HP Select Audit Software

for the Windows®, HP-UX®, Linux®, and Solaris® operating systems

Software Version: 1.02

The Configurable Normalizer Guide

Document Release Date: July 2007
Software Release Date: July 2007

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

HP provides third-party products, software, and services that are not HP Branded “AS IS” without warranties or representations of any kind from HP, although the original manufacturers or third party suppliers of such products, software and services may provide their own warranties, representations or conditions. By using this software you accept the terms and conditions.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

HP Select Audit includes software developed by third parties. The software HP Select Audit uses includes:

- ANTLR Copyright 2005 Terrence Parr.
- commons-logging from the Apache Software Foundation.
- Install Anywhere, Copyright 2004 Zero G Software, Inc.
- Jasper Decisions Copyright 2000-2006 JasperSoft Corporation.
- JavaScript Tree, Copyright 2002-2003 Geir Landro.
- Legion of the Bouncy Castle developed by Bouncy Castle.
- log4J from the Apache Software Foundation.
- Microsoft SQL Server 2005 JDBC Driver
- OpenAdaptor from the Software Conservancy.
- Oracle JDBC Thin Driver
- Quartz, Copyright 2004 - 2005 OpenSymphony
- spring-framework from the Apache Software Foundation.
- Tomahawk from the Apache Software Foundation.
- treeviewjavascript from GubuSoft.
- Xalan-Java from the Apache Software Foundation.
- Xerces-Java version from the Apache Software Foundation.

Please check the `<install_dir>/3rd_party_license` folder for expanded copyright notices from such third party suppliers.

Documentation Updates

This guide's title page contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

http://ovweb.external.hp.com/lpe/doc_serv/

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP software support web site at:

www.hp.com/go/hpsoftwaresupport

HP Software online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels and HP Passport, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

To find more information about HP Passport, go to:

http://h20229.www2.hp.com/passport-registration.html

Contents

1 Using the Configurable Normalizer

Select Audit uses a Normalizer that runs periodically as a timer thread on the J2EE server and transforms the XML elements and attributes of audit messages into columns of relational database tables. In the non-configurable normalizer, the transformation is coded in Java. In the configurable normalizer, the transformation is coded in XML and XSL as an XSLT template.

This document describes how to configure the Configurable Normalizer.

Assumptions in This Document

This document assumes you have knowledge of the following:

- XML
- XSLT
- XPATH
- SQL
- Oracle and MSSQL databases
- Java
- Log4j

About the Configurable Normalizer

The Configurable Normalizer is a process running on the Audit Server. The Audit Connector sends messages to the Audit Server in XML format. The Audit Server receives and stores the messages in the BATCH table in the Audit database. The Audit Server then normalizes the stored messages using the Configurable Normalizer.

The Configurable Normalizer performs the following tasks to normalize the messages:

- It reads messages from the BATCH table.
- It evaluates an XPath expression in an attempt to recognize and identify log messages.
- It transforms the log messages into XML documents that represent database insert operations.
- It insert the data into output tables in the Audit database, in a format suitable for analysis and reporting.
- It adds a digital signature to the output data.

Configuring the Configurable Normalizer

You must perform the following steps to configure the Configurable Normalizer:

1. Create the output tables and sequence generator objects for primary keys in the Audit database.
2. Create an XML document that declares the output tables.
3. Create an XPath expression for selecting which XSL stylesheet should be applied to a log message. The Configurable Normalizer can be configured to perform different XSL transformations for different log messages.
4. Create an XSLT template that transforms XML documents with log messages into XML documents that represent database Insert operations.
5. Load the configuration data into a row in the APPLICATION table in the Audit database. Each XSL stylesheet is configured in a separate row of APPLICATION table.
6. Populate the COMPONENT and COMPONENTEVENTTYPES tables (optional).

   If you do not perform this step, the value for these tables is set to -1 when the tables are used in reports or by the model.

Output Tables

The output tables are where the Configurable Normalizer inserts normalized log data into the Audit database. The AUDITEVENT table, which is created by Select Audit, is the default output table. You can add columns to the AUDITEVENT table or create new output tables. The Configurable Normalizer does not create any tables in the Audit database you must declared the output tables in an XML document.

You must create the output tables before normalizing log messages.

The output tables are structured in a tree. The AUDITEVENT table is always at the top of the tree. Child tables can have only one parent table and every table below the AUDITEVENT table must have a primary key column with a key that points to its parent table.

Sequence Generators

Each output table must have a sequence generator for generating the values of the primary keys. For Oracle, create a SEQUENCE object named by appending "_SEQ" to name of the table. For MSSQL, create a stored procedure and another table with an IDENTITY column. The primary key is generated by inserting to that table.

Oracle Sequence Generator Example

```
DROP SEQUENCE BOPAUDITEVENT_SEQ;
CREATE SEQUENCE BOPAUDITEVENT_SEQ
    MINVALUE 1
    MAXVALUE 999999999999999999999999999
    START WITH 1
    INCREMENT BY 1
    CACHE 20;
```

Types of Output Tables

There are two types of output tables:

- Fact tables.
- **Lookup** tables that are used for the output of data and for the translation of column values inserted into the Fact tables. The Lookup tables are populated during the normalization of log messages.

Fact Tables

Fact tables are only used for the output of data. They are structured as a tree and the AUDITEVENT table is always at the top of the tree. Lower-level child tables can have only one parent table.

Each Fact table must have the following:

- A column with a primary key.
- A column with a primary key of parent table.
- A SIGNATUREID column for the digital signature.

Fact Table Example

```
drop table BOPAUDITEVENT CASCADE CONSTRAINTS;
create table BOPAUDITEVENT (
      ID INTEGER not null,
      AUDITEVENTID INTEGER not null,
      SIGNATUREID INTEGER,
      BOPLOGTYPE_ID INTEGER,
      BOPEVENTTYPE_ID INTEGER,
      BOP_LOGID VARCHAR2(128),
      TIME_STAMP VARCHAR2(32),
      USER_ID VARCHAR2(32),
      USER_ID_TYPE VARCHAR2(16),
      USER_ACTIVEASSIGNMENT VARCHAR2(32),
      USER_ORGANIZATIONALUNIT VARCHAR2(32),
      AUTH_METHOD VARCHAR2(64),
      AUTH_USERNAME VARCHAR2(32),
      AUTH_PKI_ISSUER VARCHAR2(64),
      AUTH_PKI_SERIAL VARCHAR2(64),
      AUTH_SMS VARCHAR2(64),
      constraint BOPAUDITEVENT_PK primary key (ID)
);
create index BOPAUDITEVENT_IDX on BOPAUDITEVENT (AUDITEVENTID ASC);
alter table BOPAUDITEVENT
      add constraint BOPAUDITEVENT_FK1 foreign key (AUDITEVENTID)
      references AUDITEVENT (ID);
```

Lookup Tables

Lookup tables are used for the output of data and for the translation of column values inserted into the Fact tables. The Lookup tables are populated during the normalization of log messages.

Lookup is performed in the following way:

- The value about to be inserted into the output table selects a row of Lookup table.
- The column value from the selected row is inserted into the output table.
- If no row selected, a new row is inserted into the Lookup table.

Lookup tables do not have a digital signature but the result of the lookup is signed.

Each table must have the following:

- A primary key column.
- A unique constraint set for columns that are used as lookup keys. Without the unique constraint, the Lookup table is not populated correctly.

New rows are inserted into Lookup tables in a similar way as for Fact tables except that rows that violate the unique constraint will not be inserted.

Lookup Table Example

```
drop table BOPEVENTTYPES CASCADE CONSTRAINTS;
create table BOPEVENTTYPES (
      ID INTEGER not null,
      EVENT_TYPE VARCHAR2(128),
      constraint BOPEVENTTYPES_PK primary key (ID)
);
create unique index BOPEVENTTYPES_IDX on BOPEVENTTYPES (EVENT_TYPE ASC);
```

Declaring Output Tables

You must create an XML document that declares the Fact tables and Lookup tables. This XML document must be loaded into the APPLICATION table.

The AUDITEVENT table must always be at the root of the tree. The AUDITEVENT table has permanent default columns that cannot be changed and do not need to be declared in the table declaration document. You can add new columns to the AUDITEVENT table by declaring them in the table declaration document.

Table Declaration XML Example

```
<tables>
    <table name="BOPLOGTYPES" type="lookup">
        <column name="ID" type="NUMERIC" key="primary"/>
        <column name="LOG_TYPE" type="VARCHAR" lookupValue="ID"/>
    </table>
    <table name="AUDITEVENT" type="fact">
        <table name="BOPAUDITEVENT" type="fact">
            <column name="ID" type="NUMERIC" key="primary"/>
            <column name="AUDITEVENTID" type="NUMERIC" key="parent"/>
```

```
            <column name="SIGNATUREID" type="NUMERIC"/>
            <column name="BOPLOGTYPE_ID" type="NUMERIC"
                    lookupTable="BOPLOGTYPES" lookupKey="LOG_TYPE"/>
            <table name="BOPREQUSTEDRESOURCE" type="fact">
                <column name="ID" type="NUMERIC" key="primary"/>
                <column name="BOPAUDITEVENT_ID" type="NUMERIC"
                 key="parent"/>
                <column name="SIGNATUREID" type="NUMERIC"/>
           </table>
        </table>
    </table>
</tables>
```

The <table> Element

The `<table>` element represents a table. The tree of tables is represented by nesting `<table>` elements. The table type is configured by the `type` attribute which can have one of two values; `fact` or `lookup`. The `<table>` element must have a `name` attribute.

The <column> Element

The `<column>` element represents a column of a table. It must have a `name` attribute. The column type is configured by the `type` attribute which can have one of the following values:

- `NUMERIC`
- `TIMESTAMP`
- `VARCHAR`
- `CLOB`

A `NUMERIC` column stores a number which is inserted to table as a Java long value.

A `TIMESTAMP` column stores a date/time value which is inserted to table as a Java String value and is formatted using Java's `java.text.SimpleDateFormat` class with the following pattern:

```
"yyyy-MM-dd HH:mm:ss.SSS"
```

A `VARCHAR` column stores a text value which is inserted as a Java String value with a size limit which is determined by the database. Values longer than the size limit are truncated.

A `CLOB` column stores a text value which is inserted as a Java String value but without any size limit.

The `key` attribute of a column can have two values:

- `parent`
- `primary`

If value is `primary`, the column's value is the table's primary key.

If value is `parent`, the column's value is a value of the primary key of the parent table.

Lookups

If a column value is determined by a lookup to another table, that column must have the `lookupTable` attribute, which configures the name of a Lookup table.

The lookup is performed by using a value about to be inserted into an output table as a key to select a row of a Lookup table. A column value from the selected row of the Lookup table is inserted into the output table, instead of the original value. If the key does not select any row of the Lookup table, a new row is inserted into the Lookup table, with log data extracted from a log message being normalized.

The `lookupKey` and `lookupValue` attributes configure names of columns of a Lookup table. The `lookupKey` column is used for selecting a row of a Lookup table, indexed by a value from a log message. The value of `lookupValue` column of the selected row is inserted in an output table.

The `lookupKey` column must have a unique constraint set when a Lookup table is created in the database. Without the unique constraint, the Lookup table will not be populated correctly. New rows are inserted into Lookup tables in a similar way to Fact tables except that rows that violate the unique constraint will not be inserted.

If a `lookup` column of a Fact table does not have the `lookupValue` attribute, then a `lookupValue` attribute of a `lookupKey` column of the Lookup table takes effect. If a `lookupKey` column of the Lookup table does not have the `lookupValue` attribute, then the primary key of the Lookup table is used as a lookup value.

The XSLT Template

The XSLT template is an XML document that transforms audit message XML into an XML document that represents database inserts. The XSLT template should follow these guidelines:

- It should not insert the default columns of the AUDITEVENT table.
- It should not insert the primary key columns.
- It should not insert the parent key and SIGNATUREID columns of the Fact tables.
- It must be loaded into the APPLICATION table.

XSLT Example

```
<xsl:stylesheet version="2.0" xmlns:xsl=http://www.w3.org/1999/XSL/
Transform
 .. Namespace declarations…
  <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
    <xsl:template match="*">
     <AUDITEVENT>
     <xsl:for-each select="//LogMessage">
       <BOPAUDITEVENT>
       <xsl:for-each select="//Head">
         <xsl:attribute name="BOP_LOGID"><xsl:value-of select="LogId"/>
         </xsl:attribute>
        </xsl:for-each>
       <xsl:for-each select="//User">
         <xsl:attribute name="USER_ID"><xsl:value-of select="UserId"/>
         </xsl:attribute>
         <xsl:attribute name="USER_ID_TYPE">
           <xsl:value-of select="UserId/@oid"/>
```

```
          </xsl:attribute>
        </xsl:for-each>
        <xsl:for-each select="//Head">
          <BOPLOGTYPES>
          <LOG_TYPE>
            <xsl:value-of select="LogType"/>
          </LOG_TYPE>
          </BOPLOGTYPES>
        </xsl:for-each>
        </BOPAUDITEVENT>
      </xsl:for-each>
      </AUDITEVENT>
    </xsl:template>
</xsl:stylesheet>
```

Transforming the Log Message

The XSLT template transforms XML documents with log messages into XML documents that represent database Insert operations.

Input XML

An input XML document with log messages always has an <hpsa:Batch> element as the root element and can have many nested <hpsa:Message> elements. These two elements are generated by the Audit Connector. All XML elements nested under the <hpsa:Message> elements contain data logged by the client application.

Input XML Example

```
<hpsa:Batch hpsa:ConnectorID="unittests.can.hp.com" xmlns:hpsa="http://
www.hp.com/ov/selectaudit">
    <hpsa:Message hpsa:ClientID="AuditClientAppender"
     hpsa:ClientTime="2006-08-10T21:56:36.863Z"
                 xmlns:hpsa="http://www.hp.com/ov/selectaudit">
        <LogMessage xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                 xsi:noNamespaceSchemaLocation=
                 "file:///c:/logmessageversion10.xsd">
            <Head>
               <LogId oid="6666">555555</LogId>
            </Head>
            <User>
               <UserId oid="oid value">some text</UserId>
               <AuthenticationMethod>
                   <PKI>
                       <Issiuer>some text</Issiuer>
                       <Serialnumber>some text</Serialnumber>
                   </PKI>
               </AuthenticationMethod>
```

```
            </User>
            <RequestedResource>
                <Resource>
                    <ResourceID oid="1212">232323</ResourceID>
                </Resource>
            </RequestedResource>
        </LogMessage>
    </hpsa:Message>
</hpsa:Batch>
```

Output XML

The XSL transformation of a log message creates an XML document with the following traits:

- Database tables are represented as XML elements, with element names equal to table names.
- Table columns are represented either as XML elements or as XML attributes, with names equal to column names.

  When XML documents are normalized, white spaces are removed. To maintain white space in column names, the column value must be passed as an element value token.

The elements that represent tables must be nested to reflect the tree structure of the tables.

If a table column is represented as an attribute of a table element, the column's data is passed as the value of the attribute.

Passing text data as XML attribute values could result in unwanted changes to the data. XML attributes have their whitespace compressed by the XML parser; all other whitespace characters (newlines, tabs, and so on) are changed into space characters, and then runs of multiple spaces are reduced to a single space. To exactly preserve white space in text, a column must be represented as XML element with the text in the body of the element.

If a table column is represented as an element, the column element must be nested under its table element and the column's data is passed in the body of the column element. Since both child table elements and column elements are nested under a parent table element, a child table must have a name that is different from names of columns of its parent table to avoid confusion.

The XSL transformation is applied to the DOM sub-tree rooted at a `<hpsa:Message>` element. The root of the DOM tree is the `<hpsa:Batch>` element. In XSLT, a `"/"` pattern still means the root of the tree (not the sub-tree), and the evaluation of global variables and parameters is done from the root node.

Output XML Example

```
<AUDITEVENT>
  <BOPAUDITEVENT AUTH_PKI_ISSUER="some text" AUTH_PKI_SERIAL="some text"
   AUTH_SMS="" AUTH_USERNAME="" BOPEVENTTYPE_ID="logeventtypesome text1"
   BOPLOGTYPE_ID="logtypesttring1" BOP_LOGID="555555"
   TIME_STAMP="8888888" USER_ACTIVEASSIGNMENT="some text" USER_ID=
   "some text"
   USER_ID_TYPE="some text" USER_ORGANIZATIONALUNIT="some text">
    <BOPLOGTYPES>
```

```
      <LOG_TYPE>logtypesttring1</LOG_TYPE>
    </BOPLOGTYPES>
    <BOPEVENTTYPES>
      <EVENT_TYPE>logeventtypestring1</EVENT_TYPE>
    </BOPEVENTTYPES>
    <BOPREQUSTEDRESOURCE BOPCALLDENYREASON_ID="676767"
     BOPRESOURCE_ID="232323" CALL_DIRECTION="34534535"
     CALL_RESULT="45454"
     COMPLETE_DATA_OBJECT="1234123412" OPERATION="Erase"
     OPERATIONAL_OBJECT="some text">
      <BOPRESOURCE>
        <BOP_RESOURCEID>232323</BOP_RESOURCEID>
        <DESCRIPTION>some text</DESCRIPTION>
        <NAME>some text</NAME>
        <RESOURCE_TYPE>1212</RESOURCE_TYPE>
      </BOPRESOURCE>
      <BOPCALLDENYREASON DESCRIPTION="6796796" REASON="676767" />
     <BOPREPLYINFORMATION CARE_UNDERTAKING="some text"
       ORGANIZATIONALUNIT="some text" PATIENT_ID="some text"
       PATIENT_TYPE="some text" RESOURCE_ID="some text" />
    </BOPREQUSTEDRESOURCE>
  </BOPAUDITEVENT>
</AUDITEVENT>
```

Selecting the XSL Stylesheet

The Configurable Normalizer can support many XSL stylesheets and apply different style sheets to different log messages. The XSL stylesheets are configured in rows of the APPLICATION table, one stylesheet per row.

The XPath Expression

Each XSL stylesheet is linked to an XPath expression. The XPATHEX column contains an XPath expression that evaluates the message XML.

To choose which XSL stylesheet should be applied, the Configurable Normalizer evaluates XPath expressions (stored in rows of the APPLICATION table) to select a node in an XML document with a log message. The expressions are evaluated and namespaces are resolved relative to a `<hpsa:Message>` element node as the Context node.

If the expression selects any node in the log message, the Normalizer will apply the corresponding XSL stylesheet and table declarations to normalize the log message.

If the expression does not select a node, the XPath expression in a next row of APPLICATION table is evaluated, until end of table is reached. If none of the XPath expressions selects a node, the log message is stored un-normalized in the `MESSAGE` column of the AUDITEVENT table.

The COMPONENTEVENTTYPES and COMPONENT Tables

The node returned from the evaluation is used to set a value of COMPONENTEVENTTYPEID column of AUDITEVENT table. The following algorithm is used by the Normalizer:

1 Execute the following SQL query:

```
SELECT cet.XPATHEX, cet.ID FROM COMPONENTEVENTTYPES cet, COMPONENT c
WHERE cet.COMPONENTID = c.ID and c.APPLICATIONID = <id>
```

where <id> is the value of ID column in the row of APPLICATION table where the evaluated XPath expression is stored.

2 The node returned by XPath expression is matched as text to cet.XPATHEX value in the result set from the database query. The corresponding value of cet.ID s used as value of COMPONENTEVENTTYPEID column.

3 If there is no match in the entire result set, then COMPONENTEVENTTYPEID is set to -1.

XPath Expression Example

```
./*[local-name()=''LogMessage'']/*[local-name()=''Head'']/
*[local-name()=''LogEventType'']/@*[local-name()=''oid'']
```

Loading the Configuration

Each row of the APPLICATION table represents a Normalizer configuration. The columns of the APPLICATION table have the following meanings:

- ID: The primary key, also used to determine a value for the COMPONENTEVENTTYPEID column of the AUDITEVENT table (see a section above for details).
- APPLICATIONNAME: Text to identify the row in log and debug messages that are generated by the Normalizer. Each row should have unique value of this column.
- XPATHEX: The XPath expression evaluated on log messages.
- CLASS: Binds the row to the Configurable Normalizer. The column value must be com.hp.ov.selectaudit.auditserver.common.normalizer.ConfiguredXmlNormalizer to bind this row to the Configurable Normalizer.
- CONFIG: The bytes of XML document with XSL stylesheet, in UTF-8 encoding.
- CONFIG_TABLES: The bytes of XML document with declaration of output tables, in UTF-8 encoding.

Select Audit provides a tool for loading the values of the CONFIG and CONFIG_TABLES columns. To run the tool, execute the following command:

```
java.exe -classpath
DBTools.jar;ojdbc14.jar;sqljdbc.jar;mssqlserver.jar;msbase.jar;msutil
.jar auditDBUtil.Xml2DbUpdate <db.url> <db.username> <db.password>
<file.name> "UPDATE APPLICATION SET <column>=? WHERE ID=<id>"
```

where

<db.url>	**Oracle:** jdbc:oracle:thin:@<host>:<port>:<sid> **MSSQL:** jdbc:sqlserver://<host>:<port>;DatabaseName=<db> or jdbc:microsoft:sqlserver:// <host>:<port>;DatabaseName=<db>
<db.username>	The user name for login to the database.
<db.password>	The user password for login to the database.
<file.name>	The name of file on disk with the XML document with either XSL stylesheet or with table declarations.
<column>	CONFIG for XSL stylesheet or CONFIG_TABLES for table declarations.
<id>	The value of the primary key of the row of APPLICATION table where values should be loaded.

Testing and Debugging

The Configurable Normalizer uses log4j to output debug messages. To see debug messages from the Configurable Normalizer, the following line should be added to log4j configuration:

```
log4j.logger.com.hp.ov.selectaudit.auditserver.common.normalizer=DEBUG
```

The debug output includes the following:

- The value of the ID column in the APPLICATION table in the row with the XSLT.
- The Input message XML.
- The Output XML transformed by the XSLT from that row.

Select Audit provides a tool for applying the XSLT to message XML outside the Audit Server.

Index