Enhancements to SMI and Access logging One Pager

1. Introduction

1.1. Project/Component Working Name

SailFin 2.0 / Reporter framework

1.2. Name(s) and e-mail address of Document Author(s)/Supplier:

Peter Strand: peter.strand@ericsson.com

1.3. Date of This Document:

Version	Date	Comments	Author
0.1	2009-02-27	First version	Peter Strand
0.2	2009-03-09	Added toc	Peter Strand
1.0	2009-04-15	Updated w.r.t implemntation	Peter Strand

2. Project Summary

2.1. Project Description:

This document describes some enhancements of the current Reporter framework (Details specific to the SipMessageInspection feature is described in a separate document named "Sip Message Inspection Adapter One Pager").

The implementation classes of the current features CallFlow, AccessLog and

SipMessageInspection implements the com.ericsson.ssa.container.callflow.Reporter interface. However the current implementation only fully intercepts incoming initial requests/responses and it does not have support to intercept outgoing Sip Requests or Sip Responses.

See below for a more detailed technical description.

2.2. Risks and Assumptions:

N/A

3. Problem Summary

3.1. Problem Area:

The current Reporter framework does not support interceptions of outgoing Sip Requests or Sip Responses.

3.2. Justification:

The following problems exist with the current implementation:

- The current Sip Message Inspection feature does not log outgoing requests/responses (which limits its purpose)
- Interceptions on Servlet level does not work for subsequent requests (This affects CallFlow and SipMessageInspection)
- The current AccessLog implementation for SIP does not show the response code in the log record
- AccessLog implementation for SIP does not log outgoing messages (To be defined if this is required)

4. Technical Description:

4.1. Details:

The implementation classes of current features CallFlow, AccessLog and SipMessageInspection implements the com.ericsson.ssa.container.callflow.Reporter interface. However the current implementation only intercepts incoming requests/responses, it does not have support to intercept outgoing Sip Requests or Sip Responses.

In the Code examples section below the proposed refined Reporter interface is shown. The implementation of the changed Reporter framework involves a significant number of classes including minor impact on each Layer implementation but the "core" sip stack functionality is not affected.

An alternative implementation to the proposed refined Reporter interface would be to "simply add additional methods" for outgoing requests/responses into the existing interface. However the assumption is that it would not be a clean and distinct solution as it would require at least additionally four new methods (reportPreOutgoing(req), reportPreOutgoing(resp), reportPreOutgoing(SipServletFacade servlet, req), reportPreOutgoing(SipServletFacade servlet, resp). It would also affect the framework and the existing reporters to a significant extent.

The interceptions of incoming requests/responses is proposed to be maintained via the LayerHelper.next() methods. However the interceptions of outgoing messages cannot easily be implemented in a generic way at least not without major impact of the existing core sip stack. Therefore it is proposed to implement interceptions by selecting the appropriate places in a non-generic way. For the first release it is proposed to only intercept outgoing messages where it is needed for the features/reporters which are delivered by default (Sip Message Inspection (SMI) and AccessLog):

- ApplicationDispatcher layer (InspectionType: SERVLET) (i.e. intercept calls to request.send() and response.send() done from Servlet code)
- GrizzlyNetworkManager layer (InspectionType: LAYER) (i.e. intercept dispatch() calls in GrizzlyNetworkManager and ResponseDispatcher)

If it would be proven necessary interceptions for outgoing messages in other layers can be added later on.

4.1.1 Code examples

4.1.1.1 Interface

The current interface which is implemented by the CallFlow, AccessLog and SipMessageInspection features lacks support to intercept outgoing Sip Requests and Sip Responses looks like this:

```
package com.ericsson.ssa.container.callflow;
public interface Reporter {

    void reportPre(Layer l, SipServletRequest r);
    void reportPost(Layer l, SipServletRequest r);
```

```
    void reportPre(Layer l, SipServletResponse r);
    void reportPost(Layer l, SipServletResponse r);

    void reportPost(SipServletFacade servlet, SipServletRequest request
    void reportPre(SipServletFacade servlet, SipServletRequest request);

    void reportPost(SipServletFacade servlet, SipServletRequest request,
}
```

The refined interface including support for outgoing messages:

```
package com.ericsson.ssa.container.reporter;

public interface Reporter {
        enum InterceptionType { LAYER, SERVLET, ALL };
        InterceptionType getSupportedInterception();

    void logIncomingRequest(InterceptionType interceptType, SipServletRe
    void logOutgoingRequest(InterceptionType interceptType, SipServletRe
    void logIncomingResponse(InterceptionType interceptType, SipServletR
    void logOutgoingResponse(InterceptionType interceptType, SipServletR

    void logPostIncomingRequest(InterceptionType interceptType, SipServl
    void logPostIncomingResponse(InterceptionType interceptType, SipServ
}
```

The new Reporter interface and framework classes should reside in a new package "com.ericsson.ssa.container.reporter". A class which implements the Reporter interface should return its supported invocation type. E.g. A reporter which is configured for the ApplicationDispatcher layer may be valid for both LAYER and SERVLET interception but a reporter class must select which one of interception types it supports.

4.1.1.2 Interceptions to be added

The proposal is to implement outgoing interception points for

- ApplicationDispatcher/SipServletWrapper (InspectionType: SERVLET) (When sending requests/responses from within Servlet)
- GrizzlyNetworkManager (InspectionType: LAYER)

Besides adding outgoing interceptions it is proposed to add missing interceptions on Servlet level:

- Subsequent requests including Ack and Cancel (Cancel has to be intercepted in InviteServerTransaction)

4.1.1.2 Interception example

Example of interception implementation (in this case for an incoming request):

```
public final class LayerHelper {
    public static void next(SipServletRequestImpl req, Layer layer,
        Layer nextLayer) {
        if (nextLayer == null) {
            layer.dispatch(req);
        } else {
            Reporter reporter = nextLayer.getReporter();
            if(reporter != null) {
                reporter.logIncomingRequest(Reporter.InterceptionType.L
            }

            nextLayer.next(req);

            if(reporter != null) {
                reporter.logPostIncomingRequest(Reporter.InterceptionTy
            }
        }
    }
....
```

4.2. Bug/RFE Number(s):

4.2.1 Bug/RFE Numbers from Issue Tracker

4.2.2 Requirememt Ids that are being addressed as a part of this proposal.

N/A

4.3. In Scope:

N/A

4.4. Out of Scope:

In this proposed that only selected layers will do the interception of outgoing Sip request/responses (In this release: NetworkManager and ApplicationDispatcher)

The details of AccessLogReporter implementation are left out of this proposal (changes may be needed in AccesLog GUI and/or AccessLog file format depending on solution).

4.5. Interfaces:

4.5.1 Exported Interfaces

N/A

4.5.2 Imported interfaces

N/A

4.5.3 Other interfaces

N/A (this proposal only affects the Reporter interface which is internal to the container)

4.6. Doc Impact:

N/A (Assume this OnePager is sufficient)

4.7. Admin/Config Impact:

4.7.1 Configuration changes needed

No changes needed in dtd for domain.xml

4.7.2 CLI / GUI impact if any

No changes needed for the modified Reporter framework. (AccessLog details excluded in this document)

4.7.3 Plugin impact

The classes which are implementing the Reporter interface need to be re-implemented w.r.t. new Reporter interface (i.e. CallFlowReporter, SipMessageReporter, AccessLogReporter)

4.8. HA Impact:

4.9. I18N/L10N Impact:

4.10. Packaging & Delivery:

4.10.1 Binaries in which the code are delivered

The modified reporter framework interfaces and classes are part of building sailfin/sip-stack (ends up in comms-appserv-rt.jar).

The modified reporter classes should remain under sailfin/integration (ends up in comms-appserv-rt.jar).

4.11. Security Impact:

N/A

4.12. Compatibility Impact

4.13. Dependencies:

N/A

4.13.1 Changes required in GlassFish

No.

4.13.2 Third Party APIs

N/A

4.14 Miscellaneous

Will this component work with Ipv6 addresses	Yes
Will this component work with JDK 64bit	Yes
Will this component require configuration using a sun-specific deployment descriptor.If yes, please specify below that configuration elements needed	No

5. Open Issues

Issue No	Description	Comments	Resolution

6. Reference Documents:

7. Schedule:

7.1. Projected Availability:

SGCS 2.0