REST Web Services in XI (Proof of Concept)

I wondered whether it is possible to use REST Web Services in SAP PI (7.0) and, if so, what the required effort would be.

A simple scenario I chose for this purpose covers consuming a RESTful Web Service exposed by third party and re-publishing it as another RESTful Web Service from XI as a proof of concept. The diagram below illustrates the idea:

For the third party component, I have chosen to create a REST web service in an SAP ERP system following the logic suggested in an excellent weblog [Real Web Services with REST and ICF](#).

As there is obviously no native XI REST-interface, for the XI part I used plain HTTP in- and outbound adapter classes (ABAP OO) as a basis for the solution. For some reasons (more on this further below) it is not possible to use HTTP adapter classes in XI as they are.

The scenario described refers to the PI version 7.0. However, due to the fact that the XI HTTP adapters have hardly, if at all, changed since XI version 3.0, it may well be considered cross-version.

Step 1. Create REST Web Service in ERP

For the purpose of the demo I created a simple REST service in SAP ERP responding to two kinds of REST requests:

1. http://<custom icf adapter url>/object - Request the list of available objects
2. http://<custom icf adapter url>/object/object_id - Request details of the given object

Following the weblog [Real Web Services with REST and ICF](#), I have created a custom node in ICF and a custom handler class. Within the handler I used simple Bridge pattern logic with two dimensions: object (e.g., Business Partner, Purchase Order and so on) and method (READ, CREATE, CHANGE, DELETE). In a real world scenario it would make more sense to implement a Type Object pattern in order to provide for a better request flexibility. For the demo purposes, however, a simpler design was justified.

Step 2. Outbound REST Adapter in PI (Consume REST Web Service)

An outgoing HTTP request in PI is handled by a class CL_HTTP_PLAIN_OUTBOUND. There are two approaches to "teach" PI speak REST language here:

Option 1. Dynamically set HTTP Request properties within XI.

Apparently, at least some properties of the outgoing HTTP-Request may be set programmatically from within an integration scenario. This can be derived from the contents of the method CL_HTTP_PLAIN_OUTBOUND->SET_HEADER_FIELDS:
LOOP AT fields INTO wa_fields.
CASE wa_fields-name.
  WHEN '-request_method'. "#EC NOTEXT
  WHEN '-server_protocol'. "#EC NOTEXT
  WHEN 'Content-Type'. "#EC NOTEXT
  WHEN 'Authorization'. "#EC NOTEXT
  WHEN 'Content-Length'. "#EC NOTEXT
  WHEN OTHERS.
    CALL METHOD client->request->set_header_field
    EXPORTING
      name  = wa_fields-name
      value = wa_fields-value.
    m_trc3 wa_fields-name wa_fields-value.
ENDCASE.
ENDLOOP.

Also, in the method CL_HTTP_PLAIN_OUTBOUND->USE_HEADER_FIELDS_FROM_DY further on, some coding may be found that refers to setting HTTP header fields dynamically:

*first header
lv_attr_value = channel->get_attribute( co_field_1 ).
IF lv_attr_value IS NOT INITIAL.
  CALL METHOD dy->if_xms_msghdr30_dynamic~get_record
  EXPORTING
    im_namespace  = co_namespace
    im_name       = co_h_field_1
  RECEIVING
    re_dyn_conf_record = lv_dynf_conf_record.
  IF lv_dynf_conf_record IS NOT INITIAL.
    lv_param_value = lv_dynf_conf_record-param_value.
    CALL METHOD client->request->set_header_field
    EXPORTING
      name  = lv_attr_value
      value = lv_param_value.
  ELSEIF lv_dynf_conf_record IS INITIAL AND channel->get_attribute( co_dyn_conf_fail_on_miss_props ) = '1'.
    RAISE EXCEPTION TYPE cx_xms_syserr_plainhttp
    EXPORTING
      id = cx_xms_syserr_plainhttp=>co_id_attribute_missing_params.
  ENDIF.
  CLEAR: lv_attr_value, lv_param_value, lvDynf_conf_record.
ENDIF.

Unfortunately, to the best of my knowledge it is not possible to form the URL dynamically so the question if it is possible to use the regular class CL_HTTP_PLAIN_OUTBOUND without any modifications must remain open at this time.

Option 2. Enhance the class CL_HTTP_PLAIN_OUTBOUND
Instead, I decided to approach the problem on a lower level using implicit enhancements so that the class CL_HTTP_PLAIN_OUTBOUND can formulate requests in the REST format straight away.

Enhanced version of the class uses most of the original coding. However, I have included the following logic into request formation:

1. An outgoing message (request to the REST Web Service) is read using provided interface IF_XMS_MESSAGE_XMB
2. The REST URL is formed using fields OBJECT and OBJECT_ID from the outgoing message.
3. The HTTP method is modified from the default POST to the request-specific based upon a message field ACTION.

Obviously, the outgoing message type has to include all the fields listed above as a part of the payload.

Theoretically, there is another option of creating own adapter configuration on the basis of the existing HTTP one. This, in fact, would be the cleanest solution.

**Step 3. Inbound REST Adapter in PI (Publish REST Web Service)**

Due to the fact that the XI inbound plain HTTP adapter performs multiple validation activities on the URL, it is not possible at all to use the original handler class HTTP_PLAIN_INBOUND.
However, it is also not necessary as this adapter is nothing different than a node in the ICS that can be copied and reused, to some extent at least. What needs to be done in the custom implementation is:

1. Retrieval of the OBJECT, OBJECT_ID and ACTION from the requested URL and HTTP method

   ```
   ***Identifiziere Aktion
   l_var_action = server->request->get_header_field(name = '~request_method').
   *** Identifiziere Objekt
   l_var_path_info = server->request->get_header_field(name = '~path_info').
   SHIFT l_var_path_info LEFT BY 1 PLACES.
   SPLIT l_var_path_info AT '/'
   INTO
   l_var_object
   l_var_object_id.
   ```

2. Retrieval of the HTTP body (which will go into the DATA field in the XI message)

   ```
   *** Extrahiere REST-Anfragedaten
   l_var_rest_request = server->request->get_cdata().
   ```

3. Putting together retrieved fields conform to the XI message data type

   ```
   *** XI-Nachricht vorbereiten
   l_var_xi_msg_body = l_rif_interpreter->prepare_xi_message_body(
   i_var_object = l_var_object
   i_var_object_id = l_var_object_id
   i_var_action = l_var_action
   i_var_data = l_var_rest_request
   ).
   ```

4. Determination of the XI scenario data (service, interface and namespace) e.g. based upon configuration (for the demo purposes hard coded)
Also, (copied) original handler code must be modified so that request data are set by our custom logic and not retrieved directly from the request.

**Step 3. Configure Integration Scenario in PI**

I defined a simple synchonic scenario with a request message being routed to the REST service provider. The type of the request message with the fields OBJECT, OBJECT_ID, ACTION for the REST request and DATA for the actual REST payload:
The type of the response message defined as an xsd string:
Step 4. Enjoy

The business case proved itself doable. However, at the first glance massive changes to the PI low-level components are required.

Just after having finished the demo I found out about AXIS framework that is apparently capable of natively handling REST Web Services and also can be integrated into NetWeaver landscape. Maybe this would offer a more feasible option for using REST services in the SAP world?