Service Architecture Leveraging Tuxedo (SALT)

Release Notes

12*c* Release 2 (12.1.3)

April 2014

ORACLE

Oracle Service Architecture Leveraging Tuxedo (SALT) Release Notes, 12*c* Release 2 (12.1.3)

Contents

SALT 12*c* Release 2 (12.1.3) Release Notes

SALT 12*c* Release 2 (12.1.3) Release Notes

Date: April 2014

Table 1 Revision History

Revision Date	Summary of Change
April 2014	GA Release

The following topics are discussed in this section:

About This SALT Release

SALT12c Release 2 (12.1.3)

What's New and Improved

This release introduces the following features:

RESTful Web API

Existing Oracle Tuxedo services can be made available to and accessed by http clients as RESTful Web services eliminating the need to use SOAP/http for lightweight applications and expediting integration with other applications. RESTful web API can use XML or JSON payload for data transfer.

In addition to accessing Oracle Tuxedo services as RESTful Web services, Oracle Tuxedo applications can also access external RESTful services without having to write any code. Oracle Tuxedo applications can invoke RESTful services the same as if invoking Oracle Tuxedo services. The SALT gateway acts as the proxy for RESTful Web services.

For more information, see *Enabling the SALT Configuration Tool, REpresentational State Transfer (REST) Option* in the SALT Configuration Guide.

Custom HTTP Headers

HTTP headers can pass relevant application control information to or from Oracle Tuxedo services. For incoming RESTful Web services, any custom HTTP header is attached to an Oracle Tuxedo buffer and passed to the invoked Oracle Tuxedo service. An Oracle Tuxedo service header can be read using a provided API. Similarly, an Oracle Tuxedo application can set HTTP headers in an Oracle Tuxedo buffer using a provided API, which in turn is converted to an HTTP header by the SALT gateway.

For more information, see *Enabling the SALT Configuration Tool, Custom HTTP Headers* in the SALT Configuration Guide.

WS-Security for External Web Services

Message-level authentication is provided using an X.509 certificate to sign messages. Oracle Tuxedo can invoke an external Web service using SOAP/http with the principal identity of the X.509 certificate.

For more information, see *Configuring Oracle Tuxedo Web Services/Configuring Security Features* in the SALT Configuration Guide.

Data Transformation Tracing

Tracing all incoming and outgoing messages is enabled (including RESTful Web services, SOAP/http Web services, and all data transformation from XML to Oracle Tuxedo buffers and vice-versa).

For more information, see *XML-to-Tuxedo Data Type Mapping for External Web Services* in Data Type Mapping and Message Conversion Services of the SALT Programming Guide.

GWWS Tracing

The verb "`ws`" supersedes and replaces the previous "`ms`."

Note: You can enable GWWS Tracing by setting `TMTRACE` or `changetrace` command in the `tmadmin` utility.

For more information, see *Tracing the GWWS Server* in the SALT Administration Guide.

ECID Propagation

ECID (Execution Context ID) is propagated with each request within Oracle Tuxedo and across various products in an Oracle stack. ECID propagation enables request correlation across Oracle Tuxedo domains and Oracle products (such as Oracle WebLogic Server, Oracle Database, etc.), making it quicker to diagnose application problems.

For more information, see Configuring Tuxedo for Propagating ECID.

Dynamic Configuration and MIB

You can dynamically reload configuration file changes without any downtime. An MIB interface is provided, which enables reading Web services configuration and runtime statistics.

For more information, see *Enabling the SALT Configuration Tool, MIB Class Interface* in the SALT Configuration Guide.

XML Complex Attribute Mapping

Enhanced usability is provided by using an attribute field within a `complexType` element in a WSDL for accessing external Web services. `complexType` attribute fields are mapped “one-by-one” to corresponding FML32 fields.

For more information, see *Data Type Mapping and Message Conversion* in the SALT Programming Guide.

Configuration Tool Enhancements

The configuration tool has been enhanced to provide the following features:

- Support for RESTful Web services

- Enables import of external Web services
- Support for FireFox and Safari Web browsers
- A test client for RESTful Web services

For more information, see *The* SALT *Configuration Tool* in the SALT Configuration Guide.

Upgrade Considerations

If a previous SALT release is installed, it must be uninstalled before installing SALT 12c Release 2 (12.1.3). SALT 12c Release 2 (12.1.3) cannot co-exist with a previous SALT installation on the same Oracle Tuxedo installation.

For information, see *Migrating from SALT 1.1 Application* in the SALT Configuration Guide.

SALT Installation Prerequisites

Before installing SALT 12*c* Release 2 (12.1.3), you must ensure that Oracle Tuxedo 12c Release 2 (12.1.3) is installed.

SALT 12c Release 2 (12.1.3) is included in Oracle Tuxedo 12c Release 2 (12.1.3) installer. For more information, see the Oracle Tuxedo 12c Release 2 (12.1.3) Installation Guide.

SALT Platform Support

SALT 12*c* Release 2 (12.1.3) supports the same platforms as Oracle Tuxedo 12c Release 2 (12.1.3).

Interoperability Considerations

SALT 12*c* Release 2 (12.1.3) is compatible with (and fully supports), most industry-standard Web service development toolkits. For more information, see *Interoperability Considerations* in the SALT Interoperability Guide.

Known Issues

The following sections describe known problems in SALT 12*c* Release 2 (12.1.3). Entries include a description of the problem, and a workaround or solution where appropriate.

Each problem is listed by the Change Request (CR) or BugDB number.

- GWWS Runtime

- Interoperability

GWWS Runtime

CR Number	Description and Workaround or Solution	Found In
CR334161	**Problem:** `GWWS rejects non UTF-8 inbound SOAP request messages when SignBody WS-Security Policy is enabled.` When `GWWS` is configured with multiple encoding support, it can accept non `UTF-8` encoded SOAP requests; however, the `GWWS` internally converts all non `UTF-8` encoding messages into `UTF-8` encoding messages for later operation. If a service requires <`soap:Body`> signature verification, the `GWWS` always verify the signature against the converted `UTF-8` encoded <`soap:Body`> instead of the original <`soap:Body`> content. Thus the signature verification always failed.	2.0
	Platform: All	
	Workaround: Web service client programs must initiate SOAP requests using `UTF-8` encoding when the WS-Security Policy Assertion `SignBody` is enabled for the corresponding services.	
CR328329	**Problem:** `GWWS` **may reject valid SOAP requests if the target Tuxedo service consumes XML typed buffer as input and the input buffer is defined with "size" restriction in the Tuxedo Service Metadata definition.** `GWWS` automatically adds an additional '\0' to the end of the converted XML buffer. This additional byte may result the XML buffer length exceed the "size" value, hence reject by later Tuxedo buffer validation routine in the `GWWS`.	2.0
	Platform: All	
	Workaround: Enlarge or remove the "`size`" restriction for XML typed buffer in the Tuxedo Service Metadata Definition.	

CR Number	Description and Workaround or Solution	Found In
CR306710	**Problem: Tuxedo service may not receive the exact same non `UTF-8` encoding string as the string prepared in the SOAP request message.** If multiple encoding capability is turned on for the `GWWS`, and Web Service client programs written in Java send messages with non `UTF-8` encoding, `GWWS` may not send exact the same string value to the Tuxedo service. This is a general problem if different encoding conversion implementations are used. Java encoding implementation has slight difference from ICU encoding implementation (which is used by Tuxedo and SALT), hence an encoding string prepared by the Java program, after ICU "to `UTF-8`" and "from `UTF-8`" conversion, may not revert to the exact original string.	2.0
	Platform: All	
	Workaround: None. Customers rarely use those characters. If some characters mapping are confirmed due to ICU bugs, please contact Oracle Tuxedo Customer Support.	
9281764	**Problem:** WCF (.Net) C# clients may not handle SOAP 1.2 faults sent back by the GWWS gateway. When an Tuxedo service is exposed as a Web Service using the SOAP 1.2 style, GWWS may return faults such as service unavailable, problems during invocation, etc. When this happens, a C# client will receive an exception indicating that an invalid SOAP fault was received. This is due to the fact that GWWS does not specify an `@xml:lang` attribute in the `/Fault/Reason/Text` element returned.	11gR1
	Platform: All	

	Workaround:
	Code the C# client so that it receives the fault as an XML document which can then be parsed, as follows:
	`...`
	`catch (FaultException ex) {`
	`  MessageFault mf = ex.CreateMessageFault();`
	`  XmlReader xr = mf.GetReaderAtDetailContents();`
	`  ...`
	`}`
	`--`	

Interoperability

CR Number	Description and Workaround or Solution	Found In
CR330363	**Problem: SALT** `multiple encoding feature does not interoperable with Microsoft .NET WCF 3.0 engine.` If SALT enables multiple encoding feature, when the inbound call Tuxedo service returns `MBSTRING` or `XML` typed buffer with non `UTF-8` encoding, the SOAP response message is encoded the same as the `MBSTING` or `XML` buffer. Such SOAP response message cannot be accepted by those Web Service client applications developed using Microsoft .NET WCF 3.0 engine.	2.0
	Third-Party Web Service Toolkit: Microsoft .NET WCF 3.0	
	Workaround: Customers may need to develop custom encoder/decoder if the Tuxedo service may return non `UTF-8` typed buffers and `GWWS` multiple encoding feature is turned on. Alternatively, you may explicitly turn off the `GWWS` multiple encoding feature if you are aware all Tuxedo services in your Tuxedo domain never return non `UTF-8` buffers.	
CR296594	**Problem: SOAP fault response message cannot be accepted by Microsoft .NET 3.0 when the HTTP Content-Length exceeds 65536.** If the `GWWS` server returns a SOAP fault message when the HTTP Content-Length exceeds 65536, the .NET WCF 3.0 engine sends an exception to report the response is not well-formed. **Note:** If the `GWWS` server returns a normal SOAP message (non SOAP fault) when the HTTP Content-Length exceeds 65536, the .NET Web service engine can accept.	2.0
	Third-Party Web Service Toolkit: Microsoft .NET WCF 3.0	
	Workaround: None. Avoid to return big buffer when invoking `tpreturn()` along with TPFAIL status code in the Tuxedo service.	

CR Number	Description and Workaround or Solution	Found In
CR294785	**Problem: Apache Axis2/Java fails to handle Tuxedo FML32 TPFAIL response buffers that have field names with initial uppercase.** If a Tuxedo service returns TPFAIL with FML32 buffer, SALT maps each field as an XML segment in the SOAP fault detail, and the field name is used directly as the XML element tag name. If the FML32 buffer contains field names with initial letter uppercase, Axis2 may not recognize the SOAP fault messages that converted from this Tuxedo FML32 buffer.	2.0
	Third-Party Web Service Toolkit: Apache Axis2/Java	
	Workaround: Modify the FML32 field name to avoid use initial uppercase name. Corresponding Tuxedo application also needs to be changed and re-compiled.	
CR306978	**Problem: Apache Axis2/Java does not recognize the SOAP with Attachment (SwA) featured WSDL file generated by** SALT**.** If SwA featured WSDL file is generated by SALT, Apache Axis2 `wsdl2java` utility generates Java stub code which is different from Apache Axis. Axis2 generated stub code cannot initiate a successful call to SALT service.	2.0
	Third-Party Web Service Toolkit: Apache Axis2/Java	
	Workaround: Use Apache Axis instead for SwA featured soap calls. MTOM is an alternative attachment format that supported by SALT. You may also use MTOM feature with Apache Axis2/Java for CARRAY buffer stream.	

CR Number	Description and Workaround or Solution	Found In
CR296221	**Problem: Apache Axis wsdl2java utility fails to compile the SALT generated WSDL file if soap 1.2 binding with soap fault is defined in the WSDL file.**	2.0
	Third-Party Web Service Toolkit: Apache Axis	
	Workaround: This is an Apache Axis bug, please refer to https://issues.apache.org/jira/browse/AXIS-2614. You may define SOAP version 1.1 for SALT WSDL if Apache Axis has to be used for Web Service client programming. Or you should manually re-compile Apache Axis classes using Apache Axis source code with the fix provide in the above URL link. You may also choose another third-party Web Service client toolkit for soap 1.2 binding with soap fault feature, such as Oracle WebLogic 9.x Web Services, Apache Axis2, Microsoft .NET WCF 3.0, etc.	

CR Number	Description and Workaround or Solution	Found In

CR323477	**Problem: GWWS fails to call external Web Service applications built upon Microsoft .NET WCF 3.0 if asynchronous WS-Addressing feature is enabled.**
	SALT supports WS-Addressing feature that conforms to WS-Addressing standard 200408 submission. While initiating an asynchronous outbound call, `GWWS` always defines a `<wsa:ReplyTo>` endpoint reference in the WS-Addressing soap header. See the following sample `<wsa:ReplyTo>` segment:
	`<wsa:ReplyTo>`
	`  <wsa:Address>`
	`  http://myhost:7102/?wsa_Msg_ID=uuid:B437A4F4-AF23-111E-FFFFFFAC1622FFFFFF9F0000-6BBE`
	`  </wsa:Address>`
	`</wsa:ReplyTo>`
	Host name "`myhost`" and port number "`7102`" in the above sample indicates the listening endpoint that is created by the `GWWS` which is used to accept asynchronous soap response messages for outbound calls.
	But Microsoft .NET WCF 3.0 does not recognize the `<wsa:ReplyTo>` endpoint in the request, and always returns the synchronous response through the request connection.
	`GWWS` then always encounters time out in receiving asynchronous response because Microsoft .NET WCF 3.0 never send the response to `GWWS` expected endpoint.
		2.0

	Third-Party Web Service Toolkit: Microsoft .NET WCF 3.0	
	Workaround: None. You should disable WS-Addressing feature when initiating outbound call to external Web Service applications built upon Microsoft .NET WCF 3.0. For more information about configuring WS-Addressing feature, see "Configuring Advanced Web Service Messaging Features" in the SALT *Configuration Guide*.	

See Also

- SALT 12c Release 2 Product Overview

- SALT 12c Release 2 Administration Guide
- SALT 12c Release 2Configuration Guide
- SALT 12c Release 2 Programming Guide
- SALT 12c Release 2 Reference Guide