CODE MAGUS

XML Interface Portal - Thistle User Manual
Version 1

CML00035-01

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
www.codemagus.com

IBM Business Partner

January 6, 2010

Contents

1 Introduction

1.1 Overview of the XIP

The XML Interface Portal (XIP) provides functionality for Thistle to manipulate XML documents.

The XIP handles the tasks of actually creating the XML, validating the XML, and converting XML documents into Thistle scripts and vice versa.

This allows he user to create XML documents using normal Thistle scripts, or convert real XML buffers into Thistle structures. From within Eresia, any of the other portals can be used in conjunction with the XML Portal to perform testing tasks.

Some examples of using other portals may be to use Thistle scripts in Eresia to

- Populate XML documents with data from a spreadsheet using the Excel Portal
- Write XML documents to buffers using a File Access Method
- Send XML over a network by using a Network Control Program (NCP)
- Receive the XML response from a network server and process this response

The XML Portal also allows the user to record Thistle scripts from an XIP workspace (from within the Graphical User Interface, or GUI, of the XIP). This method of generating XML affords the user a means of generating XML documents from within an environment where the XML documents are viewed as graphical representations of the XML, which is often easier to work with than a textual representation of XML, which in this case takes the form of a Thistle script.

This manual serves to describe all of the functions of the XML Portal, and provides real examples of using the XIP's Thistle interface.

1.2 Introduction to XML Documents

The structure and layout of XML is beyond the scope of this document, and is not required to be known by the user (this is the purpose of the XIP).

However, it is important that the user understands the fundamentals of XML in order to make effective use of the XIP. This chapter will make use of a real XML document and a corresponding Thistle representation of that document as a means of illustrating the relationship between them.

XML has a hierarchical structure. This means data is stored in a manner where each piece of that data is given a name, a value, and a place where that piece of data is stored relative to the other data that is being stored.

Since XML has a hierarchical (or tree) structure the Thistle representation of an XML document must also have a tree structure.

1.3 Example of an XML document

```
<?xml version="1.0" ?>
<CD>
   <title>Obvious Tunes</title>
   <serialNumber>987654321</serialNumber>

   <artistDetails>
      <artistID>JO_0001</artistID>
      <artistURL>www.obviousjohn.com</artistURL>
      <artistName>Johnny Obvious</artistName>
   </artistDetails>

   <track trackNum="1">The sun only shines at day</track>
   <track trackNum="2">Rain is like water</track>
   <track trackNum="3">West is left of East</track>

</CD>
```

1.3.1 XML Elements

Please refer to the example in section 1.3 on page 3.

Each of the sections of this XML document are given a name. These sections are known as XML elements, or simply elements. Examples of elements in the above example are `<CD>` and `<artistDetails>`.

1.3.2 XML Attributes

Please refer to the example in section 4.1.1 on page 6.

The XML specification allows an element to have sub elements of the same name. In the above example, note that the CD has three tracks on it. For this reason the `<track ...>` element is repeated three times. The track number is described by an attribute of the element called `trackNum`.

2 User Guide

2.1 Thistle representation of the XML example

The Thistle representation of the example in section 4.1.1 on page 6 will first be listed, and then explained.

```
document[0].CD[0].title.value := "Obvious Tunes";
document[0].CD[1].serialNumber.value := "987654321";
document[0].CD[2].artistDetails[0].artistID.value := "JO_0001";
document[0].CD[2].artistDetails[1].artistURL.value := "www.johnnyobviou
document[0].CD[2].artistDetails[2].artistName.value := "Johnny Obvious"
document[0].CD[3].track.attributes.trackNum := "1";
document[0].CD[3].track.value := "The sun only shines at day";
document[0].CD[4].track.attributes.trackNum := "2";
document[0].CD[4].track.value := "Rain is like water";
document[0].CD[5].track.attributes.trackNum := "3";
document[0].CD[5].track.value := "West is left of East";
```

The Thistle code above shows that there is an index associated with each element of the XML structure. This means that the user specifies the position of each element within its parent node. In XML it is often necessary to place sub-elements in the correct sequence within an element. In XML this is known as the element sequence.

The term `.value` is a reserved word which is used to set the value of an element. The term `.attributes` is also a reserved word, and is used to list the attributes of an element. The attributes do not have `.value` appended onto them.

The above Thistle code will simply create an object in Thistle which represents the XML document. Since this is only a Thistle object that is being created, the structure will not be validated by the XML portal at this stage. The process of converting the above Thistle object into an actual XML document will be explained later, in section 2.2.1 on page 5.

2.2 Using the XML Portal

This section will describe the process of converting Thistle code (and the resulting Thistle objects) into real XML documents, as well as the reverse process of converting XML documents into Thistle objects.

2.2.1 Creating XML documents from Thistle objects

The XIP method GetDocument is used to convert a Thistle object into an XML document and results in a string representing the actual XML document in a buffer.

The invocation of this method to perform the conversion is:

```
GetDocument(my_thistle_object);
```

A brief example of the usage of this method follows. Note that the Thistle object that is created here is a shortened version of the object described in section 2.1 on page 4.

```
xmlPortal := Portal.XML.Connect();
myDocObject.document[0].CD[0].title.value := "Obvious Tunes";

xml_buffer := xmlPortal.GetDocument(myDocObject);
```

The result will be a buffer (string) that contains the following XML document

```
<?xml version="1.0" ?>
<CD>
   <title>Obvious Tunes</title>
</CD>
```

2.2.2 Creating Thistle objects from XML documents

The process of converting an XML document into a Thistle object means taking a buffer which contains an XML document, and creating a Thistle object from that buffer.

The invocation of this method to perform the conversion is:

```
GetTree(my_xml_document_string);
```

A brief example of the usage of this method follows. Note that the XML document used here is a shortened form of the buffer listed in section 4.1.1 on page 6.

```
xmlPortal := Portal.XML.Connect();

xml_string := '<?xml version="1.0" ?><CD><title>Obvious Tunes</title></

xml_object := xmlPortal.GetTree(xml_string);
```

The result will be a Thistle object which represents the XML document provided. It can be viewed in the tree view in the Automated Test Environment and its values can be changed.

3 Reference

3.1 Thistle XIP methods

During the invocation of any of the Thistle XIP methods, if any errors are present, these errors are printed to the output pane in the Automated Test Environment.

3.1.1 GetDocument

The GetDocument method takes a Thistle structure and converts it to an XML document buffer. During conversion the XML document is validated, and if it is not successful, the `GetDocument` method will fail.

```
Usage:   GetDocument(thistle_structure);
Returns: An XML document buffer as a string if successful, or an empty
string if unsuccessful
```

3.1.2 GetTree

The `GetTree` method takes an XML Document and converts it to a Thistle tree structure. During conversion the XML document is validated, and if it is not successful, the `GetTree` method will fail.

```
Usage:  GetTree(XML_document_string);
Returns: A Thistle tree if successful, or nil if unsuccessful
```

4 Appendix

4.1 Examples

4.1.1 A full Thistle script

The following example is taken directly from recording a script in the Automated Test Environment using the XML Portal.

Note that although the Thistle code here uses `with` statements to define the object, the code is the same as the example in section 2.1 on page 4.

```
Package XMLUsecase;

{ preamble }
```

```
   created by 'Tester';
   description 'Sample Description';
   date 2008-01-08T14:24:42;
   target 'Eresia XML Portal';

   interface Portal.XML : CodeMagus.XML;


begin

   xml := Portal.XML.Connect();
   with myThistleXmlObject do begin
      with ["document"][0] do begin
         with CD[0] do begin
            title.value := "Obvious Tunes";
         end
         with CD[1] do begin
            serialNumber.value := "987654321";
         end
         with CD[2] do begin
            with artistDetails[0] do begin
               artistID.value := "JO_001";
            end
            with artistDetails[1] do begin
               artistURL.value := "www.obviousjohn.com";
            end
            with artistDetails[2] do begin
               artistName.value := "Johnny Obvious";
            end
         end
         with CD[3] do begin
            with track do begin
               with attributes do begin
                  trackNum := "1";
               end
            end
            track.value := "The sun only shines at day";
         end
         with CD[4] do begin
            with track do begin
               with attributes do begin
                  trackNum := "2";
               end
            end
            track.value := "Rain is like water";
         end
         with CD[5] do begin
            with track do begin
               with attributes do begin
                  trackNum := "3
               end
            end
            track.value := "West is left of East";
         end
      end
   end

   xml_buffer := xml.GetDocument(myThistleXmlObject);

end.
```