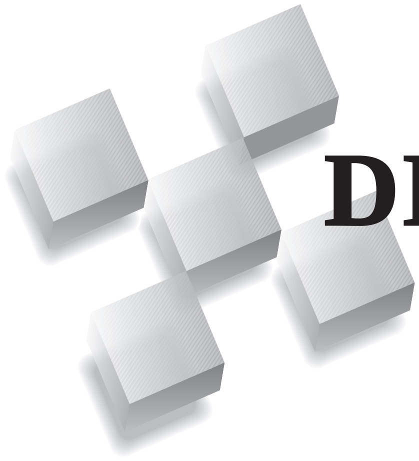


DBMaker

XML Tool User's Guide

CASEMaker®

CASEMaker Inc./Corporate Headquarters

1680 Civic Center Drive

Santa Clara, CA 95050, U.S.A.

www.casemaker.com

www.casemaker.com/support

Contents

1 Introduction

DBMaker includes two Java-based, platform-independent tools for passing data between a database and XML documents. The XML Transfer Template tool and the XML Transfer Mapping tool allow you to create custom templates that determine how data maps from a database to XML files.

Once you have created a template, you can use one of the APIs provided by DBMaker to help automate the process of synchronizing data between the database and XML files. DBMaker provides stored procedures as well as APIs in C, C++, and Java to help you accomplish this task.

2 XML Transfer Template Tool

The purpose of the XML Transfer Template (XTT) tool is to provide a customizable bridge between database data and XML documents. The bridge takes the form of a template file, the XML Transfer Template (XTT). The XTT file determines which database tables and columns to map to which XML elements and attributes. You determine the mapping using drag-and-drop operations in the XTT tool. The XTT tool ensures that XTT syntax is correct, and also aids in performing tasks such as generating schema documents (XSD) or document type definitions (DTD).

Using The XTT tool to transform data is a four-phase process – creating or importing the XTT structure, linking XTT objects to the database with SQL queries, generating DTD or XSD files if necessary, and finally generating the XML document.

Usually, linking XTT objects will only be necessary if you are importing an existing XML structure from either an XML file, XSD, or DTD. Likewise, linking will not be necessary if you are creating an XTT based on the database. Hybrid situations may exist, however; for example, where you have an existing XML structure but need to add new elements for new data.

XTT files define a map from database tables and columns to the elements and attributes of an XML file. An XTT file is a document with syntax similar to a valid XML document. Elements define tables and columns, and attributes define SQL queries, attribute names and values, and element values for XML documents generated using the XTT.

➲ **Example:**

The following is a complete XTT file that maps data from the table CARD. It maps the columns FIRSTNAME, LASTNAME, and TITLE as attributes of the element CARD, and the column NUM as a child element:

```
<?xml version="1.0" encoding="UTF-8"?>
<xtt:template xmlns:xtt="urn:schema-dbmaker-com:xml-template">
    <root>
          <CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME,
LASTNAME, TITLE, BMP from SYSADM.CARD">
              <xtt:attribute name="FIRSTNAME" value="$CARD_SQL0.FIRSTNAME"/>
              <xtt:attribute name="LASTNAME" value="$CARD_SQL0.LASTNAME"/>
              <xtt:attribute name="TITLE" value="$CARD_SQL0.TITLE"/>
              <NUM xtt:textvalue="$CARD_SQL0.NUM" />
          </CARD>
    </root>
</xtt:template>
```

If the XTT in the above example is run, the following XML file is generated:

```
<?xml version="1.0" encoding="US-ASCII" ?>
 <root>
  <CARD FIRSTNAME="Eddie" LASTNAME="Brown" TITLE="Manager">
   <NUM>1</NUM>
  </CARD>
 </root>
```

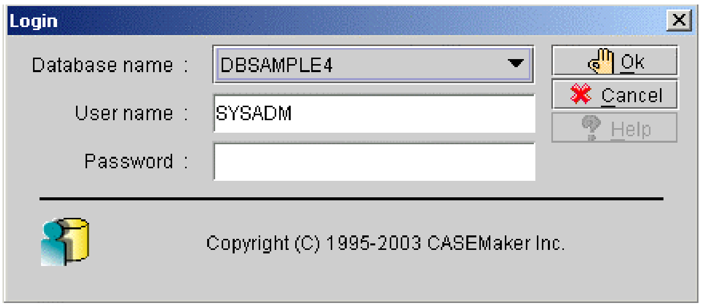
The XTT tool provides a simple user interface for scripting, validating and running XTT files. The following sections describe the user interface, and give procedures to help you quickly learn to start creating your own XTT files.

2.1 Getting to Know the XTT Tool

This section describes the elements of the XTT Tool user interface and how to log onto the database.

Opening the XTT tool and logging into a database

When you open the XTT tool from the Windows start menu you will automatically be prompted to log into the database. Select the database from which you want to export information. You must have an account on the database in order to log in. Be sure to use an account that has access to all the tables that you will need information from.



Database name : DBSAMPLE4
User name : SYSADM
Password :

Copyright (C) 1995-2003 CASEMaker Inc.

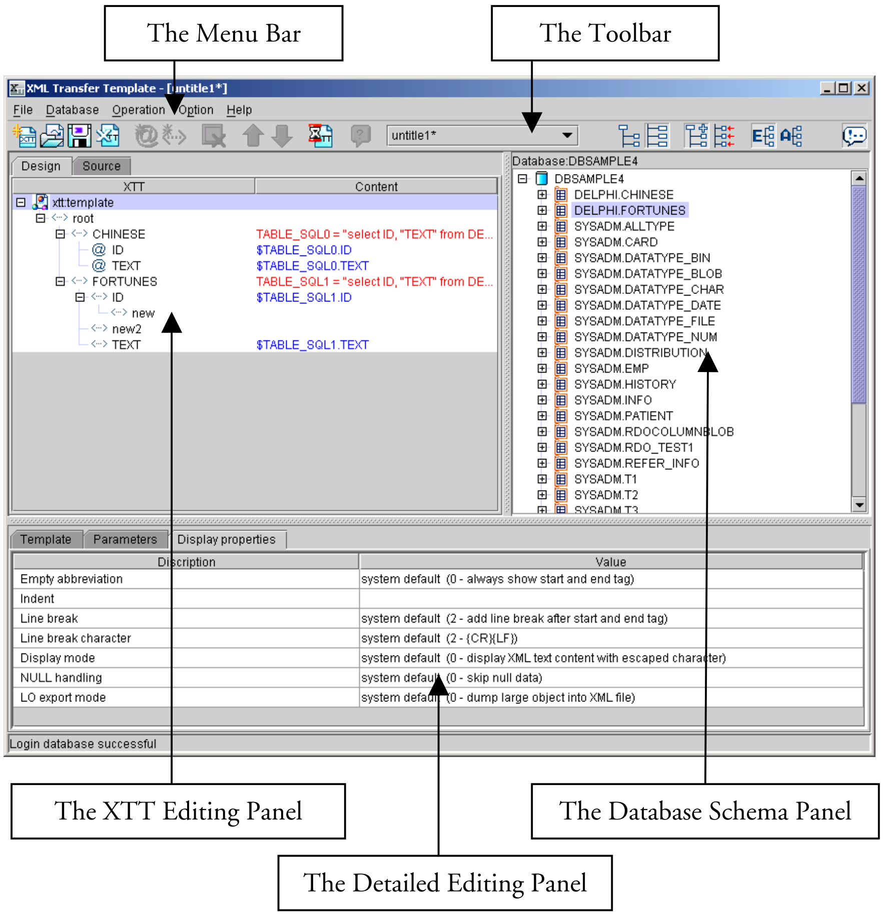
Figure 2-1: The login dialog

➲ **To open the XTT tool and log into a database:**

1. From the Windows **Start** menu, click **Start** > **Programs** > **DBMaker 4.1** > **XML Transfer Template**.
2. In the **Login** dialog, select a database and enter a user name and password
3. Click **Ok**. The XTT tool will display database tables in the Database Schema Panel.

The Main Console

The Main Console can be divided into five logical areas. Refer to Figure 2-2, below.


The Menu Bar
The Toolbar
XML Transfer Template - [untitle1*]
File Database Operation Option Help
untitle1*
Design Source
Database:DBSAMPLE4
Template Parameters Display properties
The XTT Editing Panel
The Database Schema Panel
The Detailed Editing Panel

XTT	Content
xtt:template	
root	
CHINESE	TABLE_SQL0 = "select ID, "TEXT" from DE...
@ ID	$TABLE_SQL0.ID
@ TEXT	$TABLE_SQL0.TEXT
FORTUNES	TABLE_SQL1 = "select ID, "TEXT" from DE...
ID	$TABLE_SQL1.ID
new	
new2	
TEXT	$TABLE_SQL1.TEXT

```
DBSAMPLE4
  DELPHI.CHINESE
  DELPHI.FORTUNES
  SYSADM.ALLTYPE
  SYSADM.CARD
  SYSADM.DATATYPE_BIN
  SYSADM.DATATYPE_BLOB
  SYSADM.DATATYPE_CHAR
  SYSADM.DATATYPE_DATE
  SYSADM.DATATYPE_FILE
  SYSADM.DATATYPE_NUM
  SYSADM.DISTRIBUTION
  SYSADM.EMP
  SYSADM.HISTORY
  SYSADM.INFO
  SYSADM.PATIENT
  SYSADM.RDOCOLUMNBLOB
  SYSADM.RDO_TEST1
  SYSADM.REFER_INFO
  SYSADM.T1
  SYSADM.T2
  SYSADM.T3
```

Discription	Value
Empty abbreviation	system default (0 - always show start and end tag)
Indent	
Line break	system default (2 - add line break after start and end tag)
Line break character	system default (2 - {CR}{LF})
Display mode	system default (0 - display XML text content with escaped character)
NULL handling	system default (0 - skip null data)
LO export mode	system default (0 - dump large object into XML file)

Login database successful

Figure 2-2: Elements of the Main Console

The Menu Bar

The Menu Bar consists of five menus: **File**, **Database**, **Operation**, **Options**, and **Help**. Menu item are disabled if they cannot be used. Refer to the following sections for each menu item's function.

FILE

The **File** menu consists of the following items:

- **New XTT** > **Empty XTT**: Creates a new empty XTT. Refer to *Creating an empty XTT file* for more information
- **New XTT** > **With imported DTD**: Creates a new XTT based on a DTD file. Refer to *Creating an XTT from a DTD file* for more information
- **New XTT** > **With imported XSD**: Creates a new XTT based on an XSD file. Refer to *Creating an XTT from an XSD file* for more information
- **New XTT** > **With imported XML**: Creates a new XTT based on an XML file. Refer to *Creating an XTT from an XML file* for more information
- **Open XTT**: opens the **Open** dialog with .XTT as the default file extension filter.
- **Close**: closes the XTT currently open in the XTT editing panel. If the XTT has been modified a confirmation dialog will ask to save changes.
- **Save**: saves the XTT currently open in the XTT editing panel. If the XTT has not been saved before, the **Save as** dialog will open.
- **Save as**: Opens the **Save as** dialog with .XTT as the default file extension.
- **Generate DTD**: Opens the **Save as** dialog with .DTD as the default file extension. Refer to *Generating a DTD* for more information.
- **Generate XSD**: Opens the **Save as** dialog with .XSD as the default file extension. Refer to *Generating an XSD* for more information
- Recent files: displays the most recently opened files
- **Exit**: exits the XML Transfer Template tool

DATABASE

The **Database** menu consists of the following items:

- **Connect**: Opens the **Login** dialog. A list of running databases appears in the drop down menu.
- **Disconnect**: stops the session with the database. The content in the **database schema panel** is cleared.
- **Refresh**: refreshes the **database schema panel** if a session is active.

OPERATION

The **Operation** menu consists of the following items:

- **Insert** > **Element**: inserts a new empty element into the XTT object tree. Refer to *Adding New Elements and Attributes* for more information
- **Insert** > **Attribute**: inserts a new empty attribute into the XTT object tree. Refer to *Adding New Elements and Attributes* for more information
- **Undo**: returns the XTT object tree to the state it was in before the last modification
- **Copy**: copies the selected node of the XTT object tree and all descendants
- **Cut**: cuts the selected node of the XTT object tree and all descendants
- **Paste**: Pastes the last cut or copied node of the XTT object tree and all descendants.
- **Remove**: removes the selected node of the XTT object tree and all descendants
- **Run**: executes the XTT file. Refer to *Generating XML data* for more information.
- **Validate**: checks if variable references for elements exist in the parent element, and checks if SQL commands in elements are valid in the database

OPTIONS

The **Options** menu consists of the following items:

- **Preferences**: opens the user preferences dialog
- **Tree operation options**: opens the tree operation options dialog

HELP

The **Help** menu consists of the following items:

- **Help**: opens the online help.
- **web site**: opens a browser window to the web site www.dbmaker.com.
- **about**: displays information about the XML transfer template including the build date and number, the CASEMaker technical support e-mail address, and a link to www.dbmaker.com.

The Toolbar

This section shows the toolbar items with their equivalent menu bar operations.

FILE OPERATIONS

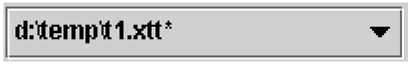
- New empty XTT= menubar > File > New XTT > Empty XTT
- Open XTT= menubar > File > Open XTT
- Save= menubar > File > Save
- Close= menubar > File > Close

XTT TREE OPERATIONS

- Add Attribute Object = menubar > Operation > Insert > Attribute
- Add Element Object = menubar > Operation > Insert > Element
- Remove Tree Node = menubar > Operation . Remove
- Move Up = Move current selected node before its previous sibling node. The element can be moved before another element but cannot be moved before an attribute.

- Move Down = Move current selected node after its next sibling node. The attribute can only be moved after another attribute but can't be moved after an element.

OPENED FILES

- 
  d:\temp\t1.xtt*

  The combo box displays all open XTT filenames. If the file has been edited, then there will be an asterisk (*) after the filename. When you choose a different filename, the XTT edit panel will reload and show the XTT tree of the newly selected file.

OPERATION OPTIONS

- Run Transfer = menubar > Operations > Run
- Help = menubar > Help > Help
- Insert as Child = Insert elements as child elements (when performing drag-and-drop operations)
- Inset as Sibling = Insert elements as sibling elements (when performing drag-and-drop operations)
- = Add a new node (when performing drag-and-drop operations)
- = Link by source structure (when performing drag-and-drop operations)
- = Add a new element below the selected element (follows **Insert as …** rule)
- = Add a new attribute within the selected element
- = Operation options – when selected, causes **Customize** dialog to appear when performing drag-and-drop operations on tables or views.

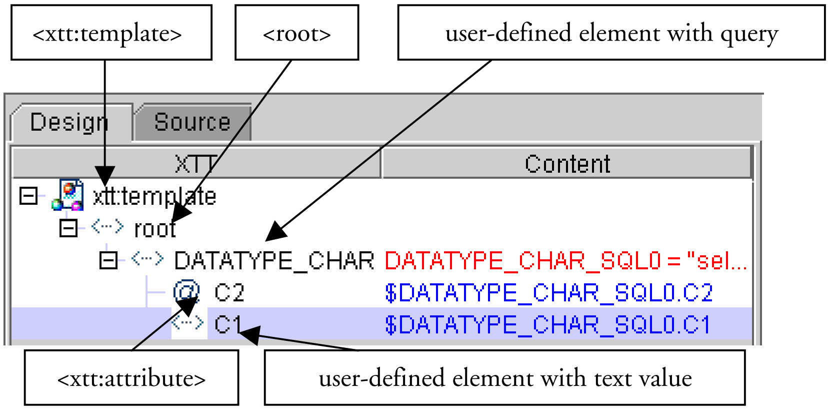
The XTT Editing Panel

The XTT editing panel consists of two views: the design view and the source view. The design view contains the XTT object tree and is displayed by default when an XTT is created or opened. The XTT object tree is a logical representation of the XTT file itself, which is a well-formed XML document. The XTT object tree consists of five types of elements, described in Table 2-1.

XTT object type	Tree node name	Content description
<xtt:template>	xtt:template	(none)
<root>	root	(none)
<xtt:attribute>	value of attribute ‘name’	value of attribute ‘value’
user-defined element without query or text value	element’s tag name	(none)
user-defined element with query	element’s tag name	value of attribute ‘xtt:query’ + ‘=’ + value of attribute ‘xtt:command’
user-defined element with text value	element’s tag name	value of attribute ‘textvalue’

Table 2-1: Element types in the XTT object tree.

The five XTT element types as they appear in the design view are illustrated in Table 2-2.


<xtt:template>
<root>
user-defined element with query
<xtt:attribute>
user-defined element with text value
Design
Source
XTT
Content

```
xtt:template
  root
    DATATYPE_CHAR   DATATYPE_CHAR_SQL0 = "sel...
      C2            $DATATYPE_CHAR_SQL0.C2
      C1            $DATATYPE_CHAR_SQL0.C1
```

Table 2-2: Design view of element types in the XTT object tree

To select an XTT tree object, left-click on it, and it will be highlighted in blue. If you right click on the design view, a pop-up menu will appear depending on the object that is highlighted. Right-clicking on an object will not cause it to be selected. Table 2-3 summarizes the pop-up menu contents for different selected elements.

Node type	**Pop-up menu**
<xtt:template>	N/A
attribute	● 'Change to ELEMENT' - the selected attribute node will be replaced by an element with the same name. The data in the **value** attribute will be copied to the **textvalue** attribute of the new element ● Cut – same as ctrl-X ● Copy – same as ctrl-C ● 'Remove' - removes the current selected node from the tree

user-defined element	● Element – creates a new element below the selected element. Whether the new element is inserted as a sibling or child element depends on whether the **Insert as Child** or **Insert as Sibling** option has been selected ● Attribute – creates a new attribute for the selected element ● 'Change to ATTRIBUTE' – only provided if the object has no sub node (either attribute or element). The selected element will be replaced by an attribute with the same name. The data in **textvalue** will be copied to the **value** attribute of the new attribute ● Cut – same as ctrl-X ● Copy – same as ctrl-C ● Paste – same as ctrl-V ● 'Remove' - to remove the current selected node from the tree

Table 2-3: pop-up menus available in the XTT editing panel design view

The **Source** tab displays the source code for the XTT file.

The Database Schema Panel

The database schema panel shows the table/views of the connected database as a schema tree. By expanding the table/view node, it will show each column as its sub node. The nodes in the schema tree can be dragged into the XTT tree panel. It will then add a new node or link the schema information into XTT tree depending on he settings selected

The Detailed Editing Panel

Detail editing panel shows the detailed properties of the selected node in the XTT tree. The general rule for text fields in the editing panel is that when an object outside the text field or area is selected the value is set. For example, the name field will be set after the next field is selected or the XTT tree selection is changed. The change can be seen on the XTT tree.

XTT:TEMPLATE

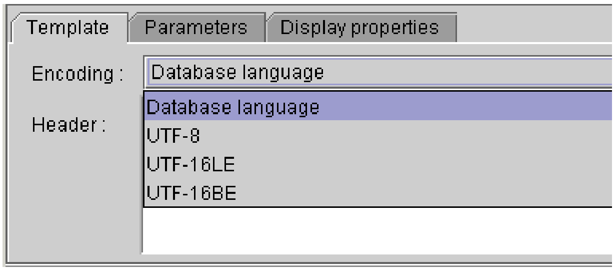
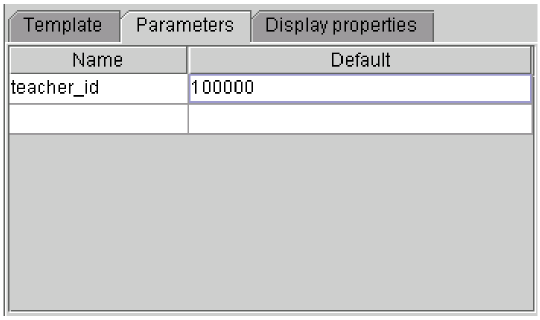


Figure 2-3: The language encoding menu of the detailed editing panel

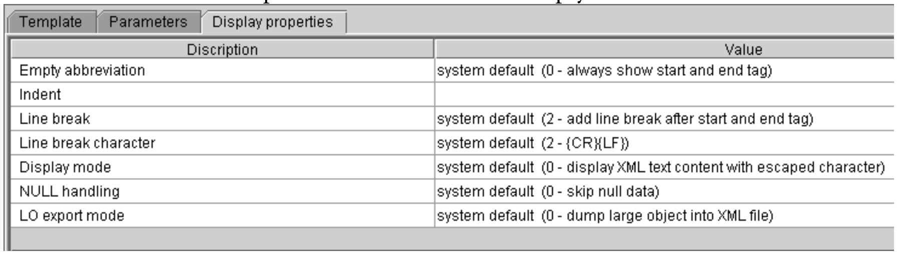
Encoding – The **Encoding** menu specifies the text encoding for any output XML file. The choices are database local (the text encoding specified in the database), UTF-8, UTF-16LE and UTF-16BE. The default is database local. If database local is Big5 and the encoding setting is database local, then the output XML will be encoded in Big5.

Header – The header box it is where you add information like a schema file or applicable XSL in the output XML file. The XTT engine will print the content in this block to the output XML file after <?xml version="1.0"…?> . Be sure to type valid XML content in this header block.



Name	Default
teacher_id	100000

Parameters – you may add as many parameters as you want. The parameter name must be unique. Press the 'delete' key to remove a selected row. The last empty row cannot be removed. The parameter default can be empty.



Discription	Value
Empty abbreviation	system default (0 - always show start and end tag)
Indent	
Line break	system default (2 - add line break after start and end tag)
Line break character	system default (2 - {CR}{LF})
Display mode	system default (0 - display XML text content with escaped character)
NULL handling	system default (0 - skip null data)
LO export mode	system default (0 - dump large object into XML file)

There are a few display settings available for the output XML file. The default value for each setting is 'system default'. The display settings will apply to all XTT objects unless specified otherwise.

Empty abbreviation –

0 – always show start and end tags for an element, even if its content is empty.

1 – to use the abbreviated form of the end tag if no sub element is produced. For example, **<Department id="1001" />.**

2 – hide the start and end tags if the element content is empty; no text, no attribute, no child element. If the element only has attributes, it will use type 1 abbreviation.

Indent – the number of indent spaces in the source document for each sub level as displayed in a text editor. For example, if the number is 2, then the start tag of the root will be indented 2 spaces, and a sub node of the root will be indented 4 spaces.

Line break –

0 – do not add any line break.

1 – add a line break after an end tag.

2 – add a line break after every start and end tag.

Line break char – the character(s) to be added as line break.

0 – {CR}

1– {LF}

2 – {CR}{LF}.

Display mode – the way to display text data. Generally, the text character '<' will be replaced with '&lt;' in the text content. But '<' can be used if it is enclosed in the CDATA section, or the text content itself is already an XML fragment and can be added into the output XML as a part of the XML content. There are 3 different ways to display text data.

0 – display XML text content with the escape character.

1 – use a CDATA section to enclose the text value.

2 – make the content native XML text.

Null handling – specify how to handle null data.

0 – skip null data. If it's an attribute, then do nothing.

1 – show empty content. For example, <NAME></NAME>.

2 – display 'NULL'. For example, <NAME>NULL</NAME>.

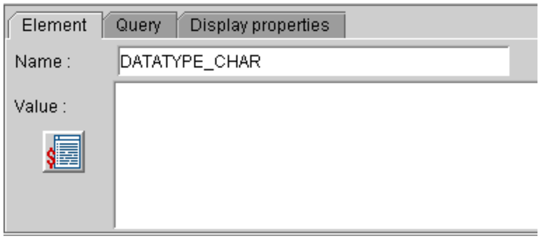
LO mode – specify how to handle large objects.

0 – dump large object data directly into the XML file. Print a string if it is CLOB type data or print in hexadecimal format if it is BLOB type data.

1 – to store large object data in external file.

USER-DEFINED ELEMENT

The following represents the properties of a user-defined element in the XTT object tree.



Name : DATATYPE_CHAR

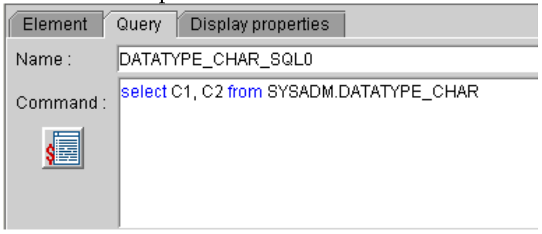
Value :

Figure 2-4: The detailed editing panel for a user-defined element

Name – the element name. It is case-sensitive and cannot be empty.

Value – the text value. This is a text expression, which can have both constant text and a variable reference. For example, you might want to add the country code to all phone number data, such as "886 - $SQL1.PHONE", where "886 –" is the constant text and "$SQL1.PHONE" is the variable reference. The XTT engine will concatenate both into the output XML file as the text value of the element.

Browse button – the browse button will list all the available parameters and variable references at the current level. You can choose the variable name(s) and insert them into the text expression field next to the browse button.



Name : DATATYPE_CHAR_SQL0

Command :

```
select C1, C2 from SYSADM.DATATYPE_CHAR
```

Figure 2-5: The query view for a user-defined element in the detailed editing panel

An element can have embedded query properties in it. A valid XTT element will have both fields (name and command) specified or both left as empty.

Name – the query name. It will be used as a variable reference in the sub nodes.

Command – SQL query statement. The statement must be able to generate a result set. For example, you cannot type a **delete table** statement here.

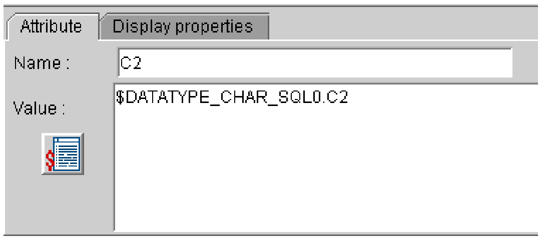
The display settings are similar to the ones in xtt:template, But the default is 'follow template setting'.

Null handling – there are only two choices instead of three. If 'follow template setting' is selected and the template has the null handling setting as '0 - skip null data', then it will be treated as '1 show empty content'.

1 – show empty content. For example, <NAME></NAME>.

2 – display 'NULL'. For example, <NAME>NULL</NAME>.

ATTRIBUTE NODE



Name : C2

Value : $DATATYPE_CHAR_SQL0.C2

Figure 2-6: An attribute viewed in the detailed editing panel

Name – attribute name. The Name field cannot be empty. Multiple attributes must have names unique to the same parent element.

Value – the attribute value; a text expression field.

There are only two display settings for an attribute node; null handling and LO mode. The default for both is 'follow template setting'.

The Customize Dialog

The customize dialog appears when performing drag-and-drop operations if you have selected **show customize dialog**. The appearance of the customize dialog depends on the settings selected in the tree operation options dialog. There are two main views for the customize dialog depending on whether you have selected **Link by source structure** or **Add a new node**.

The User Preferences Dialog

The user preferences dialog is where you select the user interface language, and the method by which you wish to view the results. You may select English, Chinese, or Japanese as the user interface language. You may also choose to select your default XML browser or your default text editor to view the output when running the XTT.

The Tree Operation Options Dialog

The tree operation options dialog is where you select the behavior of drag-and-drop operations. You may select to add database objects as elements or attributes; add objects as child nodes or sibling nodes; and to add objects as new trees, or to link data to existing elements or attributes.

2.2 Creating a New XTT

An XTT is the map by which XML data files are produced. An XTT may be created in one of four ways: from an empty XTT file, from a DTD file, From an XSD file, and from an XML file.

Creating an empty XTT file

Creating an empty XTT file is useful if you have data in a database that you want to display in XML form, but have no preconditions for how the XML data must be formatted. An empty XTT file consists only of the root node. After creating an empty XTT file, you may add elements and attributes. To learn about adding elements and attributes refer to *Editing an XTT*.

➲ **To create an empty XTT file:**

1. Open the XTT tool and log in to the database that you want to use.
2. Click **File** > **New XTT** > **Empty XTT**. The root node will appear in the XTT object tree.

Creating an XTT from a DTD file

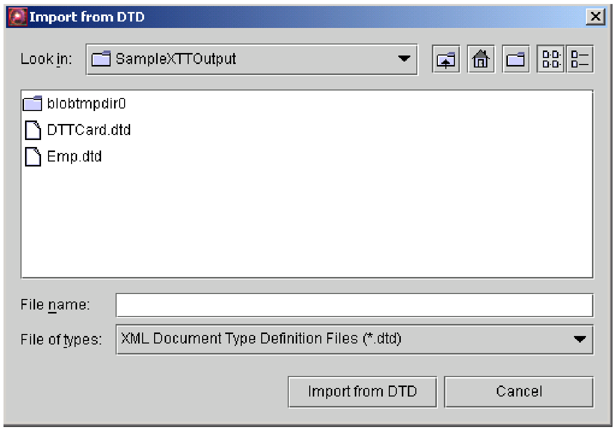
You may want to define the structure of your XML documents based on an existing Document Type Definition (DTD) file. If you have an existing XML document based on an external DTD and want to produce XML files from database data that conforms to that DTD, then you can create an XTT from a DTD file.

A root element must be specified for an XTT. During the import process a dialog allows you to select any valid element definition as the root for the XTT.

After creating the XTT, all the elements and attributes under the selected root node of the DTD will appear in the XTT object tree, however, none of the element or attribute definitions contain values – there is no method to pass SQL data to an XML file using the XTT. This may be accomplished by editing element nodes in the XTT object tree. For details on how to modify nodes in the XTT object tree, refer to Mapping Data to Elements.

➲ **To create an XTT from a DTD:**

1. Open the XTT tool and log in to the database that you want to use.
2. Click **File** > **New XTT** > **With imported DTD**.
3. In the **Import from DTD** dialog, select the DTD file to import and click **Import from DTD**.

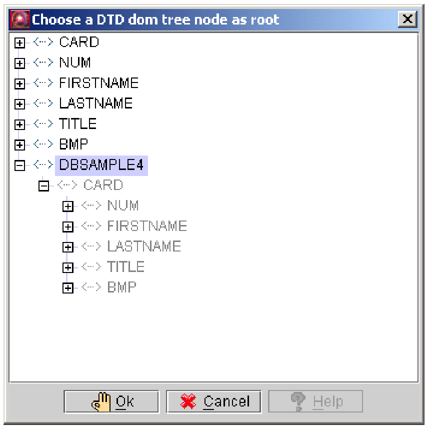

Import from DTD

Look in: SampleXTTOutput

blobtmpdir0
DTTCard.dtd
Emp.dtd

File name:
File of types: XML Document Type Definition Files (*.dtd)

4. In the **Choose a DTD dom tree node as root** dialog, select an element definition to be the root element in the XTT object tree and click **Ok**.

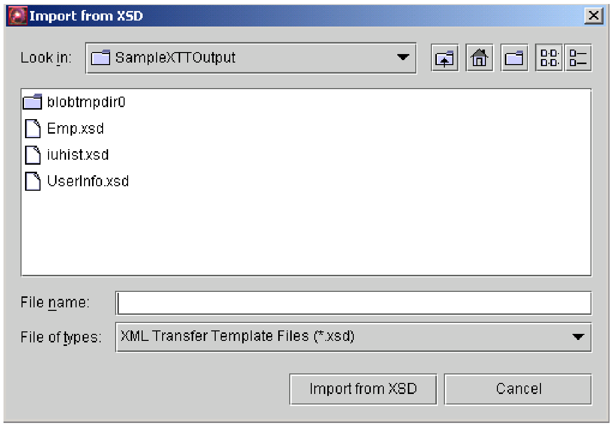


Creating an XTT from an XSD file

It is also possible to base the XTT object tree structure on an XML schema defined by an XML Schema Definition (XSD) document file. As with DTD files, a root element must be specified, and you may select any valid element definition as the root for the XTT. Also, none of the newly created attribute or element definitions in the XTT object tree has a value; you must modify the element and attribute definitions in order to get SQL data into an XML file. For details on how to modify nodes in the XTT object tree, refer to *Editing an XTT*.

➲ **To create an XTT from an XSD file:**

1. Open the XTT tool and log in to the database that you want to use.
2. Click **File** > **New XTT** > **With imported XSD**.
3. In the **Import from XSD** dialog, select the DTD file to import and click **Import from XSD**.



4. In the **Choose an XSD dom tree node as root** dialog, select an element definition to be the root element in the XTT object tree and click **Ok**.

Creating an XTT from an XML file

If you do not have a DTD or XSD file to base your XTT structure on, but need to maintain consistency with an existing XML structure, then you may create an XTT directly from an XML file. The XTT tool will parse the structure of your XML document to generate an XTT object tree. The primary difference between this method of creating an XTT versus using a DTD or XSD is that you are not given an option as to which element will constitute the root node of the XTT object tree.

➲ **To create an XTT from an XML file:**

1. Open the XTT tool and log in to the database that you want to use.

2. Click **File** > **New XTT** > **With imported XML**.

3. In the **Import from XML** dialog, select the DTD file to import and click **Import from XML**.

2.3 Editing an XTT

After you have created the root node and structure (if creating an XTT from XML, DTD, or XSD), you will want to provide content for the generated XML file. For blank XTTs, this is primarily a drag-and-drop operation from the tables in the database schema panel into the XTT object tree. The sections *Inserting a Table* and *Adding New Elements and Attributes*, describe the primary tasks you will need to accomplish if creating a new XTT.

For XTTs based on an XML, DTD, or XSD file, you will need to add query statements and values to the attribute and element definitions in the XTT object tree. These tasks are described the section *Mapping Data to Elements and Attributes*.

These tasks are not mutually exclusive and the above guidelines are only provided to enable you to quickly understand how to create a valid XTT document. At times you may find it useful to modify an element definition in a new XTT – one example being if you only wish to select values from an SQL table that meet some conditions. Or you may not need to conform precisely to the XML schema that your XTT is based on – in which case it is possible to use drag-and-drop operations to build your XTT object tree.

About the Design View

The design view of the XTT editing panel displays the XTT object tree. For a new, blank XTT, the XTT object tree contains only the XTT template node and an empty root node. An XTT object tree that has been created from an XML, XSD, or DTD will have a different root. Refer to *The XTT Editing Panel* for detailed information on the objects and functions available in the design view.

Inserting a Table

You can insert a table as a child or sibling. The first table you insert must be a child of the root node; attempting to add an element to the xtt:template node will return an error.

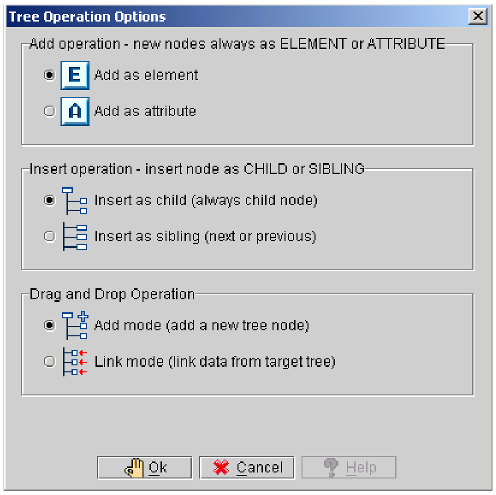
Before a table is added to the XTT object tree, you can choose which columns are to be added as elements, which columns to add as attributes, which columns to select but not add to the XTT object tree, and which columns not to select. This is accomplished in the customize dialog. Click the **show customize dialog** button if you want the customize dialog to appear when adding object to the XTT object tree.

When inserting a table, the customize dialog will display a query object name, and the structure of the database object as it will appear after it is entered into the XTT object tree. By default both parent and child objects are inserted as elements. You may also choose to insert child objects as attributes, to select database objects without inserting them as attribute objects or element objects, or to not select the database objects.

After you have inserted the first table, subsequent tables may be added as children or siblings of the first table. Tables must always be represented as elements in the XTT object tree. If you have created in the XTT an existing file, any tables you insert can be children or siblings of any of the existing XTT elements

➲ **To insert a table as a child node:**

1. Click Option > Tree Operation Options.
2. In the **Tree Operation Options** dialog click **Add as element**, **Insert as a child**, and **Add mode**.



Tree Operation Options

Add operation - new nodes always as ELEMENT or ATTRIBUTE

[✓ Add as element]

Add as attribute

Insert operation - insert node as CHILD or SIBLING

[✓ Insert as child (always child node)]

Insert as sibling (next or previous)

Drag and Drop Operation

[✓ Add mode (add a new tree node)]

Link mode (link data from target tree)

3. Select a table from the database schema panel

4. Drag the table from the database schema panel to the XTT object tree node that is to be the parent of the new node.

5. In the customize dialog, select any columns that you do not want to include and click **remove**. Select any columns that you wish to select in the SQL command, but do not want to export to the XML file and click **hidden**. Select any columns that you wish to add as attributes and click **attribute**. Select any columns that you wish to add as elements and click **element**.

6. Click **OK**. The table will appear as a new element in the XTT object tree. Columns will appear as elements and attributes, depending on how they were selected in the customize dialog.

Adding New Elements and Attributes

You may need to add elements or attributes to the XTT object tree in order to create the desired data structure. Note that the following procedure only describes how to add empty elements and attributes. Refer to defining elements and defining attributes for information on how to map data from the database to empty elements in the XTT object tree.

➲ **To add new elements or attributes:**

1. Select the element to which you will add a new element or attribute.
2. Determine the relationship between the selected element and your new object: click **insert as sibling** to make the new object a sibling of the selected element, click **insert as child** to make the new object a child of the selected element.
3. Click **add attribute object** to add an attribute. Click **add element object** to add a new element.
4. Type a name for the new object in the name box of the **detailed editing panel** and press Enter.

Mapping Data to Elements and Attributes

Elements should be associated with data. To associate an element object with data, the parent element object must contain an SQL query. The SQL query must select the table and column that will map to the child element object. After associating the parent element with an SQL query, you must associate a child element with the column.

Use the link by source structure tree operation option to associate elements with SQL queries. When you associate an element with an SQL query, use the customize dialog to associate child elements or attributes with the selected columns of the SQL query. Dragging a table from the database schema panel into a parent element in the XTT object tree will cause the customize dialog to open.

When linking element objects by source structure, the customize dialog will show the query object name, and two columns: the XTT object column, and the mapping column. The XTT object column displays all child elements and attributes of the element that you have dragged the table into. The mapping column displays any existing content, and is where you select the column content that you want to map from. Click a row in the mapping column to select an SQL data source for the corresponding XTT object.

➲ **To map data to an empty XTT element object:**

1. Click a table in the database schema panel.

2. Drag the table to an element object in the XTT object tree. The element you dragged the table into will be the parent element.
3. In the **Customize dialog**, select the mapping that you wish to perform. For each element and attribute object:
   - **a)** Click the Mapping column that corresponds to the XTT object that you wish to map database data to.
   - **b)** Select the value for the XTT object from the drop-down menu. The value is the equivalent of the column data in the database. It appears in the format of the name of the SQL query followed by a dot, and then the column name.
4. Click OK when you have completed selecting mapping values for all the XTT object nodes that you wish to map to database columns.

Saving an XTT

After you have completed editing the XTT object tree, you should save the XTT file. The saved XTT file can be later recalled for modification, or called in a stored procedure to automatically pass database data to XML files.

To save an XTT click the **Save** icon, or click **File** > **Save** from the menu bar.

2.4 Generating a DTD

If your development requirements determine that you will need to create a Document Type Definition (DTD) file to describe the structure of generated XML files, then you can use the **Generate DTD** function in the XML Transfer Template tool.

The DTD file structure will be consistent with the structure of the source XTT

➲ **Example:**

Given the following XTT file:

```
<?xml version="1.0" encoding="US-ASCII"?>
<xtt:template xmlns:xtt="urn:schema-dbmaker-com:xml-template">
    <root>
          <CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME,
LASTNAME, TITLE, BMP from SYSADM.CARD">
              <NUM xtt:textvalue="$CARD_SQL0.NUM" />
              <FIRSTNAME xtt:textvalue="$CARD_SQL0.FIRSTNAME" />
              <LASTNAME xtt:textvalue="$CARD_SQL0.LASTNAME" />
              <TITLE xtt:textvalue="$CARD_SQL0.TITLE" />
              <BMP xtt:textvalue="$CARD_SQL0.BMP" />
          </CARD>
      </root>
</xtt:template>
```

The resultant DTD file will be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT root (CARD*) >
<!ELEMENT CARD (NUM, FIRSTNAME, LASTNAME, TITLE, BMP) >
<!ELEMENT NUM (#PCDATA) >
<!ELEMENT FIRSTNAME (#PCDATA) >
<!ELEMENT LASTNAME (#PCDATA) >
<!ELEMENT TITLE (#PCDATA) >
<!ELEMENT BMP (#PCDATA) >
```

➲ **To generate a DTD file from an XTT object tree:**

1. Ensure that the XTT that you want to convert to a DTD is open.
2. Click **File** > **Generate DTD**.
3. In the **Generate DTD** dialog, select an output path and specify a file name and encoding type. Possible encoding types include: **UTF-8**, **UTF-6LE**, **UTF-16BE**, and Local Code.

4. Click **Generate DTD**, a DTD file will be created in the selected folder.

2.5 Generating an XSD

You may wish to generate an XML schema file from the logical structure represented in the XTT file. Some tools may require a schema file to be able to parse XML data. The schema file structure is consistent with the XTT object tree structure. In the example below, a small XTT file is used to generate a schema file.

➲ **Example**

Given an XTT file with the following object tree structure:

```
<?xml version="1.0" encoding="US-ASCII"?>
<xtt:template xmlns:xtt="urn:schema-dbmaker-com:xml-template">
    <root>
          <CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME,
LASTNAME, TITLE, BMP from SYSADM.CARD">
              <xtt:attribute name="NUM" value="$CARD_SQL0.NUM"/>
              <xtt:attribute name="FIRSTNAME" value="$CARD_SQL0.FIRSTNAME"/>
              <xtt:attribute name="LASTNAME" value="$CARD_SQL0.LASTNAME"/>
              <xtt:attribute name="TITLE" value="$CARD_SQL0.TITLE"/>
              <BMP xtt:textvalue="$CARD_SQL0.BMP" />
          </CARD>
      </root>
</xtt:template>
```

The resultant schema file structure will be:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="root">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="CARD" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="CARD">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="BMP" />
    </xsd:sequence>
    <xsd:attribute name="NUM" type="xsd:int"/>
    <xsd:attribute name="FIRSTNAME" type="xsd:string"/>
    <xsd:attribute name="LASTNAME" type="xsd:string"/>
    <xsd:attribute name="TITLE" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="BMP" type="xsd:string"/>
</xsd:schema>
```

➲ **To generate an XSD file from an XTT object tree:**

1. Ensure that the XTT that you want to convert to an XSD is open.
2. Click **File** > **Generate XSD**.
3. In the **Generate XSD** dialog, select an output path and specify a file name and encoding type. Possible encoding types include: **UTF-8**, **UTF-6LE**, **UTF-16BE**, and Local Code.
4. Click **Generate XSD**, an XSD file will be created in the selected folder.

2.6 Generating XML data

After creating the required transfer template and optional DTD or XSD file, you are ready to generate an XML file using the data stored in the database.

The **Run transfer** function is most useful for testing the XTT. After you have created an XTT, you can use it to generate XML documents on demand and pass data to your XML application.

The following example shows a completed XTT template file and corresponding output file.

```
<?xml version="1.0" encoding="US-ASCII"?>
<xtt:template xmlns:xtt="urn:schema-dbmaker-com:xml-template">
    <root>
          <CARD xtt:query="CARD_SQL0" xtt:command="select NUM, FIRSTNAME,
LASTNAME, TITLE, BMP from SYSADM.CARD">
              <NUM xtt:textvalue="$CARD_SQL0.NUM" />
              <FIRSTNAME xtt:textvalue="$CARD_SQL0.FIRSTNAME" />
              <LASTNAME xtt:textvalue="$CARD_SQL0.LASTNAME" />
              <TITLE xtt:textvalue="$CARD_SQL0.TITLE" />
              <BMP xtt:textvalue="$CARD_SQL0.BMP" />
          </CARD>
      </root>
</xtt:template>
```

The corresponding output file for the preceding XTT file:

```
  <?xml version="1.0" encoding="US-ASCII" ?>
- <root>
- <CARD>
  <NUM>1</NUM>
  <FIRSTNAME>Eddie</FIRSTNAME>
  <LASTNAME>Chang</LASTNAME>
  <TITLE>Manager</TITLE>
  <BMP>lobdir5\blobfile0.tmp</BMP>
  </CARD>
- <CARD>
  <NUM>2</NUM>
  <FIRSTNAME>Hook</FIRSTNAME>
  <LASTNAME>Hu</LASTNAME>
  <TITLE>Software Engineer</TITLE>
  <BMP>lobdir5\blobfile1.tmp</BMP>
  </CARD>
  </CARD>
- <CARD>
  <NUM>7</NUM>
  <FIRSTNAME>Oscar</FIRSTNAME>
  <LASTNAME>Tseng</LASTNAME>
  <TITLE>Software Engineer</TITLE>
  <BMP>lobdir5\blobfile6.tmp</BMP>
  </CARD>
- <CARD>
  <NUM>8</NUM>
  <FIRSTNAME>Jerry</FIRSTNAME>
  <LASTNAME>Liu</LASTNAME>
  <TITLE>Manager</TITLE>
  <BMP>lobdir5\blobfile7.tmp</BMP>
  </CARD>
  </root>
```

3 XML Transfer Mapping Tool

The XML transfer mapping (XTM) tool allows you to pass XML data to a database using XSL transformations. The XML Transfer mapping tool consists of three parts: an XML schema part, which displays the schema of the XML file(s) that you are using as source data; an SQL database part, which displays the database tables; and an XTM part, which displays the mapping from the XML schema to the database tables.

Using the tool can be summarized in five basic steps: opening a source XML or XSD file to create a source XML schema, connecting to a database, creating an XTM structure from the database table, mapping elements and attributes from the source XML schema, and finally storing the XTM structure as an XSL file.

Once the XSL file is created, it can be used to transform any XML file that conforms to the source schema to database data.

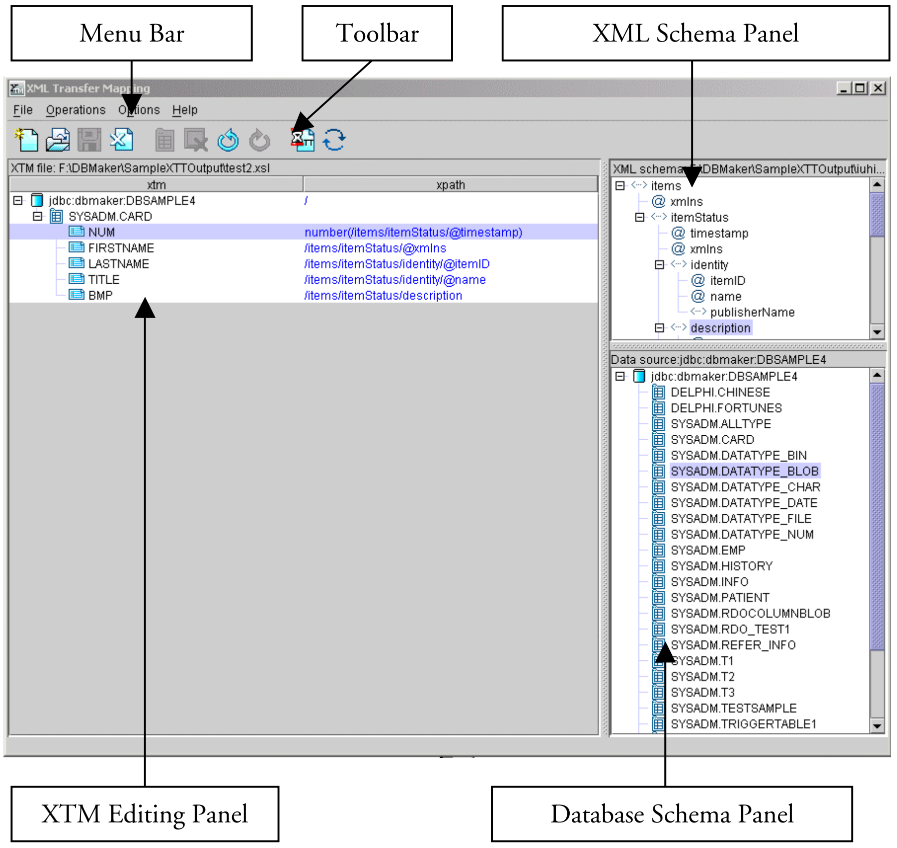
3.1 Getting to know the XTM Tool

Unlike the XTT tool, the XTM tool does not require a connection to the database when it is opened. The tool creates a database connection when you create or open an XTM, and is used to display the database schema tree, described in *XML Schema Tree*.

This chapter describes all of the major screen elements in the XTM tool.

The Main Console

The main console of the XTM tool can be broken down into five major areas as illustrated in Figure 3-1.



XTM file: F:\DBMaker\SampleXTTOutput\test2.xsl

xtm	xpath
jdbc:dbmaker:DBSAMPLE4	/
SYSADM.CARD	
NUM	number(/items/itemStatus/@timestamp)
FIRSTNAME	/items/itemStatus/@xmlns
LASTNAME	/items/itemStatus/identity/@itemID
TITLE	/items/itemStatus/identity/@name
BMP	/items/itemStatus/description

XML schema: ...\DBMaker\SampleXTTOutput\iuhi...

```
items
  @ xmlns
  itemStatus
    @ timestamp
    @ xmlns
    identity
      @ itemID
      @ name
      publisherName
    description
```

Data source:jdbc:dbmaker:DBSAMPLE4

```
jdbc:dbmaker:DBSAMPLE4
  DELPHI.CHINESE
  DELPHI.FORTUNES
  SYSADM.ALLTYPE
  SYSADM.CARD
  SYSADM.DATATYPE_BIN
  SYSADM.DATATYPE_BLOB
  SYSADM.DATATYPE_CHAR
  SYSADM.DATATYPE_DATE
  SYSADM.DATATYPE_FILE
  SYSADM.DATATYPE_NUM
  SYSADM.EMP
  SYSADM.HISTORY
  SYSADM.INFO
  SYSADM.PATIENT
  SYSADM.RDOCOLUMNBLOB
  SYSADM.RDO_TEST1
  SYSADM.REFER_INFO
  SYSADM.T1
  SYSADM.T2
  SYSADM.T3
  SYSADM.TESTSAMPLE
  SYSADM.TRIGGERTABLE1
```

Figure 3-1: Elements of the XTM main console

The Menu Bar

The menu bar consists of four menus: **File**, **Operations**, **Options**, and **Help**. Menu item are disabled if they cannot be used. Refer to the following sections for each menu item's function

The **File** menu consists of the following items:

- **New**: opens the **New XTM** dialog, which prompts you to enter source schema and database information
- **Open**: opens the **Open** dialog with .XSL as the default file extension filter.
- **Save**: saves the XTM currently open in the XTM editing panel as an XSL file. If the XTM has not been saved before, the **Save as** dialog will open.
- **Save as**: Opens the **Save as** dialog with .XSL as the default file extension.
- **Close**: closes the XTM currently open in the XTM editing panel. If the XTM has been modified a confirmation dialog will ask to save changes
- Recent files: Lists the XSL files opened during the current session
- **Exit**: Exits the XTM tool. Opens the **Save as** dialog with .XSL as the default file extension if the current XTM has not been saved

OPERATION

The **Operation** menu consists of the following items:

- **Undo**: returns the XTT object tree to the state it was in before the last modification
- **Redo**: Executes the last performed action again
- **Insert**: Inserts a new XTM control node. Opens the **New XTM Control Node** dialog
- **Remove**: Removes the selected node of the XTM object tree and all descendants

- **Run**: Opens the **Execute XTM** dialog. The XTM dialog offers the option to immediately execute the XTM and send data to the database or to generate an SQL script for later use
- **Refresh**: Queries the database to refresh database schema

OPTIONS

The **Options** menu consists of the following items:

- **Preferences**: opens the **Preferences** dialog, which allows you to choose the language that the UI is displayed in, and to specify different syntax for the connect section of the XSL file
- **JDBC Drivers**: opens the **JDBC Drivers** dialog, which allows you to connect to other data sources

HELP

The **Help** menu consists of the following items:

- **Help**: opens the online help.
- **web site**: opens a browser window to the web site www.dbmaker.com.
- **about**: displays information about the XML transfer template including the build date and number, the CASEMaker technical support e-mail address, and a link to www.dbmaker.com.

The Toolbar

- New File: Menu bar > File > New
- Open File: Menu bar > File > Open
- Save File: Menu bar > File > Save
- Close File: Menu bar > File > Close
- Add new Table/Statement Node: Menu bar > Operations > Insert

- Delete Node: Menu bar > Operations > Remove
- Undo: Menu bar > Operations > Undo {command}
- Redo: Menu bar > Operations > Redo
- Run: Menu bar > Operations > run
- Refresh Database: Menu bar > Operations > Refresh

XTM Object Tree

The XTM Object Tree is a logical representation of the structure of the XTM file. It contains two columns: the XTM column and the xpath column. The XTM column contains a graphical tree representation of objects that have been inserted from the Database Schema Tree. The xpath column contains a path to address the corresponding location in the XML schema tree.

XML Schema Tree

The XML Schema Tree provides a graphical representation of the schema of the XML, DTD, or XSD file from which the XTD will create addresses. Any XML file that conforms to the schema represented in the XML Schema Tree can be used as a data source after the XTM has been created and saves as an XSL.

Database Schema Tree

The Database Schema Tree displays a logical representation of database tables within the selected database.

3.2 Creating an XTM

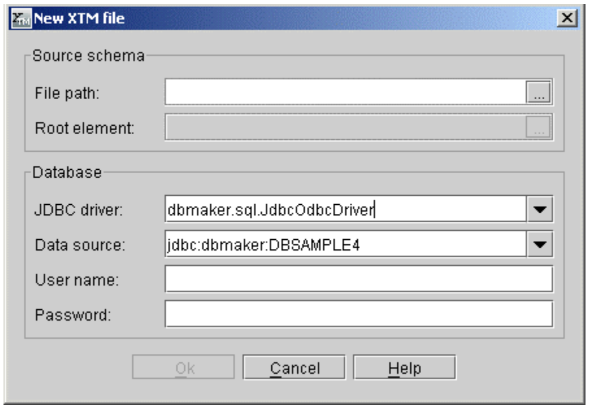
Creating an XTM requires selecting a data source and connecting to a database.

The data source must be an XML, XSD, or DTD file. You must choose one of the elements in the source file to be the root. Only child elements and attributes of the selected element will be visible in the XML schema panel after you create the XTM file. If you want to include other elements or attributes, you will need to create a new XTM file.

Connecting to a database requires a driver and a connection to the database. The standard driver for DBMaker databases is **dbmaker.sql.JdbcOdbcDriver**. For more information about drivers refer to *Adding a New JDBC Driver*. The connection to the database requires that the database is started, that a channel of communications over TCP/IP is open, and that a username and password for an account with privileges on the tables you wish to display is available.

➲ **To create a new XTM:**

1. Select **File** > **New** from the menu bar. The **New XTM file** dialog opens.



File path:

JDBC driver: dbmaker.sql.JdbcOdbcDriver

Data source: jdbc:dbmaker:DBSAMPLE4

User name:

Password:

2. In the **File path** box, type a file path or select the browse button to locate a source schema file (XML, XSD or DTD).

3. To select a root element different from the one indicated in the Root element box, click the browse button to the right. The **Choose root element** dialog appears.
4. Select the element you want to be the root from the tree. Double-click an element in the tree to expand the node and see its child elements.
5. Click **Ok**.
6. In the **Database** box, select a JDBC driver and a data source from the menus.
7. Enter the User name and password of an account on the database.
8. Click **Ok**.. The selected XML schema and database tables will appear in the XML Schema Panel and Database Schema panel, respectively.

➲ **To open an existing XTM:**

1. Select **File** > **Open** from the menu bar.
2. Select the XSL file that you want to open using the file chooser.
3. In the **File path** box, type a file path or select the browse button to locate a source schema file (XML, XSD or DTD).
4. To select a root element different from the one indicated in the Root element box, click the browse button to the right. The **Choose root element** dialog appears.
5. Select the element you want to be the root from the tree. Double-click an element in the tree to expand the node and see its child elements.
6. Click **Ok**.
7. In the **Database** box, select a JDBC driver and a data source from the menus.
8. Enter the User name and password of an account on the database.
9. Click **Ok**.. The selected XML schema and database tables will appear in the XML Schema Panel and Database Schema panel, respectively.

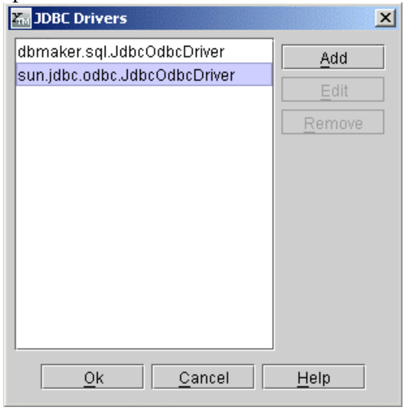
Adding a New JDBC Driver

It may be necessary to use a different JDBC driver than the ones provided. It is possible to connect to databases provided by different vendors as long as the JDBC driver is present. The XTM tool provides this function through the Options menu. It is not necessary to have an XTM open to add a new driver.

You may also edit or remove drivers that you have added. The default drivers cannot be removed or edited.

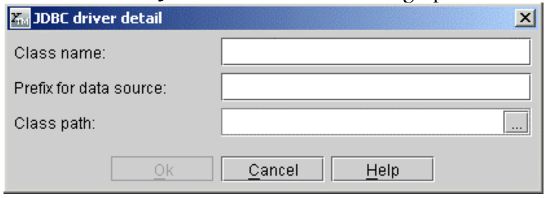
➲ **To add a new JDBC driver**

1. Select **Options** > **JDBC Drivers** from the menu bar. The **JDBC Drivers** window opens.



dbmaker.sql.JdbcOdbcDriver
sun.jdbc.odbc.JdbcOdbcDriver

2. Click **Add**. The **JDBC driver detail** dialog opens.



Class name:
Prefix for data source:
Class path:

3. Type a class name and prefix for data source into the appropriate text boxes.
4. Type a class path or click the browse button to navigate to the location of the driver.
5. Click **Ok**.
6. The new driver will appear in the list of JDBC drivers. Click **Ok**.

3.3 Mapping xpath statements to XTM object nodes

After creating a new XTM, the next step is to create a map between the nodes in the XML schema and the database tables. First you will need to add database tables to the XTM object tree, then you will need to add XML schema paths to properly map from the address in the XML schema where the data is located.

To create the structure of the XTM object tree, drag-and-drop tables from the database into the desired location in the XTM object tree. The first node must be dropped onto the root node; subsequent nodes can be dropped into the root node or any other node that represents a table. Tables cannot be dropped into a node representing a column. When you drop a table, it will appear in the XTM column with all of the columns visible.

After the XTM object tree structure is complete, drag-and-drop nodes from the XML schema panel onto the desired node in the XTM object tree. A statement will appear in the xpath column that corresponds to the xpath representation of the element or attribute address in the XML schema. Furthermore, xpath statements will be preceded by a data property if the are numerical or binary data types. A data property statement will not precede character data types. The xpath data property and corresponding data types are summarized in Table 3-1

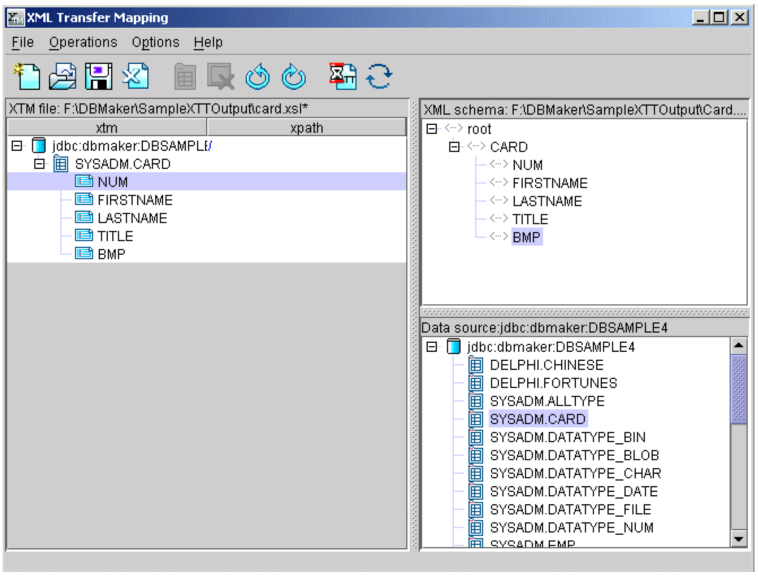
XPATH DATA PROPERTY	SQL DATA TYPE
Character data	Char
	Varchar
	Longvarchar
	Longvarbinary
	File
	Nchar
	Nvarchar

XPATH DATA PROPERTY	SQL DATA TYPE
	Nclob
Numerical data	Serial
	Smallint
	Int
	Float
	Double
	Decimal
Normalize-space (binary) data	Binary
	Date
	Time
	Timestamp

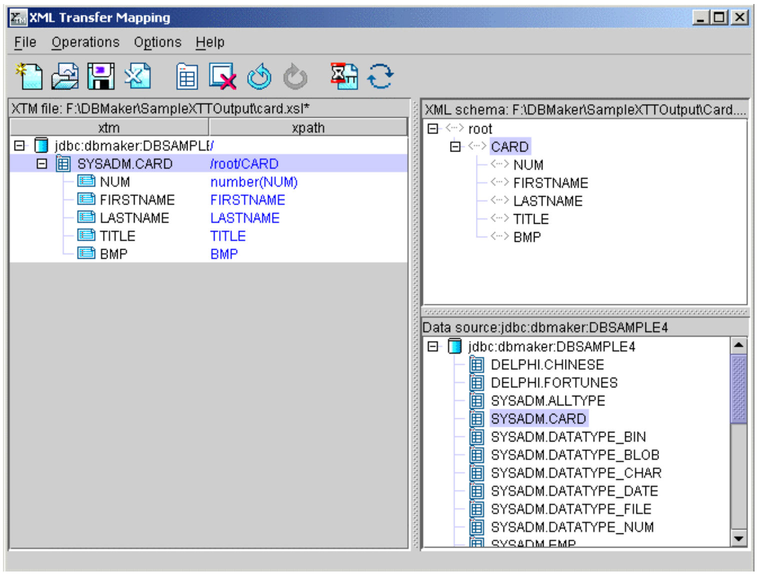
Table 3-1: xpath data properties and corresponding SQL data types

➲ **To build the structure of an XTM:**

1. Create a new XTM or open an existing XTM:
2. Drag-and-drop the table that you want to import data to from the Database schema panel to the xtm column of the XTM editing panel.



3. Drag-and drop the elements or attributes to the XTM nodes that you want to map the data to. The xpath statement appears in the xpath column.



XTM file: F:\DBMaker\SampleXTTOutput\card.xsl*

xtm	xpath
jdbc:dbmaker:DBSAMPLE	/
SYSADM.CARD	/root/CARD
NUM	number(NUM)
FIRSTNAME	FIRSTNAME
LASTNAME	LASTNAME
TITLE	TITLE
BMP	BMP

XML schema: F:\DBMaker\SampleXTTOutput\Card....

- root
  - CARD
    - NUM
    - FIRSTNAME
    - LASTNAME
    - TITLE
    - BMP

Data source:jdbc:dbmaker:DBSAMPLE4

- jdbc:dbmaker:DBSAMPLE4
  - DELPHI.CHINESE
  - DELPHI.FORTUNES
  - SYSADM.ALLTYPE
  - SYSADM.CARD
  - SYSADM.DATATYPE_BIN
  - SYSADM.DATATYPE_BLOB
  - SYSADM.DATATYPE_CHAR
  - SYSADM.DATATYPE_DATE
  - SYSADM.DATATYPE_FILE
  - SYSADM.DATATYPE_NUM
  - SYSADM.EMP

4. When you are finished, click **File** > **Save**.

5. Browse to the folder where you want to store the XTM and type a file name. The file will automatically be saved with the extension .XSL.

3.4 Executing an XTM

After creating the XTM and saving it as an XSL file, you can pass data from the source XML file to the database by executing the XTM. When executing the XTM, you can choose to save the data as an XSL file and run the XTM, or save the transformation as an SQL script. Saving as an SQL script will not enter any data into the database. You must execute the script to enter the data into the database.

If you are trying to automate data transfer using a data transfer API or stored procedure, then you should save the transformation as an XSL file.

Saving an XTM as an SQL Script

Saving an XTM as an SQL script allows you to create an SQL script that will perform the same operation on the database as executing the XTM file, only it stores the equivalent SQL commands in a script file. This method will not actually store any data in the database. Be sure to save the XTM as an XSL file before performing this operation, as it will not save either XML data or the transformation other than in the form of the SQL script.

➲ **To execute an XTM and save output as an SQL script:**

1. Click **Operations** > **Run**. The **Execute XTM** window will appear.
2. Click **Save as SQL Script**.
3. In the **Save as SQL Script** box, enter the full path and file name for the XSL file, or select a file and path by clicking the browse button.
4. In the **Source XML** box, enter a full path and file name for the XML file to import data from, or select a file and path by clicking the browse button.
5. Click **OK**. The XTM Tool will create an SQL script. You can run the SQL script from the dmSQL prompt or using the JDBA tool to enter data into the database.

➲ **Example:**

SQL script output:

```
INSERT INTO DELPHI.CHINESE (ID,TEXT) VALUES (?,?);
```

```
1,'lobdir1\clobfile0.txt';
2,'lobdir1\clobfile1.txt';
3,'lobdir1\clobfile2.txt';
4,'lobdir1\clobfile3.txt';
5,'lobdir1\clobfile4.txt';
…
```

Saving an XTM as an XSL File and Executing

Saving the XTM as an XSL file and executing performs the same operation as the XTM API or stored procedure. Executing the XTM file allows you to find errors in a transformation before you have automated it, and allows you to test the output for a given transformation to ensure that it produces the desired result.

➲ **To execute an XTM and save output as an XSL file:**

1. Click **Operations > Run**. The **Execute XTM** window will appear.
2. Click **Save as XTM and run**.
3. In the **Save as XTM and Run** box, enter the full path and file name for the XSL file, or select a file and path by clicking the browse button.
4. In the **Source XML** box, enter a full path and file name for the XML file to import data from, or select a file and path by clicking the browse button.
5. Click **OK**. The XTM Tool will create an XSL file and add new data to the selected tables in the database.

➲ **Example:**

XSL output:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0"
xmlns:xtm="dbmaker.xml.xtm.XMLTransferMap"
extension-element-prefixes="xtm">
<xsl:template match="/">
<xtm:connect driver="dbmaker.sql.JdbcOdbcDriver"
datasource="jdbc:dbmaker:DBSAMPLE4">
<xtm:table owner="DELPHI" name="CHINESE" select="/root/CHINESE">
<xtm:column name="ID" select="number(@ID)"/>
<xtm:column name="TEXT" select="@TEXT"/>
</xtm:table>
</xtm:connect>
</xsl:template>
```

```
</xsl:stylesheet>
```

4 XTM API Functions

After you have created an XTM file you are ready to automate the transfer process. The XTM API allows you to automate the process. DBMaker provides four XTM Transfer APIs; the XTM API in C++, the XTM API in C, the XTM API in Java, and the XTM API stored procedure.

The XTM API requires a few common objects to process:

- Source XML file – the data source for XTM is kept in an XML file.
- XTM file – the file specifies how to retrieve data from source XML and save into the database.

In addition, the following properties may be requires depending on how you choose to implement the API:

- Database connection – you may optionally pass the JDBC/ODBC connection to the XTM engine . If specified, then it will use this connection and ignore the element 'xtm:connect' in the XTM file. A JDBC connection can be established using a JDBC driver provided by one of several vendors, but an ODBC connection should use the DBMaker ODBC connection handle.
- User name – the user name for database connection. In some cases, you might not want to add the user name to 'xtm:connect'. Alternatively, you may pass the user name when processing XTM file. If specified, then the user name in the XTM file be used instead of the one in 'xtm:connect'.
- Password – the password used to log into the database. Like the user name, you may pass the password to the XTM engine in an API argument.

The search rule for a database connection in XTM processing is:

1. The connection details are passed in the **setConnection** method.
2. Connection with the user/password passed in the **setUser / setPassword** method.
3. Connection information is specified in <xtm:connect> in the XTM file.

If the same property is set more than once before calling the run method, then only the last input value will be used.

4.1 XTM API in C++

The XTM class provides public methods to set the required properties as described earlier, and the execute function.

Public methods:

CONSTRUCTOR AND DESTRUCTOR

Syntax

```
XTM()
```

Constructs an XTM object.

```
~XTM()
```

Destroys an XTM object.

SETCONNECTION

Sets the database connection. This database connection must be a DBMaker ODBC database connection.

Syntax

```
int setConnection( HDBC hdbc );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
hdbc	Input	Database connection handle (cannot be null).

RETURNS	DESCRIPTION
SUCCESS	Set database connection with hdbc will return SUCCESS or the following error code
ERR_XTM_INV_API	Failed to load JVM Failed to find Java class dbmaker.xml.XTM Failed to create a new instance for dbmaker.xml.XTM Failed to find Java class dbmaker.sql.JdbcOdbcConnection Failed to call setConnection
ERR_XTM_PROCESS	Possible reasons for failure: Failed to instantiate dbmaker.sql.JdbcOdbcConnection

SETUSER

Sets the user for a database connection.

Syntax

```
int setUser ( char *user );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
user	Input	user name

RETURNS	DESCRIPTION
SUCCESS	Set user will return SUCCESS or the following error code
ERR_XTM_INV_API	Possible reasons for failure: Failed to load Java virtual machine Failed to find Java class dbmaker.xml.XTM Failed to instantiate dbmaker.xml.XTM Failed to call setUser

SETPASSWORD

Sets the user's password for a database connection.

Syntax

```
int setPassword( char *password);
```

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
password	Input	password

RETURNS	**DESCRIPTION**
SUCCESS	Set password will return SUCCESS or the following error code
ERR_XTM_INV_API	Possible reasons for failure: Failed to load JVM Failed to find Java class dbmaker.xml.XTM Failed to instantiate dbmaker.xml.XTM Failed to call setPassword

GETERROR

If the return code from run(), setConnection(), setUser() or setPassword() is not SUCCESS, you may use this method to get the error message. If no errors are encountered, then this method will return NULL.

Syntax

```
const char * getError();
```

RUN

The method run() will transform the XTM content. The XTM itself is actually an XSL file. The XTM engine is an extension of the Apache Xalan XSLT processor. The transformer will transform the source XML file according to the specification in the XTM file.

Syntax

```
int run ( const char * XtmFileName,
          const char * XmlFileName);
```

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
XtmFileName	Input	XTM filename (cannot be null).
XmlFileName	Input	Input XML filename (cannot be null).

RETURNS	**DESCRIPTION**
SUCCESS	Engine successfully processed source XML file with XTM specification.
ERR_XTM_PROCESS	Possible reasons for XTM process errors include: XSTL process error – XTM or source XML file is not valid XML, or the XTM is not a valid XSL file. XTM syntax error.
SQL Error	The SQL command produced an error.

Examples:

The following code segments show implementations of the XTM API in C++

```
/* an XTM object */
XTM transfer;
const char *error = NULL;
long rc = 0;

/* start transformation */
rc = transfer.run(“c:\\temp\\abc.xsl”,”c:\\temp\\abc.xml”);

/* check if process fails.
   if so, getError to get a detailed error message. */
if( rc != 0 )
{
error = transfer.getError();
printf("%s\n",error);
}
```

or

```
/* an XTM object */
XTM transfer;
const char *error = NULL;
long rc = 0;

/* set database connection,
XTM engine will use this connection,
rather than the xtm:connection specification in XTM file. */
rc = transfer.setConnection(hdbc);

/* start transformation */
rc = transfer.run(“c:\\temp\\abc.xsl”,”c:\\temp\\abc.xml”);

/* check if process fails.
   if so, getError to get the detail error message. */
if( rc != 0 )
{
error = transfer.getError();
printf("%s\n",error);
}
```

4.2 XTM API in C

Summary

The XTM function transfers content from a source XML file into a database.

Syntax

```
int Xtm(                          char *xtmfile,
                          char *xmlfile,
                          char * msgBuffer,
                          int szMsgBuffer)

int XtmUP(                char * user,
                          char * password,
                          char * xtmfile,
                          char * xmlfile,
                          char * msgBuffer,
                          int szMsgBuffer)

int XtmH(                 HDBC hdbc,
                          char * xtmfile,
                          char * xmlfile,
                          char * msgBuffer,
                          int szMsgBuffer)
```

All three functions will process an XTM, retrieve data from a source XML file and update the database. All three functions need to specify the XTM and source XML file names in the filename arguments. The function XtmUP needs to specify a user name and password for database connection. The function XtmH needs to specify a database connection.

ARGUMENTS (FOR XTM)

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
XtmFileName	Input	XTM filename. It must be a valid file path.
XmlFileName	Input	Source XML filename (cannot be null).
msgBuffer	Output	Pointer to a buffer for any error message. The output result will be null terminated.
szMsgBuffer	Input	Size of msgBuffer.

ARGUMENTS (FOR XTMUP)

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
user	Input	User name: used to establish a database connection.
password	Input	Password: used to establish a database connection.
XtmFileName	Input	XTM filename. It must be a valid file path.
XmlFileName	Input	Source XML filename (cannot be null).
msgBuffer	Output	Pointer to a buffer for any error message. The output result will be null terminated.
szMsgBuffer	Input	Size of msgBuffer.

ARGUMENTS (FOR XTMH)

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
hdbc	Input	Database connection handle, cannot be null.
XtmFileName	Input	XTM filename: must be a valid file path.

XmlFileName	Input	Source XML filename, cannot be null.
msgBuffer	Output	Pointer to a buffer for any error message. The output result will be null terminated.
szMsgBuffer	Input	Size of msgBuffer.

RETURNS	**DESCRIPTION**
SUCCESS	Engine successfully processed source XML file with XTM specification.
ERR_XTM_PROCESS	Possible reasons for XTM process errors include: XSTL process error – XTM or source XML file is not valid XML, or the XTM is not a valid XSL file. XTM syntax error.
SQL Error	The SQL command produced an error.

Examples:

```
/* pass user name and password to XTM API. */
rc = XtmUP( user, password, xtm, xml, message, sizeof(message));

/* check if process fails.
   if so,message should contain detailed error information. */
if( rc != 0 )
{
  if( message != NULL )
{
    printf("%s\n\n",message);
  }
}
```

or

```
/* pass hdbc (odbc database connection handle ) to XTM API */
rc = XtmH(hdbc, xtm, xml,  message, sizeof(message));

/* check if process fails.
   if so,message should contain detailed error information. */
if( rc != 0 )
{
  if( message != NULL )
{
    printf("%s\n\n",message);
  }
}
```

4.3 XTM API in Java

The XTM processor also provides the Java class dbmaker.xml.XTM.

Public methods:

CONSTRUCTOR :

Constructs a XTM object.

Syntax
```
XTM( )
```

SETCONNECTION

Sets the database connection. This functions accepts the java.sql.Connection object as an argument. The connection can be established via the DBMaker JDBC driver or a JDBC driver from other vender.

Syntax
```
void setConnection( Connection conn );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
conn	Input	Object of Class java.sql.Connection (cannot be null).

SETUSER

Sets the user for a database connection.

Syntax

```
void setUser ( String user);
```

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
user	Input	User name

SETPASSWORD

Sets the user's password for a database connection.

Syntax

```
void setPassword ( String password );
```

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
password	Input	Password used for database connection

SETPARAMETER

Set parameter for XSL processor. Parameter value will be available during XTM processing. It can be referenced in the syntax as ***$parameter_name***.

Syntax

```
void setParameters ( String name, Object value );
```

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
name	Input	Parameter name. Parameter name is a qualified name as a two-part string, the namespace URI enclosed in curly braces ({}), followed by the local name. If the name has a null URL, the String only contain the local name. An application can safely check for a non-null URI by testing to see if the first character of the name is a '{' character. For example, if a URI and local name were obtained from an element defined with &lt;xyz:foo xmlns:xyz="http://xyz.foo.com/yada/baz.html"/&gt;, then the qualified name would be "{http://xyz.foo.com/yada/baz.html}foo". Note that no prefix is used.
value	Input	The parameter value.

GETERROR

If return code from run() is not SUCCESS, then use this method to get the error message. If no error, then this method will return NULL.

Syntax

```
String getError();
```

RUN

The method run() will transform the XTM content. The XTM itself is actually an XSL file. The XTM engine is an extension of the Apache Xalan XSLT processor. The transformer will transform the source XML file according to the specification in the XTM file.

Syntax

```
int run ( String xtmfile,
          String xmlfile);
```

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
xtmfile	Input	XTM filename. It cannot be null.
xmlfile	Input	Source XML filename. It cannot be null.

RETURNS	**DESCRIPTION**
SUCCESS	Engine successfully processed source XML file with XTM specification.

RETURNS	DESCRIPTION
ERR_XTM_PROCESS	Possible reasons for XTM process errors include: XSTL process error – XTM or source XML file is not valid XML, or the XTM is not a valid XSL file. XTM syntax error.
SQL Error	The SQL command produced an error.

Example:

```
/* create new XTM object */
XTM transfer = new XTM();

/* start transformation */
int rc = transfer.run(xtm, xml);

/* check if process fails.
   if so, getError for error detail. */
if ( rc != 0 )
{
  System.out.println(transfer.getError()+"\n");
}
```

4.4 XTM Stored Procedure

The XTM stored procedure provides an XTM transformation function similar to the C++, C, and Java APIs. The main differences are

- It only accepts XTM and source XML filenames.
- It only uses a current database connection.
- The filenames specified will be on the server site.

Stored procedure definition:

Syntax

```
CREATE PROCEDURE XTM(
  VARCHAR(257)  XTMFILE     INPUT,
  VARCHAR(257)  XMLFILE     INPUT );
```

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
XTMFILE	Input	XTM filename
XMLFILE	Input	Source XML filename

Privilege

The owner of this stored procedure is SYSTEM. The privileges of the stored procedure are the same as the user executing the stored procedure. For example, if there is an insert statement in an XTM file to a table on which the user does not have the privilege to insert, then an SQL error will be returned.

Example

```
dmSQL> call XTM('c:\temp\case1.xsl ','c:\temp\case1.xml');
```

5 XTT API Functions

The XTT API is provided by DBMaker for automating the output of XML data from the database to files or a location in memory. APIs are provided in C, C++, Java, and also in the form of a stored procedure. A few common properties are required for XTT to process:

- Database connection - either with a connection string or pre-connected handle.
- XTT - either the XTT filename or XTT content in memory buffer.
- Output XML - either the XML filename or a pre-allocated buffer for output XML content.
- Parameters - a string with the parameter values for the template file. The string might contain several pairs of parameter names and values. Each pair is separated by comma. The text on the left of the equal sign is the name of parameter, on the right is the value. The values may be enclosed by doubles quote or single quotes. For example, *"student_id='10012',class_id='103'"*.

5.1 XTT API in C++

The XTT class provides public methods to set the required properties as described in previous section and the execute function to run.

Public methods:

CONSTRUCTOR AND DESTRUCTOR

Syntax

```
XTT()
```

Constructs an XTT object.

```
~XTT()
```

Destroys an XTT object.

SETDATABASECONNECTION

Sets a database connection. If there is a previous connection established by an earlier call with a connection string, it will be disconnected first. Otherwise, it will be disconnected when the object is destroyed. The data source will be automatically disconnected, if the XTT has been properly destroyed. The connection can be forced to disconnect by calling *setDatabaseConnection((HDBC)NULL)*.

Syntax

```
int setDatabaseConnection( const HDBC hdbc );
```

Sets the database connection with a pre-connected odbc connection handle.

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
Hdbc	Input	Database connection handle (cannot be null).

```
int setDatabaseConnection( const char * connectionStr );
```

Sets the database connection with a connection string, and will try to connect to the database and keep the connection handle.

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
connectionStr	Input	Connection string that can be used to connect to data source (cannot be null).

RETURNS	DESCRIPTION
SUCCESS	Setting the database connection with hdbc will always return SUCCESS. If Setting a database connection with a connection string, it will return SUCCESS if it has connected successfully.
SQLConnect Error	If an error occurs when setting the database connection with a connection string, it will return this error message.

SETXTT

Sets the required input XTT before run().

Syntax

```
void setXtt( const char *xttFileName );
```

Sets the XTT filename.

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
xttFileName	Input	XTT filename

```
void setXtt( const char * xttValuePtr,
             int szXttValue );
```

Sets the XTT content.

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
xttValuePtr	Input	Pointer to a buffer with XTT content.
SzXttValue	Input	Length of xttValuePtr

SETOUTPUTXML

Sets the output XML.

Syntax

```
void setOutputXml( const char *xmlFileName );
```

Sets the output XML filename.

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
xmlFileName	Input	Output XML filename (cannot be null). The result XML will be written into the file.

```
void setOutputXml( char * xttValuePtr,
                   int szXttValue );
```

Sets a memory buffer for output content. The output buffer will be null terminated.

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
xmlValuePtr	Output	The pointer to a buffer for output XML content. The output result will be null terminated.
szXmlValue	Input	Size of xmlValuePtr buffer.

SETPARAMETERS

Sets parameter values for the template file. For example, if there are two parameters defined in XTT file:

```
<xtt:template xmlns:xtt="urn:schema-dbmaker-com:xml-template" encoding="UTF-8">
<xtt:parameter name="student_id" default="10001"/>
<xtt:parameter name="class_id" default="201"/>
...
</xtt:template>
```

The setParameters method allows you to specify the value for each parameter. The argument is a string, which may contain several pairs of parameter names and values. Each pair is separated by a comma. The text on the left of the equal sign is the name of the parameter, on the right is the value. For example, to set parameters for the XTT file in the example above:

```
setParameters("student_id='10012',class_id='103'").
```

Syntax

```
void setParameters ( const char *parameters );
```

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
parameters	Input	The parameters for the template file, in the format : *[param_name="value" [,param_name="value"]...]* Note: the value can be enclosed in single quotes or double quotes.

SETGENXMLHEADER

Set this parameter if generating the XML header <?xml...?> and printing to output xml. The default value is true. If set to false, the header will not be printed.

Syntax

```
void setGenXMLHeader( bool stat );
```

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
stat	Input	Boolean status if outputting xml with a header.

GETERROR

If the return code from run() or setDatabaseConnection() is not SUCCESS, use this method to get the error message. If there is no error, then this method will return NULL.

Syntax

```
const char * getError();
```

RUN

The method run() will load the XTT content, either retrieved from the XTT filename specified or from memory, into an XTT object model that can be processed internally. The instructions specified in XTT are processed and output printed into either the specified output XML file or output memory. If executed successfully, the return code will be 0. If an error occurs, use the getError() method to get the error message.

The second and third method call the corresponding setter methods to set the properties and then call run(). The XTT object model is cached until setXtt() is called again or the XTT object is destroyed.

Syntax

```
int run ( );
int run ( const char * connectionStr,
          const char *xttFileName,
          const char *xmlFileName );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
connectionStr	Input	Connection string that can be used to connect to a data source (cannot be null).
xttFileName	Input	XTT filename
xmlFileName	Input	Output XML filename. It cannot be null. The resulting XML will be written into the file.

```
int run ( const char * connectionStr,
          const char *xttValuePtr,
          int szXttValue,
          char *xmlValuePtr,
          int szXmlValue);
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
connectionStr	Input	Connection string that can be used to connect to a data source (cannot be null).

Argument	Input/Output	Description
xttValuePtr	Input	Pointer to a buffer with XTT content.
szXttValue	Input	Length of xttValuePtr
xmlValuePtr	Output	A pointer to a buffer for the output XML content. The output result will be null terminated.
szXmlValue	Input	Size of xmlValuePtr buffer.

Returns	Description
SUCCESS	Transformation successful.
ERR_XTT_INV_ARG	Missing required arguments/properties for XTT engine to process. For example, no XTT filename or XTT content is set.
ERR_XTT_XERCES_PARSER	Error returned by apache xerces parser. XTT is not in valid XML format.
ERR_XTT_INV_SYNTAX	Invalid XTT syntax. Possible causes: No recognized xtt element or attribute. Not exactly one user-defined element in xtt:template element. <xtt:attribute> is declared after any user-defined element in its parent. <xtt:parameter> is declared after a user-defined element under <xtt:template> Missing required attribute.
ERR_XTT_PROCESS	XTT process errors: Failed to create folder or file. Invalid variable references.
SQL Error	An SQL error occurred during processing.

Examples:

```
/* an object of XTT */
XTT transfer;

/* set database connection pre-connected odbc connection */
transfer.setDatabaseConnection(hdbc);

/* set input XTT filename */
transfer.setXtt("c:\\temp\\abc.xtt");

/* set output XML filename */
transfer.setOutputXml("c:\\temp\\abc.xml");

/* start transformation */
int rc = transfer.run();

/* check if process fails.
   if so, getError to get the detail error message. */
if ( rc != 0 )
{
  printf("%d %s\n", rc, transfer.getError());
}
```

or

```
/* an object of XTT */
XTT transfer;

/* transfer with specified arguments
- database connection string
- input XTT filename
- output XML filename
*/
int rc = transfer.run( "DSN=DBSAMPLE;UID=SYSADM;PWD=x123",
"c:\\temp\\abc.xtt",
"c:\\temp\\abc.xml");
```

5.2 XTT API in C

XTT transfers database data according to the definition in the XTT file.

Syntax

```
int Xtt(                          const HDBC hdbc,
                          const char *xttFileName,
                          const char *parameters,
                          const char *xmlFileName,
                          char * msgBuffer,
                          int szMsgBuffer )

int XttMem (              const HDBC hdbc,
                          const char *xttValuePtr,
                          int szXttValue,
                          const char *parameters,
                          char *xmlValuePtr,
                          int szXmlValue,
                          unsigned char * msgBuffer,
                          int szMsgBuffer )
```

Both functions will process the XTT, retrieve data from the database and write it to XML output. The function Xtt has the XTT source and XML destination filenames, while the function XttMem has the source content and destination buffer in memory.

Arguments (for Xtt)

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
hdbc	Input	Database connection handle (cannot be null).
xttFileName	Input	XTT filename: must be a valid file path.
parameters	Input	The parameters for the template file in the format of [param_name="value" [,param_name="value"]...]
xmlFileName	Input	Output XML filename. It cannot be null. The resulting XML will be written into the file.

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
msgBuffer	Output	Pointer to a buffer for an error message. The output result will be null terminated.
szMsgBuffer	Input	Size of msgBuffer.

Arguments (for XttMem)

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
hdbc	Input	The database connection handle, cannot be null.
xttValuePtr	Input	Pointer to a buffer with XTT content.
szXttValue	Input	Length of xttValuePtr
parameters	Input	The parameters for the template file in the format [param_name="value" [,param_name="value"]...]
outputXmlPtr	Output	The pointer to a buffer for output XML content. The output result will be null terminated.
szOutputXml	Input	Size of outputXmlPtr buffer.
msgBuffer	Output	Pointer to a buffer for the error message. The output result will be null terminated.
szMsgBuffer	Input	Length of msgBuffer.

RETURNS	DESCRIPTION
SUCCESS	Transformation successful.
ERR_XTT_INV_ARG	Missing required arguments/properties for XTT engine to process. For example, no XTT filename or XTT content is set.
ERR_XTT_XERCES_PARSER	Error returned by apache xerces parser. XTT is not in valid XML format.

RETURNS	DESCRIPTION
ERR_XTT_INV_SYNTAX	Invalid XTT syntax. Possible causes: No recognized xtt element or attribute. Not exactly one user-defined element in xtt:template element. <xtt:attribute> is declared after any user-defined element in its parent. <xtt:parameter> is declared after a user-defined element under <xtt:template> Missing required attribute.
ERR_XTT_PROCESS	XTT process errors: Failed to create folder or file. Invalid variable references.
SQL Error	An SQL error occurred during processing.

5.3 XTT API in Java

Public methods:

CONSTRUCTOR AND DESTRUCTOR:

```
XTT( )
```

Constructs a XTT object.

```
finalize ( )
```

Before destroying a XTT object.

SETDATABASECONNECTION

This function sets the database connection. Any connection established by an earlier call using the connection string method will be disconnected first. The connection can be forced to disconnect by the call *setDatabaseConnection((String)NULL)*.

Syntax

```
int setDatabaseConnection( JdbcOdbcConnection conn );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
conn	Input	DBMaker jdbc connection object (cannot be null).

```
int setDatabaseConnection( String connStr );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
connStr	Input	Connection string that can be used to connect to data source (cannot be null).

RETURNS	
SUCCESS	Setting the database connection with hdbc will always return SUCCESS. If setting the database connection with a connection string, it will try to connect using the connection specified, and will return SUCCESS if successful.
SQLConnect Error	The function will return this string if setting the database connection with a connection string fails.

SETXTT

Sets the XTT filename or content.

Syntax

```
void setXtt( String xttFileName );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
xttFileName	Input	XTT filename

```
void setXtt( byte [] xttValue );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
xttValue	Input	Byte array with XTT content.

SETOUTPUTXML

Sets the output XML filename or a memory buffer for output content.

Syntax

```
void setOutputXml( String xmlFileName );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
xmlFileName	Input	Output XML filename

```
void setOutputXml( byte [] xmlValue );
```

The output buffer is null terminated.

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
xmlValue	Input	Byte array with output XML content.

SETPARAMETERS

Sets parameter values for the template file.

Syntax

```
void setParameters ( String parameters );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
parameters	Input	The parameters for the template file, in the format of: [param_name="value" [,param_name="value"]...]

GETERROR

If the return code from run() is not SUCCESS, then use this method to get the error message. If no error is returned, then this method will return NULL.

Syntax

```
String getError();
```

RUN

Starts the XML transformation.

Syntax

```
int run ( );
int run (  String connectionStr,
           String xttFileName,
           String outputFileName );
```

ARGUMENT	INPUT/OUTPUT	DESCRIPTION
connectionStr	Input	Connection string that can be used to connect to a data source (cannot be null).
xttFileName	Input	XTT filename
xmlFileName	Input	Output XML filename (cannot be null). The resulting XML will be written into the file.

```
int run (  String connectionStr,
          byte [] xttValuePtr,
          byte [] xmlValuePtr);
```

Argument	**Input/Output**	**Description**
connectionStr	Input	Connection string that can be used to connect to the data source (cannot be null).
xttValue	Input	Byte array with XTT content.
xmlValue	Output	Byte array for output XML content. The output result will be null terminated.

Returns	**Description**
SUCCESS	Transformation successful.
ERR_XTT_INV_ARG	Missing required arguments/properties for XTT engine to process. For example, no XTT filename or XTT content is set.
ERR_XTT_XERCES_PARSER	Error returned by apache xerces parser. XTT is not in valid XML format.
ERR_XTT_INV_SYNTAX	Invalid XTT syntax. Possible causes: No recognized xtt element or attribute. Not exactly one user-defined element in xtt:template element. <xtt:attribute> is declared after any user-defined element in its parent. <xtt:parameter> is declared after a user-defined element under <xtt:template> Missing required attribute.
ERR_XTT_PROCESS	XTT process errors: Failed to create folder or file. Invalid variable references.
SQL Error	An SQL error occurred during processing.

Example:

```
    String xtt = "c:\\temp\\case1.xtt";
    String xml = "c:\\temp\\case1.xml";
    String error = "c:\\temp\\case1.log";
    String param = null;
    String dbname = "DBSAMPLE";
    String user = "SYSADM";
    String password = "";
    try
    {
      /* connect to dbmaker database via dbmaker jdbc bridge */
      Class.forName("dbmaker.sql.JdbcOdbcDriver");
      System.setProperty("DM_DRIVER_MODE", "CLIENT_SERVER");
      System.setProperty("DM_CONNECT_MODE", "CONNECT_DB");
      Connection conn = DriverManager.getConnection("jdbc:dbmaker:"+dbname,
user, password);
      /* new XTT object */
      XTT transfer = new XTT();

      /* set database connection with previous connected Jdbc connection. */
          transfer.setDatabaseConnection((dbmaker.sql.JdbcOdbcConnection)conn);

      /* set XTT filename */
          transfer.setXtt(xtt);

      /* set output XML filename */
          transfer.setOutputXml(xml);

      /* set parameter values for XTT template file */
          transfer.setParameters(param);

      /* start XML transformation */
      int rc = transfer.run();

          if( rc != 0 )
      {
        /* print out error message */
            System.out.println(transfer.getError());
          }
      conn.close();
    }
    catch( ClassNotFoundException e )
    {
      e.printStackTrace();
    }
    catch( SQLException sqle )
    {
      sqle.printStackTrace();
    }
```

5.4 XTT Stored Procedure

The XTT stored procedure provides XML transformation functionality similar to the APIs in C, C++, and Java. The major differences are:

- It only accepts XTT and output XML filenames.
- The filenames specified are on the server site.

Stored procedure definition:

```
CREATE PROCEDURE XTT(
  VARCHAR(257)  XTTFILE              INPUT,
  VARCHAR(257)  OUTPUTFILE           INPUT,
  VARCHAR(257)  PARAMETERS    INPUT );
```

ARGUMENT	**INPUT/OUTPUT**	**DESCRIPTION**
XTTFILE	Input	XTT filename
OUTPUTFILE	Input	Output XML filename
PARAMETERS	Input	Parameter values for XTT template.

Privilege

The owner of this stored procedure is SYSTEM. The privileges of the stored procedure are the same as the user executing the stored procedure. For example, if there is a query in an XTM file to a table on which the currently logged on user does not have the privilege to select, then an SQL error will be returned.

Examples

```
dmSQL> call XTT('d:\document\test\xtt\case1.xml','c:\temp\case1.xml',NULL);

dmSQL> call XTT('d:\document\test\xtt\case1.xml',NULL,NULL);
ERROR (5612): [DBMaker] xtt invalid argument :
   xml filename or xml destination buffer is required.
```