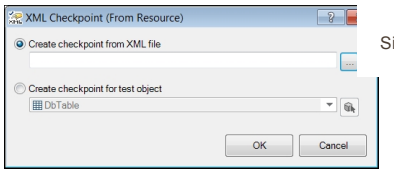
Using an XML checkpoint

Since the early 1990s XML files have been widely used for data trans
between application modules and even between different systems. A
XML file may contain, for instance, the results of a billing record for a
cellular phone customer. In such a case, it may be of high value to b
able to have preset expected results and be able to compare actual X
files with these.

How to do it...

Proceed with the following steps:

1. From the UFT menu navigate to **Design** | **Checkpoint** | **XML Checkpoint (From Resource)**, which will open the dialog by th
same title:

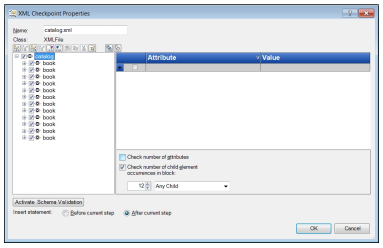


2. Click on the Browse button and select a file from the **Open XML File** dialog. In our example, we will be using a sample XML given by Microsoft with the following contents:

```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating applications with
   </book>
   <book id="bk102">
      <author>Ralls, Kim</author>
      <title>Midnight Rain</title>
      <genre>Fantasy</genre>
      <price>5.95</price>
      <publish_date>2000-12-16</publish_date>
      <description>A former architect battles corporate zombies, ar
   </book>
   <book id="bk103">
      <author>Lucas, George</author>
      <title>The Force Awakens</title>
      <genre>Sci-Fi</genre>
      <price>11.78</price>
      <publish_date>2015-12-18</publish_date>
      <description> Set approximately 30 years after the defeat of
   </book>
   <book id="bk104">
      <author>Wright, Jonathon</author>
      <title>Experiences of Test Automation</title>
      <genre>Computer</genre>
      <price>25.95</price>
      <publish_date>2012-01-09</publish_date>
      <description>Software test automation has moved beyond a luxu
   </book>
   <book id="bk105">
      <author>Crispin, Lisa</author>
      <title>Agile Testing: A Practical Guide</title>
      <genre>Computer</genre>
      <price>25.95</price>
      <publish_date>2008-12-30</publish_date>
      <description> Testing is a key component of agile development
   </book>
   <book id="bk106">
      <author>Graham, Dorothy</author>
      <title>Software Test Automation</title>
      <genre>Computer</genre>
```

```
    <price>14.95</price>
    <publish_date>1999-05-28</publish_date>
    <description> This book describes how to build and implement
  </book>
  <book id="bk107">
    <author>Hendrickson, Elisabeth</author>
    <title>Explore It!</title>
    <genre>Computer</genre>
    <price>14.95</price>
    <publish_date>2013-03-03</publish_date>
    <description> Uncover surprises, risks, and potentially seric
  </book>
  <book id="bk108">
    <author>Adzic, Gojko</author>
    <title>Specification by Example</title>
    <genre>Computer</genre>
    <price>24.95</price>
    <publish_date>2011-05-06</publish_date>
    <description> Specification by Example is an emerging practic
  </book>
  <book id="bk109">
    <author>Whittaker, James</author>
    <title>How Google Tests Software</title>
    <genre>Computer</genre>
    <price>16.95</price>
    <publish_date>2012-03-23</publish_date>
    <description> Do you need to get it right, too? Then, learn
  </book>
  <book id="bk110">
    <author>O'Brien, Tim</author>
    <title>Microsoft .NET: The Programming Bible</title>
    <genre>Computer</genre>
    <price>36.95</price>
    <publish_date>2000-12-09</publish_date>
    <description>Microsoft's .NET initiative is explored in detai
  </book>
  <book id="bk111">
    <author>O'Brien, Tim</author>
    <title>MSXML3: A Comprehensive Guide</title>
    <genre>Computer</genre>
    <price>36.95</price>
    <publish_date>2000-12-01</publish_date>
    <description>The Microsoft MSXML3 parser is covered in detail
  </book>
  <book id="bk112">
    <author>Galos, Mike</author>
    <title>Visual Studio 7: A Comprehensive Guide</title>
    <genre>Computer</genre>
    <price>49.95</price>
    <publish_date>2001-04-16</publish_date>
    <description>Microsoft Visual Studio 7 is explored in depth,
  </book>
</catalog>
```

3. At the end, click on **OK**. The **XML Checkpoint Properties** dialog will appear, as shown:



Now, for each node we will be able to define whether we wish to:

- Check the number of attributes it has (if any).
- Check the number of child elements.
- Limit our verification to a particular type of child (relevant when more than a single type is present). In our example, under the **catalog** root node, we only have **book** nodes so it does not make any difference.

When traversing the hierarchy we can view the specific values of nodes, as shown:

We can also click on the **Activate Schema Validation** button to validate the integrity of the XML document with regards to a schema (XSD), either referenced in the document or an external one:

In case our, XML is expected to be based on such a schema, this would give us a comprehensive checkpoint, which not only verifies the contents of the XML document but also its structure.

4. At the end of the definitions, we click on **OK**, and the following statement is inserted in `Action`:

```
XMLFile("catalog.xml").Check CheckPoint("catalog.xml")
```

In addition, our Object Repository now includes `XMLFile` as TO and a `CheckPoint` object of `XMLFile`. The `XMLFile` TO carries a single description property, that is, its filename, which stores its path. Of course, as with other TOs, it is possible to parameterize this property. In a similar fashion, as with the `DbTable` checkpoint, the `CheckPoint` object of `XMLFile` will have the properties as we defined in the **XML Checkpoint Properties** dialog previously.

How it works...

When invoking the `Check` method of `XMLFile` with the XML `CheckPoint` object, the target file is opened and checked against the data stored as expected results. Running the command yields a results report, as follows (here we have deliberately changed one value in the XML file to make the checkpoint fail):

If prompted to allow ActiveX Add-In, then approve in order to see **Captured Data**. Clicking on the **View XML Checkpoint Results** button in that pane will open a window titled **XML Checkpoint Results**:

We see that for each node there is a checkpoint summary giving details about the checks that were performed. As the tooltip shows, we can browse through our results very efficiently to drill-down and examine the
licking on the icon to find the next error will lead us to the
hich UFT found a discrepancy between the expected and
a: