WebSphere MQ Integrator - IDoc parser for SAP

Version 1.2

19th July 2002

Mike Johnson
WebSphere MQ Integrator Development
IBM United Kingdom Laboratories
Hursley
Hampshire
United Kingdom

Take Note!

Before using this report be sure to read the general information under "Notices".

Fifth Edition, July 2002

This edition applies to Version 1.2 of WebSphere *MQ Integrator - IDoc parser for SAP* and to all subsequent releases and modifications unless otherwise indicated in new editions.

Table of Contents

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product.

Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS-IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both

- IBM
- AIX
- MQSeries
- MQSeries Integrator
- WebSphere

The following terms are trademarks of other companies:

- SAP, IDoc, R/3 SAP AG
- Solaris Sun Corporation

- Windows NT Microsoft Corporation

Summary of Amendments

Date	Changes
14 May 2001	Initial release
24 May 2001	Remove reference to versions of MQSI beyond V2.0.1
20 July 2001	SupportPac now supports MQSI V2.0.2
26 November 2001	Initial release for AIX, no functional changes
19 February 2002	Add support for WebSphere MQ Integrator for AIX V2.1 Add support for WebSphere MQ Integrator for Sun Solaris V2.1 Add support for WebSphere MQ Integrator for HP-UX V2.1 Add support for WebSphere MQ Integrator for Windows NT/2000 V2.1

Preface

This SupportPac contains a set of parsers written for MQSeries Integrator V2.0.1, MQSeries Integrator V2.0.2 and WebSphere MQ Integrator V2.1. They are designed to operate in the AIX, Sun Solaris, HP-UX and Microsoft Windows environments and add support for parsing of input messages and creation of output messages in the SAP IDoc Version 4 format. The SupportPac includes executable programs for the AIX, and Windows environments, including metadata files and tools.

A general overview of the IDoc message is included.

For the remainder of this document, all four versions of the product will be referred to as WebSphere MQ Integrator unless otherwise specified.

A general understanding of WebSphere MQ Integrator is necessary to use this SupportPac.

Bibliography

- *IBM MQSeries Integrator for Windows NT Version 2 Installation Guide,* IBM Corporation. SC34-5600.
- *IBM MQSeries Integrator for AIX Version 2 Installation Guide,* IBM Corporation. SC34-5841
- *IBM MQSeries Integrator Version 2 Using the Control Center,* IBM Corporation. SC34-5602
- *IBM MQSeries Integrator Version 2 Programming Guide,* IBM Corporation. SC34-5603

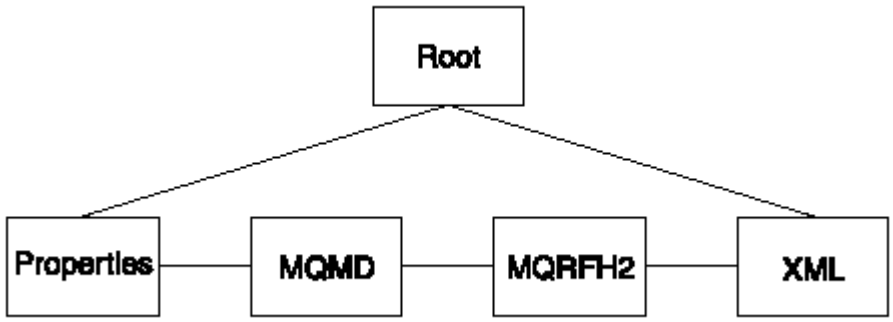
Chapter 1. Overview

The IDoc parser represents incoming data to WebSphere MQ Integrator in a format that the Integrator compute node can manipulate. It also allows data of a different format to be mapped to an IDoc stream.

From an WebSPhere MQ Integrator point of view the message consists of a one dimensional array of bits organised into bytes.

The Brokers first task is to pass the message to a message parser. The parser converts the string of bits to a tree format. The incoming message data is of a single message format, so one parser is responsible for parsing the entire contents of the message. The class name of the parser that is needed is defined in the Format field in the MQMD or the MQRFH2 header of the input message.

A typical incoming bitstream might be represented as follows:



Here we have an incoming MQSeries message which is composed of the following:

- A Message Descriptor (MD) which is given to the MD parser.
- An RFH2 header which is passed to the RFH2 parser.
- An XML portion which is handled by the XML parser.

Once the bitsream has been represented in this syntax element tree, it can be manipulated by WebSphere MQ Integrator nodes.

The Compute node constructs an output message. The elements of the output message can be defined using an SQL expression, and can be based on elements of both the input message and data from an external database.

The IDoc parser uses the Message Repository Manager (MRM) to model the segments of the IDoc data. The job of the IDoc parser at runtime is to understand each part of the bitstream so it can hand the appropriate parts of this bit stream off to the MRM for parsing.

Chapter 2. SupportPac Installation

Prerequisites

This SupportPac provides parsers to be used with MQSeries Integrator V2.0.1, V2.0.2 and WebSphere MQ Integrator V2.1. For normal use, there are no pre-requisites other than those required by the products themselves.

The minimum service level for WebSphere MQ Integrator V2.1 is CSD3.

One of the utilities is a Perl script.

Supported Platforms

This SupportPac has been developed for and tested on the following platforms:

- AIX
- Sun Solaris
- HP-UX
- Microsoft Windows NT

Installing

The zip file (ia0f.zip) should be unzipped into a temporary directory. The following files will be created:

- hdrfiddle.pl pearl script to format the SAP output
- matmas.mrp sample message set
- idoctransform.exp contains two sample message flows
- smqesmp3.c sample exit for use with MQSeries link for R/3
- sample.idoc flat file representation of an IDoc
- \AIX
  - mqsi201\imbdfsmq.lil parser executable for MQSeries Integrator V2.0.1
  - mqsi202\imbdfsmq202.lil parser executable for MQSeries Integrator V2.0.2
  - The Parser lil for V2.1 is supplied with the CSD03 level of maintenance for WebSphere MQ Integrator V2.1 for AIX
- \NT
  - mqsi201\imbdfsmq.lil parser executable for MQSeries Integrator V2.0.1

  - mqsi202\imbdfsmq202.lil parser executable for MQSeries Integrator V2.0.2
  - The Parser lil (imbdfsmq.lil) for V2.1 is supplied with the CSD03 level of maintenance for WebSphere MQ Integrator V2.1 for NT
- \Solaris
  - The Parser lil for V2.1 is supplied with the CSD03 level of maintenance for WebSphere MQ Integrator V2.1for Solaris
- \HP-UX
  - The Parser lil for V2.1 is supplied with the CSD03 level of maintenance for WebSphere MQ Integrator V2.1 for HP-UX
- ia0f.pdf This User Guide
- license2.txt

The approriate parser executable (*imbdfsmq.xxxlil*) should be copied to the following directory and renamed to *ibmbdfsmq.lil*:

- <mqsi_root>\bin (Windows)
- <mqsi_root>/lil (AIX,Sun Solaris, HP-UX)

If you are using MQSeries Integrator V2.0.2 then use the parser executable *imbdfsmq202.lil*. If you are using WebSphere MQ Integrator V2.1 then use the parser executable *imbdfsmq.lil* supplied with CSD03. This package does not include the lil for WebSphere MQ Integrator V2.1

The broker must be stopped and restarted to enable it to detect the existence of the new '**lil**'

The *idoctransform.exp* contains two sample message flows that can be used to validate the proper functioning of the parser. They must be imported into a Configuration Manager using the import function of the Control Center, and then assigned to an execution group and deployed. They expect the following local queues to be available:

- SMQ_OUTBOUND_QUEUE
- SMQ_INBOUND_QUEUE
- Dead letter queue defined for the queue manager

The utility executable and related files should be moved to a program directory. This can be the same directory as the parser or it can be a separate directory.

The matmas.mrp is a sample message set that gives an example of a message set with three available messages.

The source file *smqesmp3.c* is a sample exit that can be used with the MQSeries link for R/3. Refer to the MQSeries Link for R/3 User's Guide for details of how to build this exit. It is supplied as an example of how the link header can be removed and the MD format field changed, before the MQSeries Link for R/3 puts the message onto an MQSeries queue..

The *sample.idoc* is a flat file representation of an IDoc.

The *hdrfiddle.pl* is a pearl script for cleaning the metadata extracted from SAP.

These files should be moved to a program directory. This can be the same directory as the parser or it can be a separate directory.

Use the WebSphere MQ Integrator utility *mqsimrmimpexp* to import the sample mrm definitions. Refer to the WebSphere MQ Integrator Administration Guide for further details. Use the import facility of the WebSphere MQ Integrator to load the sample message flow. The message flow will need to be edited for the local environment.

Chapter 3. Using the Parser

The IDoc parser takes input messages in valid IDoc format and creates WebSphere MQ Integrator logical message tree structures that can then be processed by WebSphere MQ Integrator message flows. Similarly, it will take a logical message tree created by a message flow and produce the data portion of an MQSeries message in IDoc format.

The parser will create the message tree from an input message in a specified format. This format must be followed when a message tree is built in an WebSphere MQ Integrator message flow.

All logical message trees used within WebSphere MQ Integrator have a basic structure. This comprises of a single high-level element known as the root element. The user data is found in the body of the message. The body has a single high-level element that is the last child of the root element. For a SAP IDoc message the name of this high level element must be “IDOC” to match the message domain supported by the parser.

Chapter 4. SAP IDoc message formats

To manipulate IDoc data you should be familiar with the IDoc format and the content that these IDocs can have. What is documented below is a very general overview of the IDoc structure. For further information please refer to SAP's own documentation.

A Version 4 IDoc is made up of fixed sized structures. The first structure is the control structure of the IDoc the DC and is 524 bytes long. There is then one or more data structures known as DD's, each of which are 1063 bytes long.

Each DD contains the segment data of the IDoc and this segment data can be modeled in the MRM.

Each DD structure is composed of name value pairs with the last field of the structure, which is 1000 bytes long, holding the segment data.

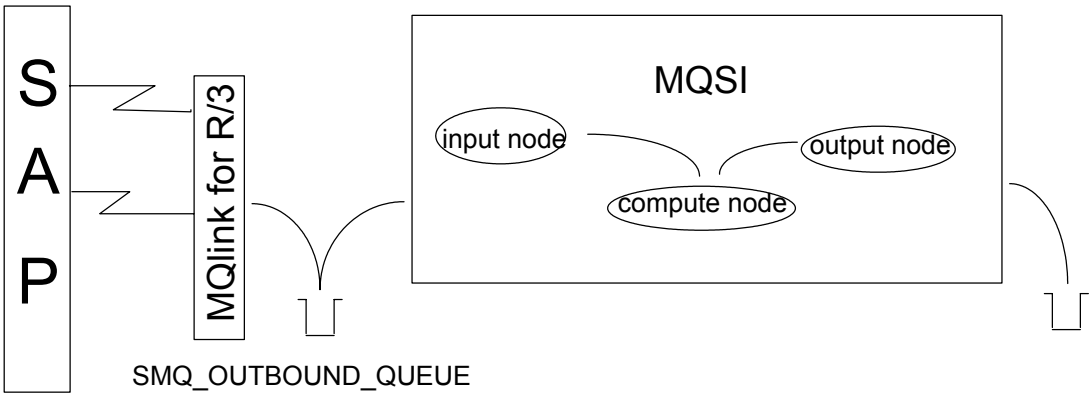
There are many message types that be held in an IDoc. Each message type can have many different segments.

Input Messages

The IDoc parser will parse any inbound message that is in a valid IDoc format. It will perform limited checking of the message format. The IDoc parser will register with the execution broker for a message domain of "IDOC". It will attempt to parse any input message that is read by WebSphere MQ Integrator. The message domain can be specified in an RFH2 header in the message itself, or as a default property on the MQInput node of an WebSphere MQ Integrator message flow.

The parser will create a logical message tree that reflects the contents of the message. The name of the top-level element of the body will be "IDOC". The rest of the data for a particular group will be built as a logical data structure under this high level element.

A typical flow might be set up as follows:

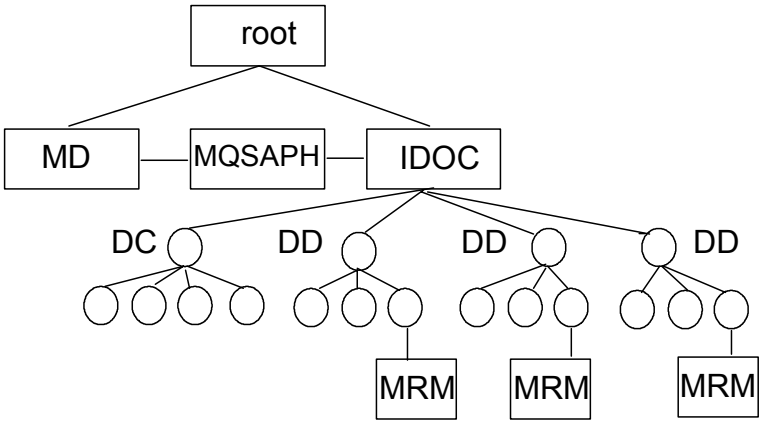


An IDoc flows out of SAP and is sent to the MQSeries link for R/3. Once this IDoc has been successfully comitted to the outbound MQSeries queue, the input node of the WebSphere MQ Integrator will read it from that queue. In this example, the input node has been configured to read from the SMQ_OUTBOUND_QUEUE and its default message domain is set to IDOC. By putting an WebSphere

MQ Integrator trace node between the input node and the compute node you can see the syntax element tree generated. The compute node can then be used to manipulate this syntax element tree.

Chapter 5. Parser Implementation

An IDoc that had been sent from SAP to the MQSeries link for R/3 would be represented in WebSphere MQ Integrator by the IDoc parser as:



The first parser that the broker calls is for the Message Descriptor (MD) of the message, it then calls the parser for the MQSAPH. This is the header of all messages from the MQSeries Link for R/3 which connects SAP to MQSeries.

The IDoc parser has a Control Structure (DC) folder with name value pairs for the elements of the DC structure. It then has one or more Data Structure (DD) folders with the last element of each DD folder holding the segment data. The MRM is then used to model each individual segment.

Output Messages

The message flow must create an output message in a similar format as the input message.

Once the Compute node has finished manipulating the syntax element tree the IDoc parser is called to rebuild the bitstream at the output node of the MQSI.

An example of some eSQL from a compute node

```
SET OutputRoot = InputRoot;

SET OutputRoot.IDOC.DC[1].tabnam = 'EDI_DC40  ';

SET OutputRoot.IDOC.DD[2].sdatatag.MRM.maktx = 'Buzzing all day';
```

The first line copies the incoming IDoc to the outgoing IDoc.

The second line sets the tabname of the first DC.

The third line is using the second DD segment, which in this example is of type E2MAKTM001, and is setting the maktx field.

See **Appendix A** for the field names in the DC and DD recognised by the IDoc parser.

Chapter 6. Customisation of the supplied metadata files

Offline Utilities

An offline utility is provided to assist in building the necessary metadata files. The first utility will process output metadata file from SAP and produce the corresponding metadata files for the MRM data dictionary.

Building the Metadata files

This section documents how the metadata file that is provided with this parser was built. An example pre-built version of the metadata file is provided with this SupportPac. This procedure should be used to create the metadata of the required IDoc data.

1. Logon to a SAP system
2. Run transaction we60
3. For ObjectName select the IDoc type of interest, for example MATMAS02
4. Select the version of the record type (A version 4 IDoc is type 3!)
5. From the Documentation pull down select 'C-header'
6. An option will be given to save the output to a file

Formatting the output for import to the MRM

Use the supplied perl script *hdrfiddle.pl* to clean the output file form SAP. It takes the metadata file of SAP's and makes it more acceptable for importing into the MRM.

In the Control Center use this file as the source for importing a message set into MQSI. Refer to the WebSphere MQ Integrator Using the Control Center Documentation on how to import C structures into the MRM.

Once structures have been imported into the MRM, each segment of the IDoc can then be created as a Message Component of this new Message Set. To create a Message Component involves associating the component with its Compound Type. It may be necessary to add the Compound Types, you have imported, to the Control Center's Workspace before you can create the appropriate Message Component for your segment.

Modeling the IDoc in the MRM

Once the C structure has been imported into WebSphere MQ Integrator you will have a number of MRM types. Each type represents a definition of an IDoc segment.

Create a message set for your object for example, matmas02. Its Custom Wire Format identifier should be set to CWF

Create messages under this message set for all the segments that the IDoc parser may need to handle. Each message name should be the name of the segment it represents and be in capital letters. For example:

Message	Type
E2MARMM	e2marmm_1_type

For each of the elements associated with each message (i.e. segment of the IDoc) it may be necessary to change the padding character to that of a space. This is because an IDoc is padded with spaces.

e.g. Model of a SAP segment in WMQI called E2MAKTM001

E2MAKTM001	Message name to WMQI	(segment name to SAP)
msgfn	First element name to WMQI	(field name to SAP)
spras	Second element name	
maktx	Third element name	
spras_iso	Forth element name	
fill954	Fifth element to make the size of this element up to 1000 characters	

In this example:
From the Message Set panel of the Control Center select the element fill954 of the message E2MAKTM001, then select Custom Wire Format tab and change the padding character to a space.

APPENDIX A

This Appendix lists all the field names of the DC and the DD structures as understood by the IDoc parser. They are documented in the form that they would be used in a SET statement of eSQL. For example:

SET OutputRoot.Properties = InputRoot.Properties;

SET OutputRoot.MQMD = InputRoot.MQMD;

Control section (DC) fields

All fields must be specified and set

The syntax is <rootname>.<Parser Name>.folder name>.<field name> =

SET "OutputRoot"."IDOC"."DC"."tabnam" =

SET "OutputRoot"."IDOC"."DC"."mandt" =

SET "OutputRoot"."IDOC"."DC"."docnum" = '0000000000000001';

SET "OutputRoot"."IDOC"."DC"."docrel" = '45B';

SET "OutputRoot"."IDOC"."DC"."status" = '30';

SET "OutputRoot"."IDOC"."DC"."direct" = '1';

SET "OutputRoot"."IDOC"."DC"."outmod" = '4';

SET "OutputRoot"."IDOC"."DC"."exprss" = '';

SET "OutputRoot"."IDOC"."DC"."test" = '';

SET "OutputRoot"."IDOC"."DC"."idoctyp" = 'MATMAS01';

SET "OutputRoot"."IDOC"."DC"."cimtyp" = '';

SET "OutputRoot"."IDOC"."DC"."mestyp" = '';

SET "OutputRoot"."IDOC"."DC"."mescod" = '';

SET "OutputRoot"."IDOC"."DC"."mesfct" = '';

SET "OutputRoot"."IDOC"."DC"."std" = '';

SET "OutputRoot"."IDOC"."DC"."stdvrs" = '';

SET "OutputRoot"."IDOC"."DC"."stdmes" = '';

SET "OutputRoot"."IDOC"."DC"."sndpor" = '';

```
SET "OutputRoot"."IDOC"."DC"."sndprt" = '';
SET "OutputRoot"."IDOC"."DC"."sndpfc" = '';
SET "OutputRoot"."IDOC"."DC"."sndprn" = '';
SET "OutputRoot"."IDOC"."DC"."sndsad" = '';
SET "OutputRoot"."IDOC"."DC"."sndlad" = '';
SET "OutputRoot"."IDOC"."DC"."rcvpor" = '';
SET "OutputRoot"."IDOC"."DC"."rcvprt" = '';
SET "OutputRoot"."IDOC"."DC"."rcvpfc" = '';
SET "OutputRoot"."IDOC"."DC"."rcvprn" = '';
SET "OutputRoot"."IDOC"."DC"."rcvsad" = '';
SET "OutputRoot"."IDOC"."DC"."rcvlad" = '';
SET "OutputRoot"."IDOC"."DC"."credat" = '';
SET "OutputRoot"."IDOC"."DC"."cretim" = '';
SET "OutputRoot"."IDOC"."DC"."refint" = '';
SET "OutputRoot"."IDOC"."DC"."refgrp" = '';
SET "OutputRoot"."IDOC"."DC"."refmes" = '';
SET "OutputRoot"."IDOC"."DC"."arckey" = '';
SET "OutputRoot"."IDOC"."DC"."serial" = '';
```

Data section (DD) fields

To access each DD segment use the array suffix ie. DD[1] , DD[2] etc.

Note the use of the ‘2’ sufix to give unique field names on the mandt and docnum fields

```
SET OutputRoot.IDOC.DD[I].segnam  = 'E2MAKTM001';
SET OutputRoot.IDOC.DD[I].mandt2   = '111';
SET OutputRoot.IDOC.DD[I].docnum2  = '9999999999999111';
SET OutputRoot.IDOC.DD[I].segnum   = '111000';
SET OutputRoot.IDOC.DD[I].psgnum   = '000111';
SET OutputRoot.IDOC.DD[I].hlevel   = '11';
```

Segment fields

Use the ‘sdatatag’ keyword to indicate to the parser that it is the element that contains the segment data which is to be manipulated. The MRM indicates that the MRM will handle the transformation. The final field identifies the actual field of the segment.

The final line is the filler for the segment as an incoming IDoc to SAP must have each segment 1000 bytes long.

```
SET OutputRoot.IDOC.DD[I].sdatatag.MRM.msgfn = '006';

SET OutputRoot.IDOC.DD[I].sdatatag.MRM.spras = 'E';

SET OutputRoot.IDOC.DD[I].sdatatag.MRM.maktx = 'Buzzing all night';

SET OutputRoot.IDOC.DD[I].sdatatag.MRM.msgfn = '006';

SET OutputRoot.IDOC.DD[I].sdatatag.MRM.spras_iso = 'EN';

SET OutputRoot.IDOC.DD[I].sdatatag.MRM.fill954 = '   ';
```

End of Document