User Guide
SITRAFFIC® Concert

OCPI 2

Usage of External Interface to implement
Soap Server Interface and the Soap Client Interface

History of change:

Version	Date	Name	Changed
0.1	14.01.09	Kaufmann	Creation
0.2	15.01.09	Kaufmann	Abgeschlossen

1 Introduction

1.1 Definitions

Begriff	Definition
SOAP	Simple Object Access Protocol

1.2 References

Nr.	Title	Version	Autor	Remark
	OCPI2_datacontents.doc			
	Soap_Protocol.doc			
	Soap_Protocol_en.doc			
	VMSInstation.doc			Only for message signs
	Protocol.xsd			
	Other data definig xsd-files			

2 Soap Server Interface

2.1 Contents of Delivery

The JAR-Files with the AXIS Open Source SOAP server and client functionality.

ExampleExtIF_Libs.zip:

- axis.jar
- commons-discovery.jar
- commons-logging.jar
- dom4j.jar
- jaxrpc.jar
- log4j.jar
- saaj.jar
- wsdl4j.jar

The JAR-Files realize the Externalinterface and ExternalType functionality and a application with exambles.

ExampleExtIF.zip:

- externalinterface.jar
- externaltypes.jar

2.2 Realize the Soap Server Interface getContentInfo()

```
   /* (non-Javadoc)
    * @see
com.siemens.sitraffic.external.soap.ExternalIf#getContentInfo(com.siemens.sitraffic.external.soap.protocol.GetCo
ntentInfoType)
    */
   public GetContentInfoResponseType getContentInfo(GetContentInfoType getContentInfo) throws
RemoteException
   {
      GetContentInfoResponseType response = new GetContentInfoResponseType();
      response.setContentInfoList(new GetContentInfoResponseTypeContentInfoList());
      response.setLastStart(startTime);
      // assume no error
      response.setErrorCode(ErrorCode.value1);
      response.setErrorText("");

      // check user access
      if (checkAccess(getContentInfo) == false)
      {
         // return error
         response.setErrorCode(ErrorCode.value2);
         response.setErrorText("Wrong user or password");
         return response;
      }
      // build response list, this server offers only detector values
      GetContentInfoResponseTypeContentInfoListContentInfo[] ctInfoList = new
GetContentInfoResponseTypeContentInfoListContentInfo[1];
      // create object type info
      GetContentInfoResponseTypeContentInfoListContentInfo objectTypeInfo = new
GetContentInfoResponseTypeContentInfoListContentInfo();
      objectTypeInfo.setObjectTypeName(new NMToken("TrafficData_detector_currentValue"));

      URI dataTypeURI = null;
      try
      {
         TypeDesc typeDesc = DetectorType_Helper.getTypeDesc();
         FieldDesc elemDesc = typeDesc.getFieldByName("currentValue");
         QName dataXmlType = elemDesc.getXmlType();
         dataTypeURI = new URI(dataXmlType.getNamespaceURI() + ":" + dataXmlType.getLocalPart());
      }
      catch (Exception e)
      {
         // should not happen
         dataTypeURI = new URI();
      }
      objectTypeInfo.setObjectTypeURI(dataTypeURI);
      // set access rights of the type: only read and write is allowed
      objectTypeInfo.setCreateable(false);
      objectTypeInfo.setDeleteable(false);
      objectTypeInfo.setReadable(true);
      objectTypeInfo.setWriteable(true);

      ctInfoList[0] = objectTypeInfo;
      response.setContentInfoList(new GetContentInfoResponseTypeContentInfoList(ctInfoList));

      return response;
   }
```

2.3 Realize the Soap Server Interface get()

```
   /* (non-Javadoc)
    * @see
com.siemens.sitraffic.external.soap.ExternalIf#get(com.siemens.sitraffic.external.soap.protocol.GetType)
    */
   public GetResponseType get(GetType get) throws RemoteException
   {
      // build default positive empty response
      GetResponseType response = new GetResponseType();
      response.setErrorCode(ErrorCode.value1);
      response.setErrorText("");
      response.setLastStart(startTime);
      response.setDataList(new DataList());
      response.setStoretime(Calendar.getInstance());
      response.setPosition(new UnsignedInt(0));

      // check user access
      if (checkAccess(get) == false)
      {
         // return error
         response.setErrorCode(ErrorCode.value2);
         response.setErrorText("Wrong user or password");
         return response;
      }
      // check if we can offer the requested type
      if (!get.getObjectType().toString().equals("TrafficData_detector_currentValue"))
      {
         // return error
         response.setErrorCode(ErrorCode.value2);
         response.setErrorText("Illegal object type access");
         return response;
      }
      // here we do not care about the filter. the filter should be used to select the objects
      // we only deliver some sample data

      // the position should select values which are newer than the values with that position. here
      // we use the position to select the start index of our sample data
      int position = get.getPosition().intValue();
      DataListDs[] dsList = getData(ObjectState.value2); // data are modified

      if (position < dsList.length)
      {
         DataListDs[] reqList = new DataListDs[dsList.length - position];
         for (int i = position; i < dsList.length; i++)
         {
            reqList[i - position] = dsList[i];
         }
         response.setDataList(new DataList(reqList));
      }
      else
      {
         response.setDataList(new DataList(null));
      }
      response.setStoretime(Calendar.getInstance()); // should be the latest time of the data stored at the server
      response.setPosition(new UnsignedInt(dsList.length));

      return response;
   }
```

2.4 Realize the Soap Server Interface getInquireAll()

```
   /* (non-Javadoc)
    * @see
com.siemens.sitraffic.external.soap.ExternalIf#inquireAll(com.siemens.sitraffic.external.soap.protocol.InquireAllTy
pe)
    */
   public InquireAllResponseType inquireAll(InquireAllType inquireAll) throws RemoteException
   {
      // build default positive empty response
      InquireAllResponseType response = new InquireAllResponseType();
      response.setErrorCode(ErrorCode.value1);
      response.setErrorText("");
      response.setLastStart(startTime);
      response.setDataList(new DataList());
      response.setStoretime(Calendar.getInstance());
      response.setPosition(new UnsignedInt(0));

      // check user access
      if (checkAccess(inquireAll) == false)
      {
         // return error
         response.setErrorCode(ErrorCode.value2);
         response.setErrorText("Wrong user or password");
         return response;
      }
      // check if we can offer the requested type
      if (!inquireAll.getObjectType().toString().equals("TrafficData_detector_currentValue"))
      {
         // return error
         response.setErrorCode(ErrorCode.value2);
         response.setErrorText("Illegal object type access");
         return response;
      }
      // here we do not care about the filter. the filter should be used to select the objects
      // we only deliver some sample data

      DataListDs[] dsList = getData(ObjectState.value1);
      response.setDataList(new DataList(dsList));
      response.setStoretime(Calendar.getInstance()); // should be the latest time of the data stored at the server
      response.setPosition(new UnsignedInt(dsList.length));

      return response;
   }
```

2.5 Realize the Soap Server Interface method put()

To realize data inputs via **put** the following method has to be defined:

```
   /* (non-Javadoc)
    * @see
com.siemens.sitraffic.external.soap.ExternalIf#put(com.siemens.sitraffic.external.soap.protocol.PutType)
    */
   public PutResponseType put(PutType put) throws RemoteException
   {
      // build default positive empty response
      PutResponseType response = new PutResponseType();
      response.setBadList(new PutResponseTypeBadList());
      response.setErrorCode(ErrorCode.value1);
      response.setErrorText("");
      response.setLastStart(startTime);

      // check user access
      if (checkAccess(put) == false)
      {
         // return error
         response.setErrorCode(ErrorCode.value2);
         response.setErrorText("Wrong user or password");
         return response;
      }
      // check if we can offer the requested type
      if (!put.getObjectType().toString().equals("TrafficData_detector_currentValue"))
      {
         // return error
         response.setErrorCode(ErrorCode.value2);
         response.setErrorText("Illegal object type access");
         return response;
      }
      // read data, they are of type TrafficData_detector_currentValue
      PutTypePutListPutds[] putds = put.getPutList().getPutds();
      if (putds == null)
         System.out.println("Put without data");
      else
      {
         for (int i = 0; i < putds.length; i++)
         {
            PutTypePutListPutds putObj = putds[i];
            IdentifierType idType = putObj.getIdentifier();
            CurrentValueType curVal = (CurrentValueType) putObj.getData();
            System.out.println("---------- Put ----------");
            System.out.println("Identifier: " + idType.getIdent().toString());
            System.out.println("Timeline:   " + curVal.getTimeline().getTimestamp().getTime().toString());
            ValueType[] val = curVal.getValue();
            if (val != null)
            {
               for (int j = 0; j < val.length; j++)
               {
                  ValueType cv = val[j];
                  System.out.println("  Vehicle type: " + cv.getVehicle().toString());
                  System.out.println("  Count:        " + cv.getCount());
               }
            }
            System.out.println("State:       " + curVal.getDetectorState());
         }
      }
      return response;
   }
```

3 Soap Client Interface

3.1 Contents of Delivery

The delivery JARs: Realizes External Interface functionality.

- externalinterface.jar
- externaltypes.jar

3.2 Usage of the Soap Client Interface getContentInfo()

```
   /**
    * Read the content info and put the result to standard out
    * @throws RemoteException
    */
   public void getContentInfo() throws RemoteException
   {
      System.out.println("\n-------- Get content info example --------\n");

      GetContentInfoType ctType = new GetContentInfoType(new NMToken("Admin"), "", watchdog);

      GetContentInfoResponseType response = service.getContentInfo(ctType);

      performCommonResponse(response);

      GetContentInfoResponseTypeContentInfoListContentInfo[] ctInfo =
response.getContentInfoList().getContentInfo();
      if (ctInfo == null)
      {
         System.out.println("No content info");
         return;
      }
      for (int i = 0; i < ctInfo.length; i++)
      {
         GetContentInfoResponseTypeContentInfoListContentInfo info = ctInfo[i];
         System.out.println("------------------------------");
         System.out.println("  Object type name: " + info.getObjectTypeName());
         System.out.println("    Uri:    " + info.getObjectTypeURI());
         System.out.println("    Read:   " + info.isReadable());
         System.out.println("    Write:  " + info.isWriteable());
         System.out.println("    Create: " + info.isCreateable());
         System.out.println("    Delete: " + info.isDeleteable());
         System.out.println("    Updatecycle: " + info.getUpdateCycle());
         if (info.getActivateMaxWatchdog() != null)
            System.out.println("    Activate max watchdog: " + info.getActivateMaxWatchdog());
      }
   }
```

3.3 Usage of the Soap Client Interface inquireAll()

```
/**
 * Read the latest TrafficMessage_Incidents
 * @throws RemoteException
 */
public int inquireAllIncidents() throws RemoteException
{
    System.out.println("\n-------- Inquire all example --------\n");

    InquireAllType iqAllType = new InquireAllType(new NMToken("Admin"), "", watchdog,
          new NMToken("TrafficMessage_Incidents"), new FilterList(), null);

    InquireAllResponseType response = service.inquireAll(iqAllType);

    performCommonResponse(response);
    System.out.println("Storetime:  " + response.getStoretime().getTime().toString());
    System.out.println("Last position: " + response.getPosition());

    DataListDs[] ds = response.getDataList().getDs();
    if (ds == null)
    {
       System.out.println("Got no data");
       return response.getPosition().intValue();
    }
    for (int i = 0; i < ds.length; i++)
    {
       System.out.println();
       System.out.println("  Object stored at " + ds[i].getTstore().getTime().toString());
       System.out.println("  Object state: " + ds[i].getObjectState().toString());
       System.out.println("  Identifier: " + ds[i].getIdentifier().getIdent().toString());

       // the data must be of type Incidents TMType
       TMType message = (TMType) ds[i].getData();
       System.out.println("  Message ID:   " + message.getAdmin().getId());
       System.out.println("  Messgae Name: " + message.getAdmin().getName());
    }
    return response.getPosition().intValue();
}
```

Please note that further error processing might be possible using the returned badList and the returned error code.

3.4 Usage of the Soap Client Interface get()

```
/**
* Read the content info and put the result to standard out
* @throws RemoteException
*/
public void getContentInfo() throws RemoteException
{
	System.out.println("\n-------- Get content info example --------\n");

	GetContentInfoType ctType = new GetContentInfoType(new NMToken("Admin"), "", watchdog);

	GetContentInfoResponseType response = service.getContentInfo(ctType);

	performCommonResponse(response);

	GetContentInfoResponseTypeContentInfoListContentInfo[] ctInfo =
		response.getContentInfoList().getContentInfo();
	if (ctInfo == null)
	{
		System.out.println("No content info");
		return;
	}
	for (int i = 0; i < ctInfo.length; i++)
	{
		GetContentInfoResponseTypeContentInfoListContentInfo info = ctInfo[i];
		System.out.println("------------------------------");
		System.out.println("  Object type name: " + info.getObjectTypeName());
		System.out.println("    Uri:    " + info.getObjectTypeURI());
		System.out.println("    Read:   " + info.isReadable());
		System.out.println("    Write:  " + info.isWriteable());
		System.out.println("    Create: " + info.isCreateable());
		System.out.println("    Delete: " + info.isDeleteable());
		System.out.println("    Updatecycle: " + info.getUpdateCycle());
		if (info.getActivateMaxWatchdog() != null)
		System.out.println("    Activate max watchdog: " + info.getActivateMaxWatchdog());
	}
}
```

3.5 Usage of the Soap Client Interface put()

```
/**
 * Put current traffic data to the server
 * @param identifier
 * @param trafficData
 * @throws RemoteException
 */
private void put(CurrentValueType[] trafficData) throws RemoteException
{
    // Build the put list
    PutTypePutListPutds[] putds = new PutTypePutListPutds[trafficData.length];
    for (int i = 0; i < trafficData.length; i++)
    {
        IdentifierType idType = new IdentifierType(null, new NMToken(trafficData[i].getId()));
        PutTypePutListPutds putObj = new PutTypePutListPutds(idType, trafficData[i]);
        putds[i] = putObj;
    }
    PutTypePutList putList = new PutTypePutList(putds);

    // create the put type with user, password, watch dog, object type an put list
    PutType toPut = new PutType(new NMToken("Admin"), "", watchdog,
          new NMToken("TrafficData_detector_currentValue"), putList);

    // execute the put
    PutResponseType response = service.put(toPut);

    performCommonResponse(response);

    PutResponseTypeBadList badList = response.getBadList();
    PutResponseTypeBadListBadds[] badds = badList != null ? badList.getBadds() : null;
    for (int i = 0; i < (badds != null ? badds.length : 0); i++)
    {
        String ident = badds[i].getIdentifier().getIdent().toString();
        System.out.println("Bad object \"" + ident + "\", error " + badds[i].getErrorCode().getValue() +
              ": " + badds[i].getErrorText());
    }
}
```

4 Utilities

The PMBrowser.exe will be used as test facility to test the SoapServerInterface.

Logfiles in the SoapServerInterface can be used to trace each SoapRequest including the contents of the request and the reponse.

5 Test procedure

First Step:
Tracing of Logfiles. The Logfiles will be supplied to the communication partner. The communication partner verifies the logfiles for correctness.

Second Step:
Connection of client and server via LAN. Working with test data.

Third Step:
Test under real time conditions.