code>param1</code>. You can optionally include <code>param2</code> as the second parameter. And the last parameter is either <code>param3</code> or <code>param4</code>.</p> <pre>MyCommand param1 [param2] {param3 param4}</pre>

How to Contact TIBCO Customer Support

The Professional Edition of TIBCO General Interface is unsupported. If you would like support for this product, you may upgrade to the Enterprise Edition and purchase a Support Contract.

For self-service support, education, and access to the TIBCO Developer Network, visit:

<http://developer.tibco.com>

For an overview of TIBCO Support Services and information about getting started with TIBCO Product Support, visit:

<http://www.tibco.com/services/support>

Chapter 1 Introduction

This manual explains how to migrate projects from previous TIBCO General Interface releases to the current release.

The migrations paths are as follows:

- TIBCO General Interface 3.1.x to 3.4

The migration path is a direct migration from 3.1.x > 3.2 or 3.3 > 3.4. See Migrating Projects from 3.1.x to 3.4 on page 3.

- TIBCO General Interface 3.2 or 3.3 to 3.4

The migration path is 3.2 or 3.3 > 3.4. See Migrating Projects from 3.2 or 3.3 to 3.4 on page 21.

For more information on General Interface, see *TIBCO General Interface Release Notes*.

Migrating Projects from 3.1.x to 3.4

This chapter explains how to migrate your 3.1.x projects to TIBCO General Interface 3.4. The migration path is 3.1.x > 3.2 or 3.3 > 3.4. No migration is needed from 3.2 to 3.3.

The migration steps fall into two categories: required and optional. If your application will be deployed on Firefox, also complete the steps in Migrating Projects for Firefox on page 19.

Topics

- *Installation and Set Up, page 4*
- *Required Steps, page 5*
- *Optional Steps, page 6*
- *Required Steps for Firefox, page 7*
- *Class Loading, page 8*
- *Project Settings, page 10*
- *Data Mapping, page 14*
- *XSL Changes, page 16*
- *Browsers and Layouts, page 17*
- *Relative Paths, page 18*
- *Migrating Projects for Firefox, page 19*

Installation and Set Up

Before you can migrate your 3.1.x projects to 3.4, you need to migrate to 3.3. Complete these steps to migrate your 3.1.x projects to 3.3:

1. Install TIBCO General Interface 3.3. For more information about installing TIBCO General Interface, see Installation on page 1 in *TIBCO General Interface Installation*.
2. Double-click **GI_Builder.html** or **GI_Builder.xhtml** in the installation directory to start TIBCO General Interface Builder in Firefox or Internet Explorer.
3. Choose or create a *workspace* directory after General Interface Builder initializes. The *workspace* is the directory that contains your projects, custom add-ins, custom prototypes, and your user settings for General Interface Builder. Separation of the General Interface install directory from the workspace allows for easier application deployment. For more on workspaces, see Choosing a Workspace on page 18 in *TIBCO General Interface Developer Guide*.
4. Make backup copies of all projects you are migrating.
5. Copy projects that you would like to migrate to 3.3 from your previous JSXAPPS folder into your new *workspace/JSXAPPS* folder.
6. Copy any custom user prototypes from your *user/prototypes* directory to the *workspace/prototypes* directory. Components saved to this folder display in the User folder of the Component Libraries palette.
7. Copy any custom add-ins to the *JSX/addins* or *workspace/addins* directory. Typically, add-ins to be used by a team of developers would be saved to the *JSX/addins* directory and posted by an administrator to a location accessible to the team. Add-ins for individual use can be saved to the *workspace/addins* directory.

Required Steps

The following steps are required for updating your 3.1.x project to 3.3:

1. Open the 3.1.x project and project files in General Interface Builder 3.3 and resave all project files.



If you're using Firefox and have files encoded in UTF-16, open the 3.1.x project and project files in Internet Explorer first and resave the files before opening in Firefox. See Character Encoding on page 19.

2. Specify class loading options to improve application performance with the new dynamic class loading features in General Interface Builder 3.3. See Class Loading on page 8.
3. Modify records that specify auto loading in the application configuration file. New auto load options have been added. See Auto Load Options on page 11.
4. If your projects use add-ins, update the add-ins as follows:
 - **Charting and custom add-ins** If your project uses Charting or custom add-ins, enable the add-in on the Add-ins page of the Project Settings dialog (Project > Project Settings).
 - **Project settings** Application configuration files must be modified for custom add-ins to work in General Interface 3.3. See Project Settings on page 10.
 - **Mapping add-in** If your project includes data mapping, see Data Mapping on page 14.
5. If your projects have rules files, update all rules files. See Data Mapping on page 14.
6. If your projects use any custom XSL, you must update the XSL. See XSL Changes on page 16.



If any project files in your migrated project are displayed in red in the Project Files palette, you need to update the references to those files. Right-click the file in the Project Files palette and choose **Edit Profile**. Modify the path in the URI field and click **Save**.

7. Migrate your updated 3.3 projects to 3.4. See Migrating Projects from 3.2 or 3.3 to 3.4 on page 21.

Optional Steps

The following steps are optional:

1. Manually merge your customized `logger.xml` files with the new logging system configuration file (`GI_HOME/logger.xml`).

In General Interface 3.2 and later releases, the `logger.xml` file has new attributes and functionality, such as sound for the logging system and class loading options. The logging file has also been moved from the `GI_HOME/JSX` directory to `GI_HOME` and must be deployed accordingly. For more information, see *Logging and Application Monitors* on page 175 and *Posting Application Files* on page 215 in *TIBCO General Interface Developer Guide*.

2. To take advantage of MSXML 4 or later, if installed, remove the Internet Explorer parameters from the application configuration file. See *Internet Explorer Parameters* on page 12.
3. For consistent layout behavior in both Firefox and Internet Explorer using a Block, follow the recommendations specified in *Browsers and Layouts* on page 17.
4. Change paths to relative URLs for more flexible development and deployment. See *Relative Paths* on page 18.
5. Replace all deprecated List and Grid components with Matrix components. Note that List and Grid are not supported in Firefox.

Required Steps for Firefox

The following additional steps are required for updating your 3.1.x project to 3.3 for Firefox deployment:

1. Open any files that are saved in UTF-16 encoding in Internet Explorer first and resave the files before opening in Firefox. See Character Encoding on page 19.
2. Complete the required steps described in Required Steps on page 5.
3. Replace all deprecated List and Grid components with Matrix components. List and Grid are not supported in Firefox.
4. If you're using XSL files, verify that they meet certain requirements to work correctly in Firefox. See XPath and XSLT Requirements on page 20.

There are also additional optional steps you can complete. See Optional Steps on page 6.

Class Loading

General Interface 3.3 supports *dynamic class loading* for more efficient performance. Dynamic class loading, also known as lazy loading, means that classes are loaded as they're needed at the last possible moment.

jsxlt Parameter



The `jsxlt` parameter is no longer supported in 3.4. If you're migrating to 3.4, skip this topic.

The `jsxlt` deployment parameter is a runtime configuration parameter that determines how classes are loaded. The `jsxlt` deployment parameter is located in the `script` element on the web page that launches the deployed application. For example,

```
<script type="text/javascript" src="JSX/js/JSX30.js"
  jsxapppath=" ../workspace/JSXAPPS/PROJECT_DIR/"
  jsxmanualhome="true"
  jsxlt="true"
>
</script>
```

When the `jsxlt` deployment parameter is set to `true`, the default setting, all required classes are loaded as the system initializes. Optional classes are loaded by the component file and the `jsx3.require()` method.

If you don't want to use dynamic loading for your 3.1.x classes, set the `jsxlt` parameter to `false` (`jsxlt="false"`) or remove it. However, if you want to take advantage of dynamic class loading, you need to add the `jsxlt` parameter to your launch page or create a new launch page with the General Interface 3.3 Deployment Utility (Project > Deployment Utility).

jsx3.require() Method

The `jsx3.require()` method can be used to load classes explicitly. Use the fully qualified class name when using the `jsx3.require()` method. For example,

```
jsx3.require("jsx3.net.Form");
```

Only classes that can be found by the system class loader are loaded. Custom classes can be added on the Classpath page of the Project Settings dialog (formerly Deployment Options). To open the Project Settings dialog, choose **Project > Project Settings**.

When a component file is deserialized, the class of each object encountered in the file is dynamically loaded if it's not already loaded. Therefore, it's often not necessary to use the `jsx3.require()` method with classes that descend from `jsx3.app.Model`. However, if JavaScript code references these classes and if the code executes before a deserialization automatically loads the class, you must use the `jsx3.require()` method to explicitly load the class.

The `jsx3.require()` method must be called at least once before making these types of references:

- A static reference to a class descending from `jsx3.gui.Model` (typically `jsx3.gui.**`).
- Any references to subclasses of `Model` that execute before the class is loaded through object deserialization.



The General Interface Builder debugger classes are dynamically loaded. To use the JavaScript Step Through Debugger in General Interface Builder, you must use the `jsx3.require()` method before any `jsx3.ide.debug()` statements to load debugger classes. For example,

```
jsx3.require("jsx3.ide.Debugger");
```

Project Settings

When you create a new project in TIBCO General Interface Builder, a default application configuration file is automatically created as part of the project in the project directory: *workspace/JSXAPPS/project_dir/config.xml*. The application configuration file contains application configuration data, such as project settings, application deployment, and file locations.

You can modify the project settings in the Project Settings dialog (Project > Project Settings) or in the application configuration file. Any changes you make in the Project Settings dialog are saved to the configuration file. Some changes can only be made in the configuration file.

Paths

Application resources can now be specified using relative paths, which allows for easier portability of code from one project to another and simplifies the relocation of applications in the JSXAPPS folder hierarchy.

When you open a 3.1.x project in General Interface Builder 3.3, paths in the *config.xml* file are updated automatically to relative paths after the upgrade prompt.

However, you must modify the path for the initial component. You can do this on the Deployment page of the Project Settings dialog (Project > Project Settings) in the IDE or in the application configuration file (*workspace/JSXAPPS/PROJECT_DIR/config.xml*). Simply remove *JSXAPPS/PROJECT_DIR/* from the path as shown in Example 2.

Example 1 config.xml for 3.1.x

```
<record jsxid="objectseturl"  
  type="string">JSXAPPS/PROJECT_DIR/components/appCanvas.xml  
</record>
```

Example 2 Revised config.xml for 3.3

```
<record jsxid="objectseturl"  
  type="string">components/appCanvas.xml  
</record>
```

Auto Load Options

New file auto load options have been introduced in General Interface 3.2. In prior General Interface releases, auto loading could be set to true or false. In General Interface 3.2 and above, there are four auto load options: Manually as needed, At init, At full init, and At light init. Available options vary by file type. For more information about auto load options, see File Profile Dialog on page 265 in *TIBCO General Interface Developer Guide*.

Before using the new options, you need to modify the `onLoad` `jsxid` for some files in the application configuration file as follows:

1. Open the application configuration file, which is located in `workspace/JSXAPPS/PROJECT_DIR/config.xml`.
2. Find `jsxid="includes"` and notice that there are multiple records of type `map`. Each type `map` record has a record with `jsxid="onLoad"`.
3. Modify each child record of type `map` with a `jsxid` of `onLoad` as follows:
 - a. Change the `jsxid` `onLoad` value to **load**.
 - b. Change the type from `boolean` to **number**.
 - c. Change the record value to the new desired value, such as **0**, **1**, **2**, or **3**. For values, see Table 1: on page 11.

For example, to set `logic.js` to load automatically when the application initializes, change the record from this:

```
<record jsxid="onLoad" type="boolean">true</record>
```

To this:

```
<record jsxid="load" type="number">1</record>
```

Table 1: Auto Load Values

Auto Load Option	3.2 or 3.3	Prior to 3.2
Manually as needed	0	false
At init	1	true
At full init	2	Not available
At light init	3	Not available



The Auto Load option is disabled for GUI component files, such as `appCanvas.xml`. You can specify a GUI component file to automatically load when the application initializes in the Initial Component field on the Deployment page of the Project Settings dialog

4. Save the configuration file and reload the project.

Once you've modified the application configuration file, you can also set auto load options in General Interface Builder. Right-click a file in the Project Files palette and choose Edit Profile. Select an option from the Auto Load drop-down list in the Edit Profile dialog and click Save.

Internet Explorer Parameters

To take advantage of MSXML 4 or later, if installed, remove the following Internet Explorer parameters from the application configuration file.

1. Open the application configuration file:
`workspace/JSXAPPS/PROJECT_DIR/config.xml`.
2. Remove the following Internet Explorer-specific parameters:

```
<record jsxid="xmlregkey"
  type="string">Msxml2.FreeThreadedDOMDocument.3.0</record>
<record jsxid="xslregkey"
  type="string">Msxml2.XSLTemplate.3.0</record>
<record jsxid="httpregkey" type="string">Msxml2.XMLHTTP</record>
```

However, you can pass these parameters at runtime using the General Interface runtime parameters. See *Configuring the Runtime* in *TIBCO General Interface Developer Guide*.

Custom Add-ins

If you've created a custom add-in for General Interface 3.1.x, you need to edit the project `config.xml` file of the add-in as follows:

1. Open the project `config.xml` file of the add-in located at
`workspace/JSXAPPS/PROJECT_DIR/config.xml`.
2. Add this new record to the configuration file:
`<record jsxid="jsxversion" type="string">3.3</record>`
3. Save the file.

You might also want to update your add-ins to use the new General Interface features, such as class loading and relative paths. See [Class Loading](#) on page 8 and [Relative Paths](#) on page 18.

Data Mapping

This section explains the steps for migrating General Interface 3.1.x projects that use data mapping to General Interface 3.2 and 3.3.

For data mapping, there are two steps to updating your project:

1. Update rules files.
2. Modify the `loadResource()` method call in the JavaScript code.

Updating Rules Files

The format of data mapping rules files has changed in General Interface releases after 3.1.x. Mapping rules files from 3.1.x will not run in 3.3. General Interface Builder includes logic for converting 3.1.x rules files to 3.3. Simply open each 3.1.x rules file in the XML Mapping Utility (formerly SOAP Mapping Utility) in General Interface Builder 3.3 and save it. The 3.1.x rules file is automatically updated to the 3.3 format. To open the XML Mapping Utility, choose **Tools > Communication > XML Mapping Utility**.

Modifying the `loadResource()` Method Call

The JavaScript code generated by the XML Mapping Utility has changed due to a signature change in the `loadResource()` method (`jsx3.app.Server`). The rules file ID is now passed as a parameter instead of the URL. Update all legacy code generated by the XML Mapping Utility as shown in Example 4.

Example 3 Code for 3.1.x

```
var objService = new jsx3.net.Service(Rules_File_URL, Operation_Name);  
objService.setNamespace(namespace);
```

Example 4 New Code for 3.3

```
var objService = Server_Name.loadResource(Project_Resource_File_Id);  
objService.setOperation(Operation_Name);
```

setOutboundStubURL() and setInboundURL() Methods

Note that these URLs are now resolved relative to the context server. For example, if the project directory for the context server is `test`, then the following inputs are valid and equivalent:

- `jsxapp://test/xml/typical.xml`
- `xml/typical.xml`
- `JSXAPPS/test/xml/typical.xml`

If the `test` project directory is nested in a subdirectory, such as `JSXAPPS/samples/test`, then the following inputs are valid and equivalent:

- `jsxapp://samples/test/xml/typical.xml`
- `samples/test/xml/typical.xml`
- `JSXAPPS/samples/test/xml/typical.xml`

XSL Changes

The General Interface XSL templates for the classes that implement the `jsx3.xml.Cacheable` interface (Select, Menu, List, Grid, Matrix, and Tree) have changed in 3.2 and later releases and are not backwards compatible. Existing custom templates must be recreated starting from the default 3.3 XSL templates, which are located in the `GI_HOME/JSX/xsl` directory. Because custom templates will not be supported in the future, this functionality is deprecated.

Introduced in General Interface 3.2, XML transformers are the preferred replacement for custom XSL templates. XML transformers are used to transform the source XML of a GUI control implementing the `jsx3.xml.Cacheable` interface before the XML is stored in the XML cache. For example, a transformer could transform non-CDF source XML into CDF-compliant XML or affect the visual style of the control by constructing a `@jsxstyle` CDF attribute from other information in the source document. For more information, see `jsx3.net.Cacheable` in *TIBCO General Interface API Reference* and the inline IDE documentation for the XML Transformers property in the Properties Editor palette.

For Firefox, XSL must meet certain requirements. See XPath and XSLT Requirements on page 20.

Browsers and Layouts

To avoid unexpected layout behavior when using relative positioning, such as misaligned GUI components, it's recommended to use Block as a container **only** if it meets at least **one** of these requirements:

- The Block is owned by a layout manager, such as LayoutGrid, Tab, Stack, and Splitter.
- The Block is relatively positioned and has a width of 100%.
- The Block is absolutely positioned.

Relative Paths

Now that General Interface Builder supports relative paths, you can update your project files to use relative paths. Although this isn't required, specifying relative paths to application resources increases the portability of code from one project to another and prevents problems when renaming projects.

Modify paths in the following files:

- Modify the path for the initial component in the Project Settings dialog (Project > Project Settings) or the application configuration file. See Project Settings on page 10.
- Modify paths in JavaScript code to use relative URLs. To update your code, remove this portion of the path: `JSXAPPS/PROJECT_DIR/`.
- Modify paths in the component serialization files or in the Properties Editor palette to use relative URLs.

For more information, see URI Resolution on page 53 in *TIBCO General Interface Developer Guide*.

Migrating Projects for Firefox

This section discusses how to migrate projects to be deployed on Firefox from 3.1.x to 3.3.

GUI Components

Replace all deprecated List and Grid components with Matrix components. List and Grid are not supported in Firefox.

Character Encoding

Projects that were localized prior to General Interface 3.2 may have been saved in UTF-16 encoding. If you open these files in General Interface 3.3, they might contain junk characters. Although General Interface 3.3 might read these Unicode files, it's best to resave the project in General Interface Builder 3.3 on Internet Explorer before you proceed with the project migration. **Be sure to make a backup of your project before beginning.**

For applications loaded from the local disk, such as General Interface Builder, Firefox reads only non-XML files that are encoded in a standard 8-bit encoding. Firefox can read local XML files in any encoding supported by the host system only if the encoding is included in the XML declaration.

To re-encode any non-XML files from UTF-16 to 8-bit ASCII, you can use General Interface Builder 3.3 running in Internet Explorer or a text editor.

To re-encode non-XML files from UTF-16 to 8-bit ASCII, complete these steps:

1. Open General Interface Builder in Internet Explorer.
2. Choose **Tools > IDE Settings** to open the IDE Settings dialog.
3. Make sure the **Output character encoding** field is blank.
4. Open and re-save each file.

You will not be able to include any non-ASCII characters in these plain text files. For the best compatibility with Firefox, all extended ASCII and 16-bit characters should be externalized in XML files that declare their character encoding in the XML declaration.

XML files do not need to be re-encoded as described above, although they can be. However, if an XML file is encoded in UTF-16 or any other non-ASCII character encoding, the encoding must be added to the XML declaration. Add or modify the first line of the XML file with the following XML declaration:

```
<?xml encoding="UTF-16"?>
```

Note that component serialization files are also XML files. If they have been encoded in UTF-16, they must also be modified as described above.

XPath and XSLT Requirements

XSL must meet these requirements to work properly in Firefox:

- The XSL must include the following namespace:

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

- The XSL must output valid XML. Balanced tags are required. For example,

```
<xMessage>Hello World!</xMessage>
```

- The only output formats supported for XSLT processing are HTML 4.0 and XML.
- XSLT implementation does not support the namespace axis, limiting the ability to query and discover namespaces. The DOM-based interface also fails to implement this axis.
- XSLT implementation does not support the `node-set()` method, which means that complex parameters and result tree fragments cannot be resolved.
- XSLT implementation does not allow output escaping to be disabled, which means that escaped entities cannot be resolved during a transformation.

Chapter 3

Migrating Projects from 3.2 or 3.3 to 3.4

This chapter explains how to migrate your 3.2 and 3.3 projects to TIBCO General Interface 3.4. The migration path is 3.2 or 3.3 > 3.4. No migration is needed from 3.2 to 3.3.

Topics

- Installation and Set Up on page 22
- Migration Steps on page 23

Installation and Set Up

Complete these steps to migrate your 3.2 or 3.3 projects to 3.4:

1. Make backup copies of all projects you are migrating.
2. Install TIBCO General Interface 3.4. For more information about installing TIBCO General Interface, see Installation in *TIBCO General Interface Installation*.
3. Double-click **GI_Builder.html** or **GI_Builder.xhtml** in the installation directory to start TIBCO General Interface Builder in Firefox or Internet Explorer.
4. Choose or create a workspace directory after General Interface Builder initializes. For more information on workspaces, see Choosing a Workspace on page 18 in *TIBCO General Interface Developer Guide*.



Create a new workspace to get the updated 3.4 sample applications. Choosing an existing workspace doesn't replace previous sample applications with updated 3.4 sample applications. This built-in functionality is designed to prevent workspace files from getting overwritten.

5. If you chose an existing workspace, copy any custom add-ins to the 3.4 `JSX/addins` directory. Typically, add-ins to be used by a team of developers would be saved to the `JSX/addins` directory and posted by an administrator to a location accessible to the team.
6. If you created a new workspace, complete these steps:
 - a. Copy projects that you would like to migrate to 3.4 from your previous `workspace/JSXAPPS` folder into your new `workspace/JSXAPPS` folder.
 - b. Copy any custom user prototypes from your `workspace/prototypes` directory to the new `workspace/prototypes` directory. Components saved to this folder display in the `User` folder of the Component Libraries palette.
 - c. Copy any custom add-ins to the 3.4 `JSX/addins` or new `workspace/addins` directory. Typically, add-ins to be used by a team of developers would be saved to the `JSX/addins` directory and posted by an administrator to a location accessible to the team. Add-ins for individual use can be saved to the `workspace/addins` directory.

Migration Steps

The following steps are required for updating your 3.2 and 3.3 project to 3.4:

1. Update the project for changes to class loading.
2. Review Changes in Behavior in 3.4 on page 24 to see if your applications are affected by changes to General Interface 3.4.
3. **Optional** Rewrite any JavaScript code that uses deprecated APIs. See *Deprecated APIs in TIBCO General Interface Release Notes*.
4. **Optional** Replace any Block components that use an iframe with the new IFrame component.

Class Loading

There are two changes to class loading in 3.4:

- The At Lt Init and At Full Init Auto Load options are no longer supported. If your project uses these class loading options, see Auto Load Options on page 23.
- The `jsxlt` parameter is no longer supported. If your deployed projects use `jsxlt="false"`, see `jsxlt` Parameter on page 23. If your project uses `jsxlt="true"`, no changes are needed.

Auto Load Options

The Auto Load options for JavaScript files, At Lt Init and At Full Init, are no longer supported. If you've used these settings in your project, you need to reset the JavaScript file to Auto Load (At Init).

To change the Auto Load setting to At Init, complete these steps:

1. Right-click the JavaScript file in the Project Files palette.
2. Choose Auto Load from the context menu. The file name now displays in a bold font in the Project Files palette.

jsxlt Parameter

The `jsxlt` launch parameter is no longer available as of 3.4.0. All applications now load in the `jsxlt="true"` mode. If you deployed your project using `jsxlt="false"`, use `jsx3.require()` to explicitly load classes, so that the project is compatible with dynamic class loading. See `jsx3.require()` Method on page 8.

Changes in Behavior in 3.4

To see if any of these changes affect your applications, run your applications in General Interface 3.4 and check the following behaviors. Then modify your applications as needed.

- Menu
 - Menu positioning and cascading may be different. See GI-63 in *TIBCO General Interface Release Notes*.
 - Long menus now render with auto-scroll arrows instead of with scroll bars.
- TabbedPane
 - Provides an automatic right and left scrolling mechanism for navigating through tabs that are hidden when they exceed the viewable space.
- Tree
 - Support for range selection using Shift+click and multiple selection using Ctrl+click.
 - Support for drag-and-drop of multiple records.
 - Drag-and-drop insertion between nodes and insertion as last child of node.
 - Context menu can operate on multiple rows.
- Dialog
 - When deserialized a dialog box top-left position could be at a negative position if the min-width and min-height were larger than the client window and no top-left positions were defined. Left and top are now never less than zero. However, since the resize control is in the bottom right, call `constrainPosition(true)` if there is a chance that the dialog is larger than its parent.
- Form elements
 - All form elements now inherit font color from an ancestor block.
- `setValue()` and Matrix
 - If a Matrix is a single selection and the passed value is an array with more than one element, an illegal argument exception is now thrown.
- `jsx3.app.Model.getChild()` returns null instead of undefined. Unless specifically documented, application should be careful about checking against undefined or null value returned by General Interface functions. Developer should check against known value.

For example, instead of this:

```
if (objJSX.getChild('block') == null)
```

or

```
if ( typeof(objJSX.getChild('block')) == 'undefined' )
```

Do this instead:

```
if ( !(objJSX.getChild('block') instanceof jsx3.app.Model) )
```

or

```
if ( ! (objJSX.getChild('block') )
```

For more information on new features, changes in functionality, and known and closed issues, see *TIBCO General Interface Release Notes*.

Index

A

add-ins

- configuration files 10
- custom 12
- enabling Charting 5
- Mapping, 3.2 and 3.3 14
- Auto Load option 11, 23

B

- browsers and layouts 17

C

- changes in behavior in 3.4 24
- character encoding 19
- Charting add-in, enabling 5
- class loading
 - Auto Load options 23
 - jsxlt parameter 8, 23
- config.xml 10
- configuration files
 - custom add-ins 10
 - Internet Explorer parameters 10
 - paths 10
 - project configuration files 10
- customer support x

D

- data mapping
 - 3.2 and 3.3 14

- deployment parameters 8

E

- encoding 19

F

- Firefox
 - character encoding 19
 - migrating projects 7
 - XPath and XSLT requirements 20

I

- Internet Explorer parameters 12

J

- jsx3.require() method 8
- jsxlt parameter 8

L

- loadResource() method call 14

M

- Mapping add-in, 3.2 and 3.3 14
- mapping rules files, 3.2 and 3.3 14
- migrating 3.1 projects 3
- migrating 3.1.1 projects 3
- migrating 3.2 and 3.3 projects 21
- migrating projects to Firefox 7, 19

P

- parameters
 - class loading 8
 - deployment 8
 - Internet Explorer 12
 - runtime 12
- paths 10, 18
- projects
 - migrating to 3.3 3
 - migrating to 3.4 21
 - migrating to Firefox 7, 19

R

- relative paths 18
- rules files, updating 14
- runtime parameters 12

S

- setInboundURL() Methods 15
- setOutboundStubURL() 15
- support, contacting x

T

- technical support x

- typographical conventions viii

U

- updating rules files 14

X

- XPath and XSLT requirements 20
- XSL changes 16