End-to-End Tracing for OData applications on iOS and Android

Applies to:

This document and code samples provided here is valid for product Sybase Unwired Platform (SUP) - Online Data Proxy (ODP), release version 2.2.

Summary

This document describes the usage of End-to-End Tracing feature of SUP ODP on iOS and Android Platforms. The aim of the article is to help an SUP OData application developer enable the client side E2E tracing so that the application user can trigger the generation of Business Transaction Xml (BTX) and upload the BTX to Solman for analysis.

Authors: RESHMA L RAGHU, RADHAKRISHNA RAMASESHU

Company: SAP Labs India Pvt. Ltd.

Created on: 22 November 2012

Author Bio

Author name : Reshma L Raghu

Company name : SAP Labs India Pvt. Ltd.

Field of work : Associate Developer, iOS Platform, ODP, Mobile Platform IND

Author name : Radhakrishna Ramaseshu

Company name : SAP Labs India Pvt. Ltd.

Field of work : Associate Developer, Android Platform, ODP, Mobile Platform IND

Table of Contents

Introduction

E2E tracing feature can be used to trace the path followed by each client request fired from the client/device as it passes through various components (device-SUP-Gateway-Backend). The request is tracked using SAP-Passport and Business Transaction Xml.

On iOS platform, a new library “libSUPSupportability” will expose the features to generate the SAP-Passport and Business Transaction Xml. Application developers can use E2E feature in two ways:

- Use libSUPSupportability as a standalone library to generate SAP-Passport and BTX
- Use ODP wrapper to generate SAP-Passport, BTX and upload BTX.

End-to-End Trace Use Case/Data Flow

- Application user calls SUP Admin with a problem.
- SUP Admin enables E2E Trace using SCC.
- SUP Admin informs trace level and asks Application User to start trace – User starts trace.
- Application user tries a transaction.
- Application user stops tracing – BTX is generated.
- Application user uploads BTX to SolMan through SUP
- SUP Admin views BTX in SolMan
- SUP Admin views E2E trace records through SCC
- SUP Admin looks through E2E records and BTX to identify the problem.

Using wrapper APIs in OData SDK in iOS

One way of enabling end to end tracing is by using supportability-related APIs defined in the libSUPProxyClient library.

This is a code sample:

```
@try{
   /*There are four trace levels: LOW, MEDIUM, HIGH and NONE*/
   [SDMRequestBuilder setTraceLevel:HIGH];

   /*Start the tracing*/
   [ODPRequest startTrace];

   /*Make the requests*/
   id<SDMRequesting> serviceDocRequest =
[SDMRequestBuilderrequestWithURL:[[NSURLalloc]
initWithString:@"http://ldcig8p.wdf.sap.corp:50018/sap/opu/odata/iwfnd/RMT
SAMPLEFLIGHT/"]];
[serviceDocRequest setUsername:@"perfios"];
[serviceDocRequest setPassword:@"perfios"];
[serviceDocRequest setRequestMethod:@"GET"];
[serviceDocRequestsetTimeOutSeconds:60];
```

```
      [serviceDocRequest startSynchronous];

        /*Stop the tracing*/
        [ODPRequest stopTrace];
        /*Upload the BTX to Solman via SUP*/
        NSError* error = nil;
       [ODPRequest uploadTraceWithError:&error];
        if(error)
        {
                NSLog(@”Upload failed : %@”,error);
        }
        else{
                NSLog(@”Upload successfull”);
  }

    }
    @catch (NSException *exception) {
            NSLog(@"excption:  %@",[exception description]);
    {
```

Using Tracing APIs in Standalone Library in iOS

You can perform end to end tracing using standalone libSUPSupportability library. libSUPSupportability library provides two functionalities:

- generation of SAP Passport and Correlation ID for HTTP request and response messages.
- handling the generation of structured E2E Business Transaction XML .

This is a sample code :

```
/*Get an instance of TraceController*/
SUPE2ETraceController *traceController = [SUPE2ETraceController
sharedController];
/*Set the trace level: LOW, MEDIUM, HIGH or NONE*/
[traceController setTraceLevel:HIGH];
[traceController createTransaction];
SUPE2ETraceStep *traceStep=[traceController createStep];
SUPE2ETraceRequest *traceRequest = [traceController createRequest];
[SDMRequestBuilder setRequestType:HTTPRequestType];
id<SDMRequesting> serviceDocRequest = [SDMRequestBuilder requestWithURL:[[NSURL
alloc]
initWithString:@"http://ldcig8p.wdf.sap.corp:50018/sap/opu/odata/iwfnd/RMTSAMPLE
FLIGHT/"]];
[serviceDocRequest setUsername:@"perfios"];
```

```
[serviceDocRequest setPassword:@"perfios"];

[serviceDocRequest setRequestMethod:@"GET"];

[serviceDocRequest setTimeOutSeconds:60];

[serviceDocRequest addRequestHeader:@"SAP-PASSPORT" value:[traceRequest
PassportHttpHeader]];

[serviceDocRequest addRequestHeader:@"X-CorrelationID" value:[traceRequest
CorrelationIdHttpHeader]];

NSMutableDictionary* finHeaders = [[NSMutableDictionary alloc]
initWithDictionary:serviceDocRequest.requestHeaders];

NSDictionary *theDict=[NSDictionary dictionaryWithObjectsAndKeys:[NSNull
null],@"requestLine",[serviceDocRequest.url
absoluteString],@"requestURL",finHeaders,@"requestHeader",[serviceDocRequest
requestMethod],@"requestMethod",[NSString stringWithFormat:@"%d",
([serviceDocRequest.postBody length])],@"sent", nil];

[traceController updateRequestElementsInRequest:traceRequest
withDictionary:theDict];

[traceRequest markSending];

[traceRequest markSent];

[serviceDocRequest startSynchronous];

[traceRequest markReceiving];

[traceRequest markReceived];

[traceController updateRequestElementWithValue:traceRequest :[NSString
stringWithFormat:@"%d", [serviceDocRequest responseStatusCode]] :@"returnCode"];

[traceController updateRequestElementWithValue:traceRequest :[NSString
stringWithFormat:@"%d",[[serviceDocRequest responseString] length]] :@"rcvd"];

NSDictionary* finalDict = [NSDictionary dictionaryWithObject:[serviceDocRequest
responseHeaders] forKey:@"responseHeader"];

[traceController updateRequestElementsInRequest:traceRequest
withDictionary:finalDict];

[traceRequest endRequest];

[traceStep endStep];

[traceController getXML]);

[traceController sendXML:[[NSURL alloc] initWithString:@"
http://ldai2sd7.wdf.sap.corp:50089/E2EClientTraceUploadW/UploadForm.jsp"]];
```

Using wrapper APIs in OData SDK in Android

One way of enabling end to end tracing is by using supportability-related APIs defined in the libSUPProxyClient library.

This is a code sample:

```
ODPClientConnection ocl = ODPClientConnection.getInstance();
ocl.startTrace();
ocl.setTraceLevel(ocl.TrcLvl_MEDIUM);
        /*Fire the request(s)*/
ocl.stopTrace();
try {
    ocl.uploadTrace();
} catch (ODPException e) {
    e.printStackTrace();
}
```

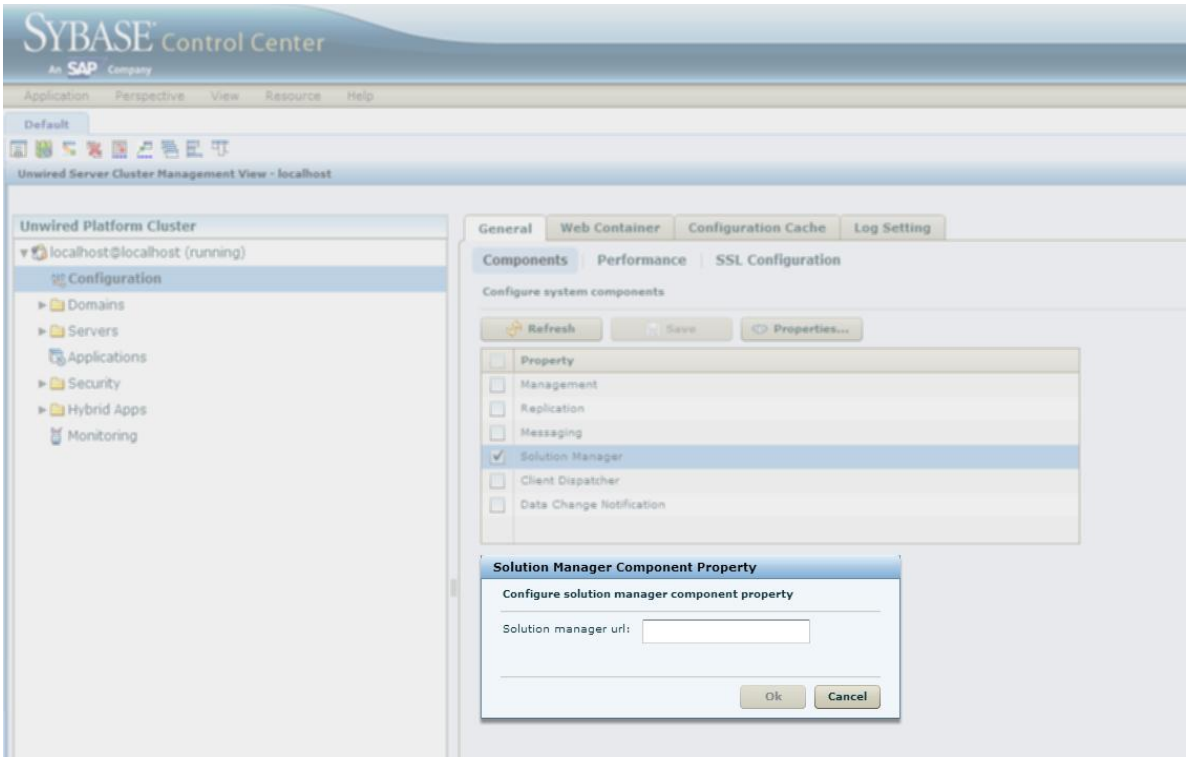
End-to-End tracing server side

1. Setting Solman URL in SCC

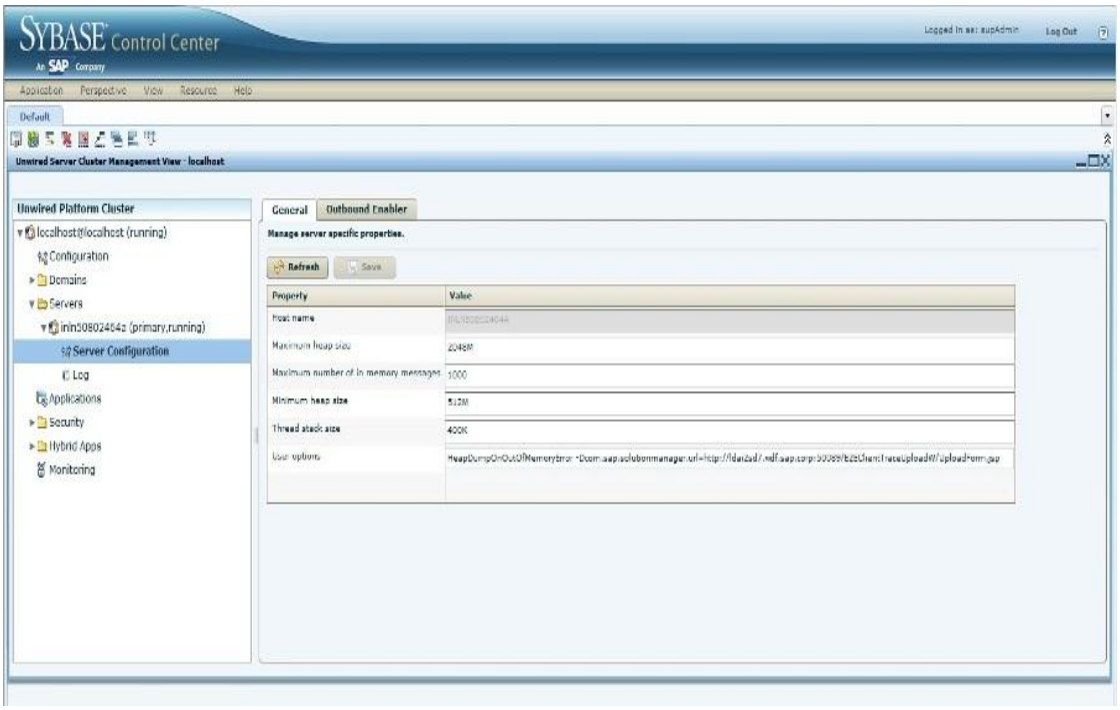
Cluster level setting:

Go to SCC → Configuration → General → Solution Manager → properties

And enter the Solam Url, example - http://ldai2sd7.wdf.sap.corp:50089/E2EClientTraceUploadW/UploadForm.jsp

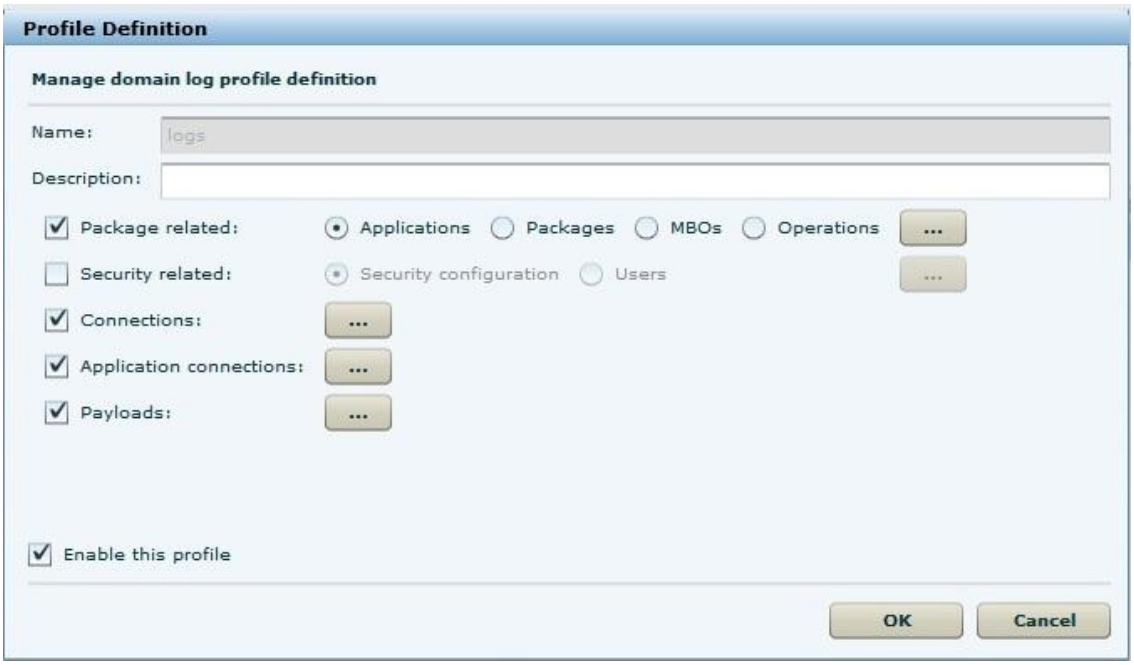


Server level setting:
Go to SCC → Servers → Server Configuration → General → User options
And append the url in the end in the following format-
Dcom.sap.solutionmanager.url=<SolMan URL>

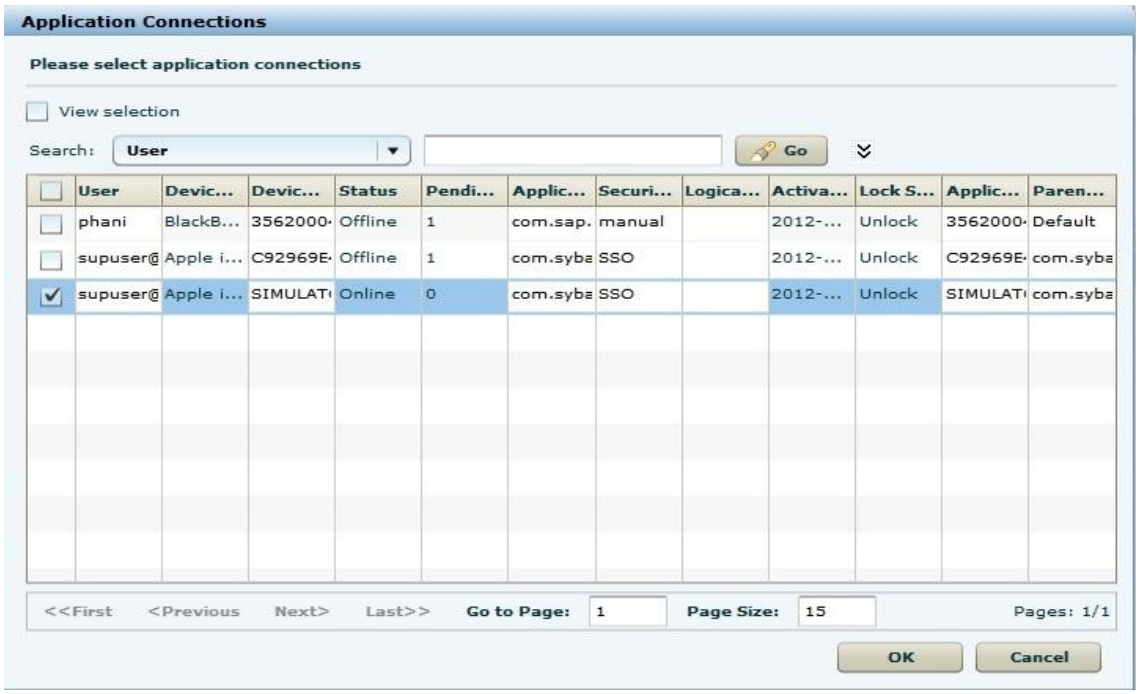


2. Enabling Tracing for an ApplicationConnection in SCC

Go to SCC → Domains → default → Log → Settings → properties → Application connections



Check the user for which tracing has to be enabled.



Related Content

Sample Android app

Sample iOS app

Beginners guide to iOS OData application

Copyright