PUBLIC

Integrating Custom Client with DOE

SAP® NetWeaver Mobile 7.1

Target Audience

- Techonology consultants
- Software Developers

Document version: 1.1 – June, 2009

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options.
	Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, F2 or ENTER.

Icons

Icon	Meaning
	Caution
	Note / Important
	Example
	Recommendation / Tip

Change History

Version	*Comments*
1.0	First release of the guide
1.1	Correction in Section 5.1 - Registration

Table of Contents

1. Business Scenario

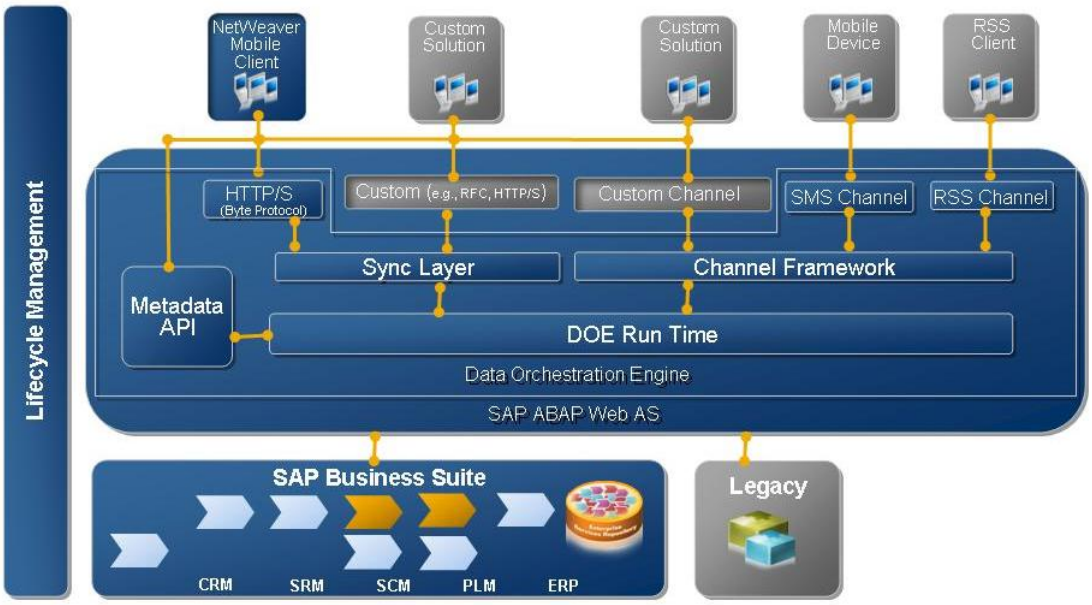
SAP NetWeaver Mobile 7.1 comprises of Data Orchestration Engine (DOE) and mobile clients. The mobile client shipped by SAP supports a limited set of receivers. This list is available in the Product Availability Matrix. However, there are devices based on numerous platforms such as Windows Mobile, RIM, Symbian, Linux, Palm, iPhone, Android, and so on which provide a rich native user experience. To allow customers and partners to choose any client technology that suits their business requirements, we have provided an interface on the DOE that allows you to integrate any client technology with SAP NetWeaver Mobile. This enables any client technology to leverage middleware capabilities such as data distribution, data integrity, device management, scalability and so on.

2. Introduction

Data Orchestration Engine (DOE) is a message-oriented middleware, which is capable of staging data from various back-end systems and distributing this data to different receivers based upon their preferences. The DOE also enables bi-directional asynchronous data exchange with guaranteed data integrity. It enables synchronization of business data with receivers from various back-end systems in a scalable way. The DOE is not only a high scalable synchronization middleware for applications built on client technology shipped by SAP, but is also open for integration with other development environments.

The lifecycle activities for receivers such as registration, application deployment, device recovery, and so on, are handled by SAP NetWeaver Mobile for receivers based on the SAP NetWeaver Mobile Client. For receivers based on other technologies/platforms, the lifecycle management activities have to be handled jointly by the DOE and the custom solution. This in described in detail in the following sections.

The architecture of DOE from the *openness* perspective is outlined in the diagram below:



Data Object Message Exchange

The data object message exchange between DOE and receivers can take place through the Sync Layer or Channel Framework.

- ### Sync Layer

  The Sync Layer contains a set of APIs exposed by DOE for data object message exchange. The APIs and message formats are described in the following sections. The custom client can exchange messages using the APIs in Sync Layer. The SAP NetWeaver Mobile Client uses the Sync Layer and HTTP/S for message exchange.

  This document explains how to develop a custom client using the sync layer.

- ### Channel Framework

  Data object message exchange can also take place through channel framework. The channels which are built on top of the channel framework can work on any messaging formats. SAP NetWeaver Mobile also ships SMS and RSS channels which send out data object messages in SMS and RSS formats respectively.

3. Prerequisites

- Development experience in ABAP Language
- Knowledge of SAP NetWeaver Mobile architecture
- Knowledge of SAP Web AS's web server capabilities, especially, use of Internet Communication Framework (ICF) Services

4. Meta Data APIs

You need to know the definition of data objects to build a client application. The following meta-data APIs can be used to understand the definition of Data Objects. These APIs are RFC enabled ABAP function modules.

4.1 MSB_GET_LIST_OF_SCV

This API is used to get the list of software component versions (SWCV) available in DOE. If a caption (name) is specified, the API returns the matching SWCV. If '*' is specified, it returns all SWCV in the landscape.

Parameter Name	Type	Description
CAPTION	Importing	Software component version caption (Name)
SCV_GUID	Importing	Software component version ID
BASIS_SCV	Importing	Flag - If set to 'X', selects only Basis SWCV
ET_SCV	Table of type MSB_SCV	List of SWCVs

Note

Do not provide any values to other optional parameters.

4.1.1 MSB_SCV

Field Name	Description
SCV_GUID	Unique identifier for the SWCV.
CAPTION	Caption of SWCV. It is a concatenation of Name, Version and Vendor fields
NAME	Name of the SWCV
VERSION	Version of the SWCV
VENDOR	Vendor
RELFORSHIP	Released for shipment. Once SWCV is released for shipment, only compatible changes to the data objects can be made
IS_MMW_BASIS	Flag to indicate if the SWCV is a Basis SWCV -> "SAP BASIS 7.10" SWCV
IS_BWCMPATIBLE	The SWCV works in backward compatible mode to support SAP MI 2.5 applications on NW Mobile 7.10

Note

Other fields of the structure are either self explanatory or can be ignored.

4.2 MSB_GET_LIST_OF_MBO

This API is used to get the list of data objects for a given software component version.

Parameter Name	Type	Description
SCV_GUID	Importing	Software component version ID. Can be obtained from the previous API, MSB_GET_LIST_OF_SCV
E_BDATA_MODEL	Importing	Setting this flag to X will prevent device local, logs, generic sync data objects to reach the client. Use this only to filter out the above mentioned objects
E_GET_INHERITED_DO	Importing	To get the data objects present in all the inherited SWCV of this SWCV
ET_MBO	Table of type MSB_MBO_HDR	List of data objects

Note

Do not provide any values to other optional parameters.

4.2.1 MSB_MBO_HDR

Field Name	Description
MBO_NAME	Data Object Name
MBO_TYPE	Data Object Category Possible values are STD, REG, SUB, MON. Only STD and MON are applicable on the client
MBO_ID	Unique identifier for a data object
DIRECTION	Direction of message flow. Values: Upload, download or bi-directional
MBO_TRANS	Unique identifier for a data object version
VERSION	Data Object Version
ACTIVE	Data object version's runtime is ready for use
SWCV_NAME	Unique identifier for the SWCV

Note

Other fields of the structure are either self explanatory or can be ignored.

4.3 MMW_DDIC_IMPORT_BAPI_WRAPPER

This API is used to get the back-end structure information for the nodes and node attributes. Typically, this API is used to get the labels of fields for UI development.

Parameter Name	Type	Description
I_SCV_GUID	Importing	Software component version ID
I_MBO_NAME	Importing	Data Object name
ET_MMW_BE_FIELD	Table	Backend fields for this data object
ET_DD07V	Table	View on fixed values and domain texts
ET_DDFIELDS	Table	Field information like domain and so on

Exceptions

Exception	Description
BE_DEST_NOT_FOUND	Unable to find the specified backend destination
BE_CONNECTION_FAILURE	Unable to connect to the backend
BE_DTEL_NOT_FOUND	Data element not found
BE_DOMA_NOT_FOUND	Domain not found
BE_TABLE_NOT_FOUND	Table not found
BE_FIELD_NOT_FOUND	Unable to find the field
MMW_MBO_NOT_FOUND	Unable to find the data object
MMW_NODE_NOT_FOUND	Unable to find the data object node
MMW_NODE_ATTR_NOT_FOUND	Unable to find the node attribute
BE_ADAPTER_NOT_FOUND	Unable to find the backend adapter

Note

Do not provide any values to other optional parameters.

4.4 MSB_GET_NODE_INFO

This function module is used to get the node information for a given data object.

Parameter Name	Type	Description
SCV_GUID	Importing	Software component version id
MBO_NAME	Importing	Data object name
IS_CLIENT	Importing	Client specific or not
ET_NODE	Table Type MSB_NODE	List of nodes for this data object

Note

Do not provide any values to the other optional parameters.

4.4.1 MSB_NODE

Field Name	Description
NODE_NAME	Node Name
HIERARCHY	Hierarchy of the node in the data object tree.
PNODE_NAME	Name of the parent node.

Note

Other fields of the structure are either self descriptors or can be ignored.

4.5 MSB_GET_NODE_ATT_INFO

This function module is used to get all the attributes and their information like data type, length, generated names etc for a given node

Parameter Name	Type	Description
SCV_GUID	Importing	Software component version id
MBO_NAME	Importing	Data object name
NODE_NAME	Importing	Node name
ET_NODE_ATT	Table type MSB_NODE_ATT	Table of node attributes

Note

Do not provide any values to other optional parameters. Other fields of the structure are either self explanatory or can be ignored.

4.5.1 MSB_NODE_ATT

Field Name	Description
ATTR_NAME	Attribute name
ATTR_POS	Position of attribute in the node

LOWERCASE	Attribute value is case sensitive
IS_SYNCKEY	The attribute is Sync Key Synckey is the primary key of a node. It is a field of 32 characters, which is a globally unique identifier
IS_PKEY	Attribute value refers to Sync Key of the parent. Referential integrity between parent and child instance is maintained through this attribute
IS_BEKEY	The attribute is a backend (application) key field defined by application developers
IS_BEFIELD	The attribute is a backend (application) field defined by application developers
IS_TMEMO	The attribute is a text memo field. Text memo field represents un-restricted length character field. It is used to store text files. This is equivalent to CLOB.
IS_BMEMO	The attribute is a binary memo field. Binary memo fields are equivalent to BLOB and can be used to store any type of files.
GRP_FIELD	Used to group fields like Currency-Currency Key, Quantity-Units, etc.
GEN_NUM	The generated alpha-numeric alias for the attribute name. Used in all data object message exchange with DOE.
P_BEKEY	The attribute refers to the backend key of the parent node.

 Note

Other fields of the structure are either self explanatory or can be ignored.

4.6 MSB_GET_ASCN_INFO

You can obtain information about referential integrity (foreign key) definition between two data objects using this API.

Parameter Name	Type	Description
SCV_GUID	Importing	Software component version id
MBO_NAME	Importing	Data object name
NODE_NAME	Importing	Node name
ASCN_KEY	Importing	Association key.
MBO_TRANS	Importing	Data object trans id
IS_PARTIAL_ASSO	Importing	Set to 'X' to get partial association also
ET_ASCN	Table of type MSBIN_ASCN	Table of associations
ET_FIELD_MAP	Table of type MSBIN_ASCN_FLD	Field mapping between the data object nodes

ET_LINK	Table of type MSBIN_ASCN_LINK	Link information for this association

Note

Do not provide any values to the other optional parameters.

Exception

COULD_NOT_GET_INFO – This exception is raised when the API is unable to fetch the relevant information.

4.6.1 MSBIN_ASCN

Field Name	Description
NODE_NAME	Referring Node Name (current node)
ASCN_KEY	Name of the association
ASCTD_SWCV	SWCV of the referred data object
ASCTD_MBO	Referred data object
ASCTD_NODE	Referred data object node (always root node of the referred data object)
ASCTD_MBO_TRANS	Unique identifier for the referred data object version
ASCN_TYPE	Association type: Complete or Partial association. Values C and P respectively

 Note

Other fields of the structure are either self explanatory or can be ignored.

4.6.2 MSBIN _ASCN_FLD

Field Name	Description
NODE_NAME	Referring Node Name (current node)
ASCN_KEY	Name of the association
ATTR_NAME	Name of the referring attribute
ASCTD_ATTR	Name of the referred attribute
KEYTYPE	Values: B (Backend Key), S (Synckey)
GNAME	Generated alpha-numeric alias of the referring attribute
ASCTD_GNAME	Generated alpha-numeric alias of the referred attribute
MBO_NAME	Referring Data Object name
SWCV_NAME	SWCV of the referring data object

 Note

Other fields of the structure are either self explanatory or can be ignored.

4.6.3 MSBIN_ASCN_LINK

Field Name	Description
NODE_NAME	Referring Node Name (current node)
ASCN_KEY	Name of the association
LINK_FLD	Name of the link attribute
LINK_VAL	Defined value for the link attribute
GEN_NAME	Generated alpha-numeric alias of the link attribute

Note

Other fields of the structure are either self explanatory or can be ignored.

5. Sync Layer

The APIs of the Sync Layer are RFC enabled ABAP Function Modules. These can be accessed by external client application via RFC or HTTP(s) protocols. The recommended approach is to implement a HTTP Handler in an ICF service. The HTTP Handler then delegates the HTTP(s) requests from the receiver to the Sync Layer RFC APIs.

5.1 Registration

Before a receiver can exchange data object messages with DOE, it has to register itself with the DOE. The prerequisites are:

- Logical devices with relevant values assigned to device attributes are available on the DOE. For more information on creating logical devices, refer to device administration.
- Device attributes for registration are defined in the transaction SDOE_DEV_REGISTER are defined. The user can choose any attribute as a registration parameter depending on his business need.
- There are two additional parameters that have to be part of the registration parameters apart from customer-defined mandatory parameters.
    - Physical ID: It is a 32 character field which uniquely represents a physical device. The receiver should send the Physical ID and Device ID to the DOE in all message exchanges. The physical ID-device ID combination is used by DOE to validate the physical device.
    - User: This should be a valid user in the SAP Web AS where DOE is installed.

From the physical receiver, call the REGISTER_DEVICE API. The signature of the REGISTER_DEVICE API is as follows:

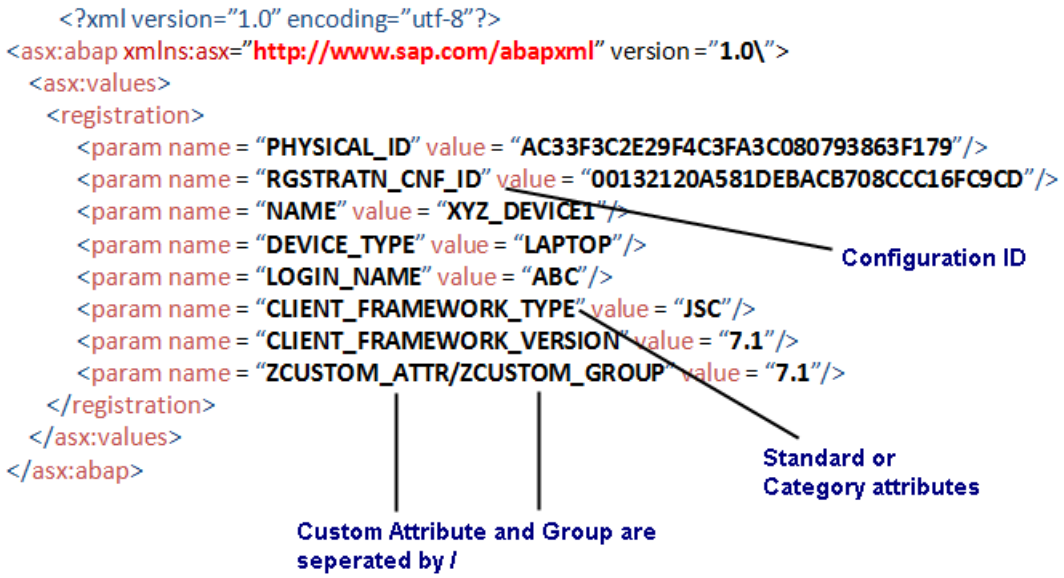
Parameter Name	Type	Description
XML_SOURCE	Importing	This parameter contains the xml source for registration.
DEVICE_ID	Exporting	If registration is successful, then device id of the logical device is sent to the client via this parameter.
EV_X_XMLD	Exporting	This field contains the XML response to the registration command
REGISTERED	Exporting	This field is used to convey the registration status

Exceptions:

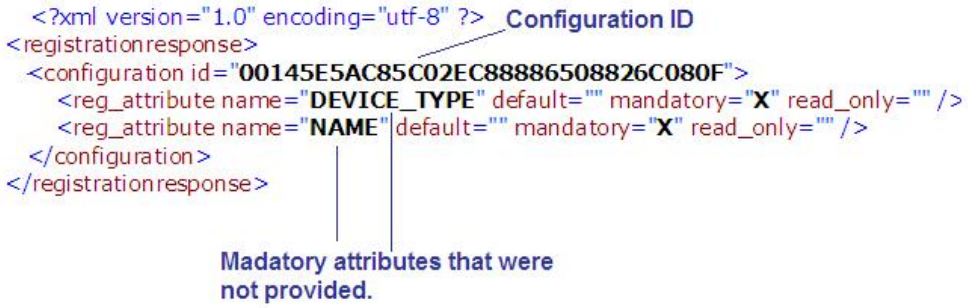
Exception	Description	Client Reaction
REGISTRATION_ERROR	This is raised when DOE is not able to find a suitable device as per the criteria specified	Check for enabled unregistered devices. If there are devices, check if the criteria specified matches to the device. If there are no devices create the device and enable it

ERROR_OTHERS	General exception	
REG_CLNT_SERVER_CONFIG_DIFFER	This is raised when there is configuration mismatch between the DOE and the mobile device. Configuration mismatch arises when certain fields are marked as mandatory for registration but their values are not sent or the values that are sent are improper.	When this exception is raised, connecting again with the configuration id provided by DOE and the correct configuration value will solve the problem. The configuration ID can be obtained from the response XML (EV_X_XMLD) or from the data base table SMMW_DEV_REGISTR in DOE.

A sample xml source is shown below:



A sample response XML if the registration fails is shown below:



5.2 Quality of Service

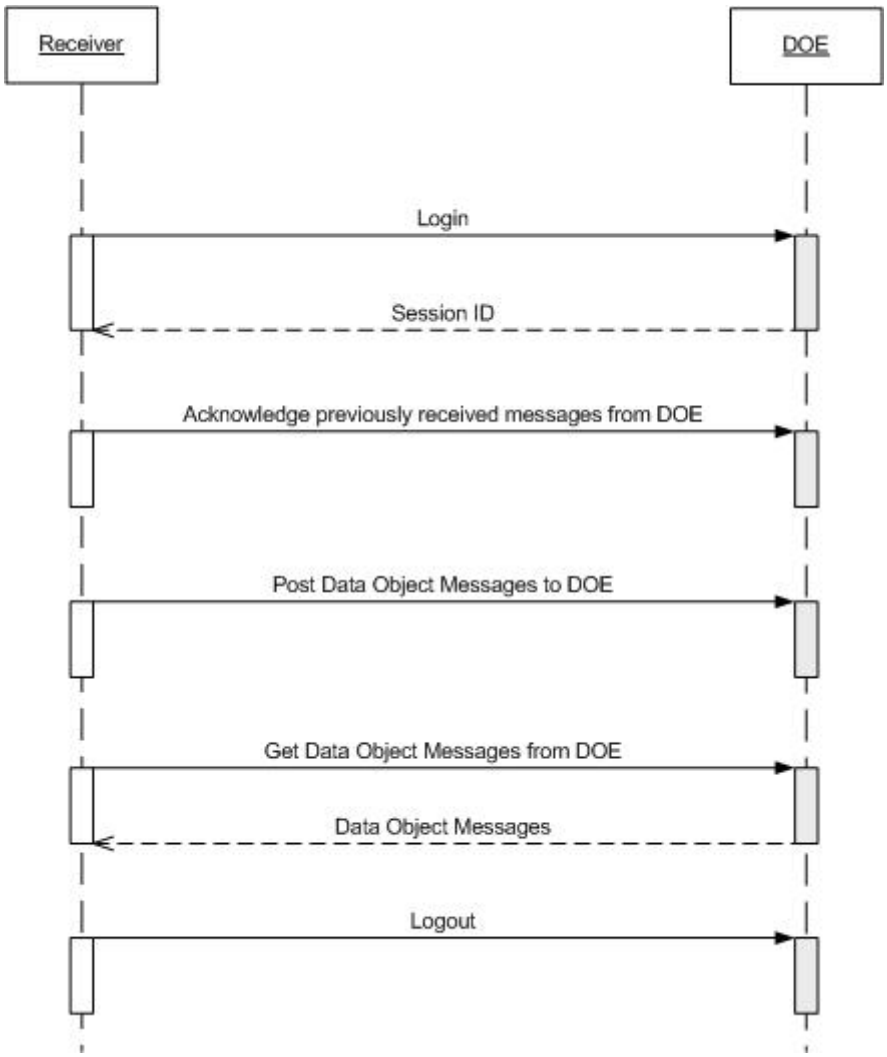
DOE guarantees *exactly once* and *in-order* delivery of messages. However messages may get network drop issues, packet loss issues, or other exceptions while communicating with DOE and this can impact message delivery.

To ensure that no messages are lost and are delivered in-order, every message should be sequentially numbered. These sequential numbers are called message IDs.

DOE assign each message a number (in sequence order) and this number can be used on the receiver to make sure that no messages are lost.

5.3 Message Exchange

The data object messages are exchanged between DOE and the receiver as shown in the Sequence Diagram below. The custom software on the receiver should follow the same procedure. All the calls are synchronous.



5.4 Login

You use RFC enabled function module MMW_LOGIN, to login to the DOE.

A Session ID is returned on successful login. This login is in addition to the ABAP Web AS login.

The parameters of this function module are given below:

Parameter Name	Type	Use
DEVICE_ID	Importing	This contains the device id of the logical device in DOE. This is obtained during registration process
PHYSICAL_ID	Importing	Physical id of the mobile device. The combination of physical id – device id is used to validate the mobile device
TIMEOUT	Importing	Wait time in case of no response before disconnecting
SESSIONID	Exporting	Unique identifier created for every successful login to DOE. This will be used to identify the devices during further interaction

 Note

All the other parameters are optional and can be ignored.

Exceptions

Exception	Description	Client Reaction
ERR_USER_AUTHORIZATION	This is raised when there is a mismatch in the physical id – device id combination sent by the mobile device	Check for the physical id-device id sent from the device.
ERR_SESSION_CREATION	If DOE is unable to create a session id, then this exception is raised	This could be due to system issues. Try to login again.

5.5 Acknowledgement

Use RFC enabled function module MMW_CONFIRM_MSG_RCVD to acknowledge the previously received messages from DOE.

Parameter Name	Type	Use
SESSION_ID	Importing	Unique identifier created during login to DOE.

LAST_READ_MSG_ID	Importing	Indicated last received message ID from DOE. DOE deletes all the messages received and confirmed by the receiver. All the messages are retained until the receiver acknowledges.

Note

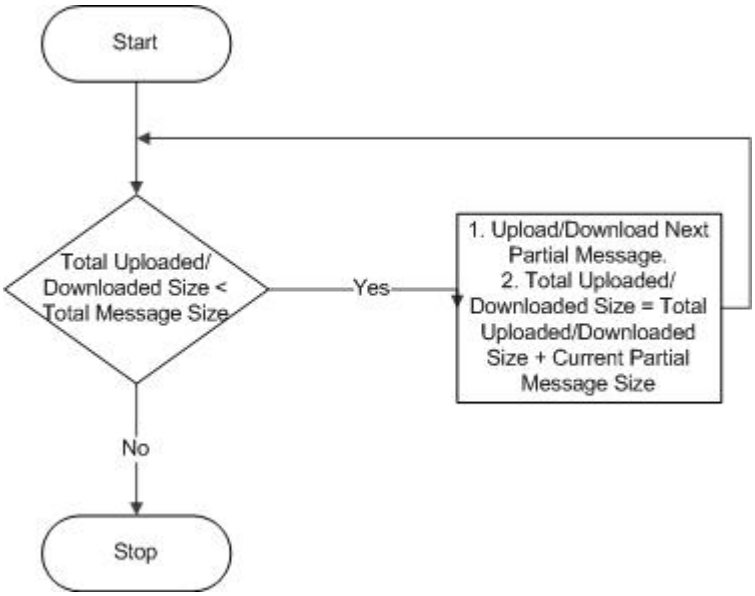
All the other parameters are optional and can be ignored.

Exceptions

Exception	Description	Client Reaction
ERR_SESSION_INVALID	This exception is raised when the session id passed is invalid	Please check the session id obtained during login. If the session id is lost or empty, try login again
ERR_SESSION_EXPIRED	This exception is raised when the session has timed out and the device is still trying to use the old session id	The session would have expired because of system being slow or no operation performed for a specified period of time mentioned in the timeout parameter. To recreate a new session, login again
ERROR_INVALID_MSG_ID	This exception is raised when the message id sent by the device for confirmation is invalid	Check the last received message id.
ERR_LDQ_QUEUE_IN_USE	When the outbound queue is already used by another process, this exception is raised	Retry again later on in next sync.
ERR_INVALID_SEQUENCE_NO	When the sequence number of message to be read from the LDQ is incorrect, this exception is raised	Check the last received message id.

5.6 Partial Messages

If the message size is too big, it can be broken down into number of smaller messages and exchanged. The logic is same for uploading messages to DOE and downloading messages from DOE as shown in the flowchart below:



The partial messages have to be uploaded /downloaded consecutively and in-order. The unit of message size is the number of characters.

5.7 Post Messages

This module is used to upload messages from client to DOE. The message from client should be converted to the format understood by DOE before calling this function module. For more information on message formats, refer to message format.

Parameter Name	Type	Use
SESSION_ID	Importing	Unique identifier created during login to DOE.
IN_MESSAGE_DETAILS	Importing	This is a table which contains the message contents to be uploaded to DOE. The parameter is table of type SMMW_MSG_DETAILS.
OUT_MESSAGE_ID	Exporting	This gives information about the message id of the last uploaded message to DOE. This information can be used to ensure Quality of Service.
OUT_POS	Exporting	Total size of a message uploaded so far. Particularly useful while uploading partial messages as described in the diagram above.

Exceptions

Exception	Description	Client Reaction
ERR_SESSION_INVALID	This is raised when the session id sent by the client is invalid	Please check the session id obtained during login. If the session id is lost or empty, try login again

ERR_SESSION_EXPIRED	This exception is raised when the session has timed out and the device is still trying to use the old session id	The session would have expired because of system being slow or no operation performed for a specified period of time mentioned in the timeout parameter. To recreate a new session, login again
ERR_PROCESSING_ERROR	This exception is raised when there is a message id mismatch i.e. the message id expected by the server and the one sent by the client are different	Check for the last message id sent and last received message id.
ERR_OTHERS	General exception	

5.7.1 SMMW_MSG_DETAILS

Field Name	Description
MBO_TRANS	Data Object Version Identifier
MSG_ID	A sequence number for message identification used for guaranteeing quality of service. If a message is split into number of partial messages MSG_ID should be same for all the partial messages.
MSG_TYPE	Type of Message. Explained in detail in the Message Types Section.
MSG_START_POS	This specifies the start position of a partial message in the complete message.
TOT_MSG_LENGTH	Total length of the message.
MSG_CONTENTS	Actual XML document containing the data object instance(s).
SENDING_MSG_LEN	Message length sent to/received from DOE. This will be different from total length of message in case of partial messages
SYNC_KEY	Unique identifier for every data object instance. The value is the value of the root node Sync Key of the data object instance.

5.8 Get Messages

For downloading (Getting) the message from DOE, use the RFC enabled function module MMW_GET_MESSAGES.

When this function module is executed without passing any optional parameters, it will return the number of messages yet to be downloaded by receiver.

Parameter Name	Type	Use
SESSION_ID	Importing	Unique identifier created during login to DOE.

IN_MSGCOUNT	Importing	Number of messages requested by the device
IN_MSGSIZE	Importing	Size of messages requested by the device. If both message count and size are 0, no processing happens
IN_LAST_READ_MID	Importing	Message id of the last read message.
IN_LAST_READ_POS	Importing	Position of the last read message. Used in the case of partial message
IN_CONFIRM	Importing	This specifies whether DOE can confirm all the messages sent so far. To confirm, set the value to 'X'
OUT_MSGCOUNT	Exporting	Number of messages sent by DOE
OUT_MSGSIZE	Exporting	Size of the messages sent by DOE
OUT_TOTAL_TIME	Exporting	Time taken for processing the request
OUT_LAST_READ_MSG_ID	Exporting	Message id of the last sent message
OUT_LAST_READ_POS	Exporting	Position of the last sent message. This is used in case of partial message
OUT_MSGCOUNT_WAITING	Exporting	Number of message waiting in the outbound queue
OUT_MSGSIZE_WAITING	Exporting	Size of the message waiting in the queue
OUT_MESSAGE_DETAILS	Exporting	This is a table which contains the message contents sent by DOE. The parameter is table of type SMMW_MSG_DETAILS

Exceptions:

Exception	Description	Client Reaction
ERR_SESSION_INVALID	This exception is raised when the session id passed is invalid	Please check the session id obtained during login. If the session id is lost or empty, try login again

ERR_SESSION_EXPIRED	This exception is raised when the session has timed out and the device is still trying to use the old session id	The session would have expired because of system being slow or no operation performed for a specified period of time mentioned in the timeout parameter. To recreate a new session, login again
ERR_LDQ_QUEUE_IN_USE	When the outbound queue is already used by another process, this exception is raised	
ERR_INVALID_SEQUENCE_NO	When the sequence number of message to be read from the LDQ is incorrect, this exception is raised	
ERR_OTHERS	General exception	

5.9 Logout

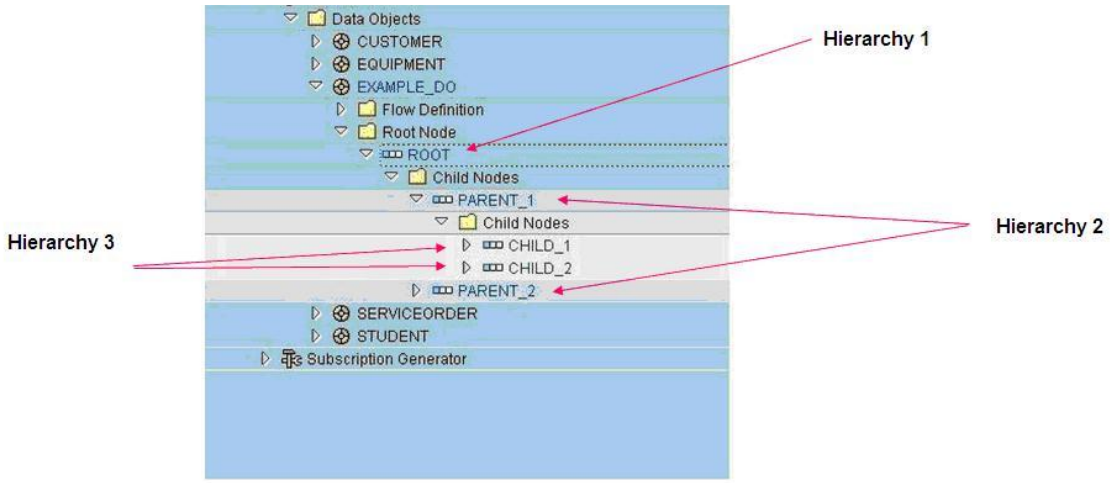
You use the RFC enabled function module MMW_LOGOUT, to log out or terminate an existing session.

The parameters of this module are given below:

Parameter Name	**Type**	**Use**
SESSION_ID	Importing	Unique identifier created during login to DOE. This is used to identify which session needs to be terminated
END_TIME	Importing	Session end time
END_DATE	Importing	Session end date

6. Message Format

DOE uses XML format to exchange messages. UTF-8 is the encoding format. The XML message format is explained based on the data object shown below:



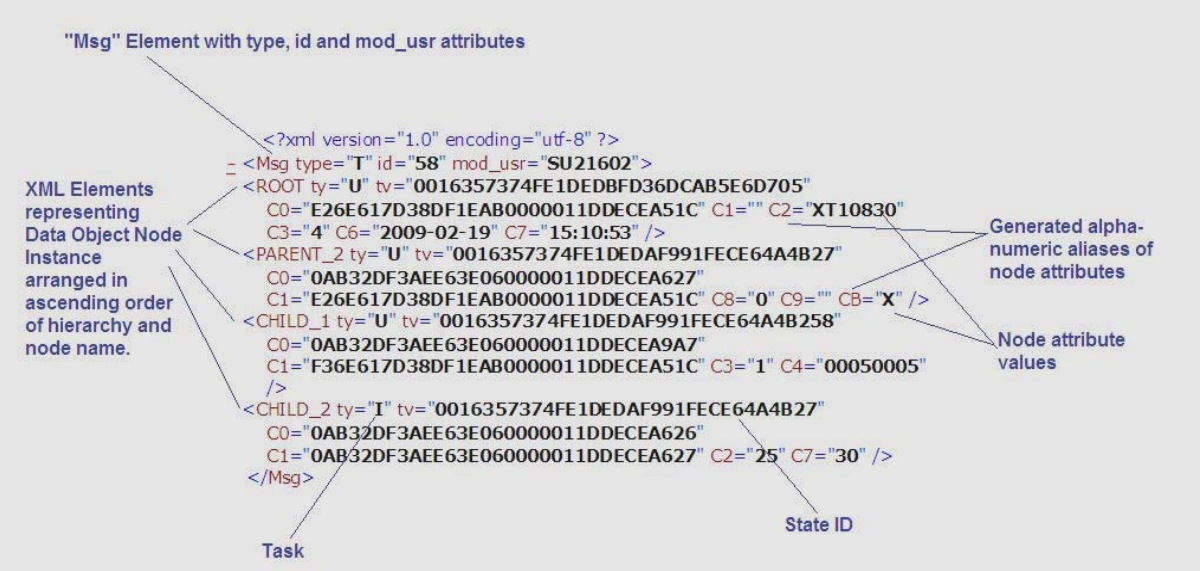
The EXAMPLE_DO has 5 nodes – a root node at hierarchy level 1, two nodes at hierarchy level 2 and two nodes at hierarchy level 3.

The XML document has to following the following rules:

- Each message has the top level XML element named “Msg”. The attributes of this element are:
  - Type: Represents message type. For more information, refer to Message Types.
  - Id: An ID sent from the receiver in a transactional message and echoed back by DOE during confirmation and rejection of that message. It is a character field of length 32. This Id can be used to by receiver to identity the original transactional message.
  - Mod_usr: Creator of the message. Should be a valid SAP Web AS user.
- Each node instance is represented as a XML element and the node attributes as attributes of the XML element. Only those attributes which have to be updated will be available in the node element. This is done to optimize the bandwidth utilization. It is recommended that the receiver too sends only the changes attributes.
- The XML elements representing the data object node instances are arranged in ascending order of hierarchy level and node name.
- Each XML element representing the data object node instance have following additional attributes:
  - BEF_IMG (tx) – If set to 1, the attribute values of the node instance is the image of the node instance in DOE. The node instance with BEF_IMG set to 1 will available only if a message is rejected by DOE. The rejection by DOE is intimated to receiver via the rejection message.
  - TASK (ty) – action to be performed on this node instance. It can have values:
    - I for Insert
    - U for Update
    - D for Delete
    - N for No change

- STATE_ID (tv) – Used for conflict detection. The state id received from DOE has to be echoed back to DOE every time the instance is changed and uploaded. The receiver should not create or change the value of this field.

- All the node attributes are represented by the generated alpha-numeric alias.



7. Message Types

Message Type	Value of the "type" attribute in the "Msg" element of the XML message	Direction
Transactional	T	Receiver to DOE
Confirmation	C	DOE to Receiver
Rejection	E	DOE to Receiver
Import	I	DOE to Receiver
Bulk	B	DOE to Receiver
Zap	Z	DOE to Receiver
Current State	2	DOE to Receiver

7.1 Transactional Message

- A data object message instance created, updated or deleted on the client and uploaded to DOE is called transactional message.
- Each transactional message should always have exactly one data object instance.
- Only the changed attributes or node instances need to be uploaded. Unchanged attributes need not form part of the message.
- The transactional messages are processed asynchronously in the DOE.
- The business acknowledgement of this message is sent separately as confirmation or rejection message from DOE. The business acknowledgement is different from the message acknowledgement described in the Quality of Service section. The business acknowledgment is validation of the message from business perspective and it is application specific. Whereas, the message acknowledgement, is a technical acknowledgement of the message received by DOE.

7.2 Confirmation

- A confirmation message is a business acknowledgement of a transactional message indicating that the message was successfully processed by DOE and the backend.
- The confirmation message may have further updates to the data object instance. The updates may include, for example, addition of a new child node instance, removal of a child node instance, change of a node attribute value, etc. These updates are expected to be applied on the local data base on the receiver so that the data object instance is in sync with the DOE.

7.3 Rejection

- A rejection message is a business acknowledgement of a transactional message indicating that the message was not processed successfully by DOE or the backend.
- The rejection message will have the image of the transactional message plus image of the node instances in DOE. Only the image of those node instances which are sent in the original transactional message will be part of the rejection message. The DOE image of node instances can be identified by value of the BEF_IMG attribute of the node element in the XML message. The DOE image of node instances will have BEF_IMG value as "1".

- The rejection message will also contain the reason for rejection. The reason for rejection can be understood from the XML element SMMW_ERRS_LIST whose attributes and meaning is as follows:

  Important

  Please do not confuse the term message used here with the data object message. The message here means the error text.

Field	Alias	Format	Length	Description
ERR_ID	A	CHAR	32	GUID – Representing the record in this table with a unique identifier
NODE_ERR	B	INT1	3	Segment or transaction error (0=transaction;1=segment). If the error has occurred due to some inconsistency in a segment data (i.e. node data) the value is 1.
ERR_TSTMP	C	INT4	10	Time Stamp representing when the error was logged in the system
ERR_KEY	D	CHAR	32	Primary key of segment row, which caused the error (i.e. the SYNCKEY of the segment) if the NODE_ERR is 1
ERR_INFO	E	CHAR	255	Additional error Information compiled by DOE
MSG_TYP	F	CHAR	1	Message type: S Success, E Error, W Warning, I Info, A Abort
MSG_ID	G	CHAR	20	Message Class: A logical group of application messages, relevant only in ABAP stack
NUMBER	H	NUMC	3	Message Number: The actual number of the message in the Message Class. Again relevant for ABAP stack
MESSAGE	I	CHAR	220	Message Text: Actual error text
LOG_NO	J	CHAR	20	Application log: log number
LOG_MSG_NO	K	NUMC	6	Application log: Internal message serial number
MESSAGE_V1	L	CHAR	50	Message Variable1. Equivalent to ABAP's sy-msgv1
MESSAGE_V2	M	CHAR	50	Message Variable2. Equivalent to ABAP's sy-msgv2
MESSAGE_V3	N	CHAR	50	Message Variable3. Equivalent to ABAP's sy-msgv3
MESSAGE_V4	O	CHAR	50	Message Variable4. Equivalent to ABAP's sy-msgv4
PARAMETER	P	CHAR	32	Parameter Name.
MSGROW	Q	INT4	10	Lines in parameter
MSGFIELD	R	CHAR	30	Field in parameter
MSGSYSTEM	S	CHAR	10	Logical system from which message originates.

- On receiving rejection message, the receiver has to first update its local store with the state IDs sent from the DOE and then can take either of the two courses of action:
  - Update the local data store with the image sent from the DOE or
  - Retain the image of the instance on the client, resolve the reason for rejection and upload again to DOE. Please note that the state ID has to be updated even if receiver decides to retain the local image.

7.4 Import

- Any changes to a data object instances in DOE have to be sent to all subscribed receivers. The changes include creation of a new instance; update or deletion of an existing instance. These data object instances are sent to receivers either as bulk messages or import messages.
- Each import message will have exactly one data object instance.
- Only the changed attributes or node instances are sent.

7.5 Bulk

- Bulk message contains more than one data object instance in a single message.
- Only the instances to be inserted on the client are bulked together by DOE in the bulk message.
- On receiving the bulk message, the receiver should insert all the node instances in the local data store.
- Number of instances that are bulked together can be configured in the DOE configurations. The parameter is CLIENT_PACKAGE_SIZE.

7.6 Zap

- On receiving a ZAP message, the receiver is expected to delete all the instances of the data object. The format of the ZAP message is as follows:

```
<?xml version="1.0" encoding="utf-8"?>

<Msg type=”Z” >

</Msg>
```

7.7 Current State

- The current state message represents the current image of an instance in DOE.
- When the receiver receives this message, it is expected to replace the instance on the local data store with the instance in the message.
- Each message will have exactly one data object instance.

8. Miscellaneous APIs

8.1 Remote Download of Applications

Mobile applications that are developed can be downloaded and deployed to the receivers via the DOE. To use this functionality, following steps needs to be performed:

1. Develop a mobile application and group the individual files in any format – exe, jar, war, etc.,
2. Upload these files as “add on” to DOE using the transaction SDOE_ARCHIVE_UPLOAD.
3. Once uploaded, these filed are available as MCDs (Mobile Component Descriptors) in the NetWeaver Mobile Administration portal.
4. These MCDs can be assigned to the logical devices on the DOE.
5. Data object message of ARCHIVE (present in SAP Basis 7.10 SWCV) data object instances are created for the receiver. These messages can be extracted to receiver using the Get Messages API.
6. On the receiver extract the Synckey from the ARCHIVE node (root node) and call the RFC enabled function module MMW_EXTRACT_ARCHIVE. The signature is as follows:

Parameter Name	Type	Use
SESSION_ID	Importing	Unique identifier created during login to DOE.
ARCHIVE_ID	Importing	Synckey of the archive data object
START_POSITION	Importing	Position from which data needs to be sent. This is applicable when the archive data is not sent completely but its sliced and sent
REQUESTED_SIZE	Importing	Client can request the size of archive to be sent by setting this field value. If the client wants to retrieve the full archive, the size_requested should be set to -1
TOTAL_ ARCHIVE_SIZE	Exporting	Size of the archive
RESPONSE_OFFSET	Exporting	Offset from the start position. This given the information of the total number of bytes sent
RESPONSE_SIZE	Exporting	Size of archive sent in the message
ARCHIVE_CONTENT	Exporting	The actual archive contents

Exceptions

Exception	Description	Client Reaction
INVALID_SYNCKEY	This exception is raised when the	Check for the synckey of archive

	synckey sent in the request is invalid	present in the device with the synckey of the data object. Synckey is the unique identifier for each instance of the data object
ARCHIVE_NOT_FOUND	When the archive is not uploaded to DOE or if the archive is empty, this exception is raised	Check if the archive is actually uploaded and also if the archive is not empty

Note

If the archive is a deployable unit, the receiver can automatically deploy the application after downloading the archive.

8.2 Password Change

You use the RFC enabled function module MMW_SYNC_PASSWORD_CHANGE, to change the password of the SAP WebAS user.

Parameter Name	**Type**	**Use**
USER_NAME	Importing	User name for whom the password needs to be changed. By default it's the sync user
OLD_PASSWORD	Importing	Old password
NEW_PASSWORD	Importing	Changed password
STATUS	Exporting	Status flag representing if the operation is done or not. Values: 'F' for fail and 'S' for success.
ERROR_TEXT	Exporting	Reason for failure