WebSphere Message Broker IDoc parser for SAP tools update Version 2.2

21st January 2008

Lee Hollingdale and Roy Saxton
IBM United Kingdom Laboratories
Hursley Park
Hampshire
United Kingdom

Property of IBM

Take Note!

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

Third Edition, January 2008

This edition applies to Version 2.2 of *WebSphere Message Broker - IDoc parser for SAP tools update* and to all subsequent releases and modifications unless otherwise indicated in new editions.

Notices

Trademarks and service marks

History of changes

Date	Changes
19-07-2002	SupportPac IA0F Version 1.2 released
19-07-2003	SupportPac IA0F Version 1.2 withdrawn
19-07-2007	SupportPac IA0F Version 2.0 released, comprising updated tools to support the IDoc parser (which is now incorporated into the WebSphere Message Broker Version 6.0 base product).
06-11-2007	SupportPac IA0F Version 2.1 released, with enhanced parsing and variable name substitution for C header files exported from SAP.
21-01-2008	SupportPac IA0F Version 2.2 released, with support for WebSphere Message Broker Version 6.1.

What's new in Version 2.2

SupportPac IA0F Version 2.2 has been updated to provide support for WebSphere Message Broker Version 6.1. The documentation has been updated, and a customized options file is provided for creating message definitions in Message Broker Version 6.1.

A note on terminology:

Message Broker Version 6.1 documentation and SupportPac IA0F documentation differ slightly in the terms they use for SAP artifacts, as detailed in the following table.

Message Broker Version 6.1 term	**SupportPac IA0F term**
File IDoc	'Flat' IDoc
ALE IDoc	R/3 Link format IDoc

What's new in Version 2.1

C header files exported by SAP may contain variable names that would cause a problem if you were to try to import them unchanged into Message Broker toolkit. Version 2.1 of this SupportPac has enhanced the parsing and variable name modification capabilities of IDocHeaderTweak. It eliminates invalid variable names according to the following model:

1. Check for all C++ reserved words and map to safe alternatives.
2. Check for characters not in the set [a-zA-Z0-9_] and replace them with underscore. If there are no valid characters originally in the identifier (for example !£$@), this will lead initially to an identifier with all underscores that is transformed further as described below.
3. Check for identifiers with leading underscores and delete as far as the first [a-zA-Z0-9].
4. If the identifier is all underscores (which would be the first transformation from all invalid characters), replace with XXX_<sfx> where <sfx> is a number that is incremented on every such substitution.
5. Check for identifiers with leading numerals and replace the leading numeral with lower case x.

Examples:

• __cplusplus[nn]	-> cplusplus_[nn]	(Reserved word - mapped to a safe alternative)
• 999trs[nn]	-> x99trs[nn]	(leading numeral - substituted with x)
• hj%$?qwerty[nn]	-> hj___qwerty[nn]	(some invalid characters - replaced by _)
• _abcd[nn]	-> abcd[nn]	(leading underscore - deleted)
• !£#?[nn]	-> XXX_1[nn]	(all invalid characters, first found)
• &&£/!![nn]	-> XXX_2[nn]	(all invalid characters, second found)

You should review your programming logic appropriately in order to use these modified variable names.

Preface

This SupportPac contains a set of tools written for WebSphere Message Broker Version 6.0 and WebSphere Message Broker Version 6.1. They are designed to help you create message sets from IDocs in the SAP IDoc Version 4 format.

You should understand how Message Sets are used in WebSphere Message Broker in order to use this SupportPac.

Bibliography

Online documentation for WebSphere Message Broker V6.0 is available from this link:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp

Online documentation for WebSphere Message Broker V6.1 is available from this link:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r1m0/index.jsp

Chapter 1. Introduction

New in WebSphere Message Broker Version 6 is the complete integration of the IDOC parser, which was previously only available in SupportPac IA0F. This parser allows your message flows to consume and produce IDocs in the format used by the MQSeries Link for R/3.

What is the MQSeries Link for R/3?

At its simplest, the MQSeries Link for R/3 is a pair of applications that communicate with SAP using ALE on one side and WebSphere MQ (formerly MQSeries) on the other. It provides bi-directional transport of IDocs between SAP systems and WebSphere MQ (MQSeries) queue managers.

It was the first messaging product to receive the SAP ALE Message Handling Systems Certification, and although it is no longer the IBM strategic direction for WebSphere MQ/SAP integration, this SupportPac is provided for the convenience of customers still using the R/3 Link.

Along with the new IDOC parser, WebSphere Message Broker Version 6 has also added a skeletal IDoc message definition as one of the built-in message types available. However, this message definition contains only a definition of the generic DC and DD parts of the IDoc structure — you have to create definitions for the application-specific parts of the structure yourself.

Supplied Applications

SupportPac IA0F no longer contains the IDOC parser, but it provides new utility programs that help you to create the application-specific parts of your IDoc structures.

The utility programs are:

- IDocConverter

  Converts 'flat' IDocs (ones that are saved to the file system by SAP) into ALE format.
- IDocHeaderTweak

  Pre-processes a C header file describing an ALE IDoc so it can be imported by Message Broker.
- IDocMsgSetTweak

  Post-process a message set created by importing a C header file describing an ALE IDoc.

The Java code for all the utilities is packaged into a single Jar file called `IDocUtils.jar` and each utility has its own batch file to simplify its execution.

Other chapters in this document:

Chapter 2 describes how to install the SupportPac.

Chapter 3 explains how to use the utility programs.

Chapter 4 gives details of a scenario that uses the utility programs.

WebSphere Message Broker Version 6.1

WebSphere Message Broker Version 6.1 deprecates the IDOC parser, and instead recommends that the MRM parser is used to consume and produce IDocs in the format used by the MQSeries Link for R/3. New capability added to the MRM parser in recent releases means that it is capable of parsing such IDocs, and there is no need for a dedicated IDOC domain or parser. Note that the MRM parser uses its Tagged/Delimited String (TDS) physical format to model IDocs, instead of Custom Wire Format (CWF).

Because TDS is used, the MRM parser can also parse 'flat' IDocs without conversion, which the IDOC parser cannot. When using the MRM parser, you therefore do **not** need to run utility *IDocConverter*.

Message Broker Version 6.1 upgrades the C importer to perform the pre-import and post-import processing logic in Java utilities *IDocHeaderTweak* and *IDocMsgSetTweak*. This applies to both MRM and IDOC domains. These utilities do **not** need to be run when using Message Broker Version 6.1.

A further advantage of using the MRM domain is that IDocs can be transformed using the Mapping node (the Mapping node does not support the IDOC domain).

Chapter 2. SupportPac Installation

Prerequisites

This SupportPac provides utility programs and other resources for use with WebSphere Message Broker Version 6.0 and 6.1.

Supported Platforms

This SupportPac has been developed and tested on platforms running the Microsoft Windows operating system. The artifacts produced by the utility programs contained in this SupportPac should be transferred in ASCII mode to the appropriate target platforms.

Installing

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

- IDocUtils.jar – the packaged utilities
- IDocConverter.bat – batch file to invoke IDocConverter
- IDocHeaderTweak.bat – batch file to invoke IDocHeaderTweak
- IDocMsgSetTweak.bat – batch file to invoke IDocMsgSet
- ia0f.pdf - this User Guide
- license (directory) – containing the license files
- IDocOptions610.xml - options file for use with Message Broker 6.1 mqsicreatemsgdefs command

The batch files are command line utilities, and you have a choice of moving these to a directory that is included in the PATH of your local environment, or moving them to another directory which will be the location from where you execute them.

You may either add the IDocUtils.jar file to the CLASSPATH of your local environment, or move it to the same directory as the batch files.

Chapter 3. Using the utility programs

IDocConverter

Use *IDocConverter* if you want to process 'flat' IDocs and either you are using Message Broker Version 6.0, or you are using Message Broker Version 6.1 and the IDOC parser.

This utility takes a "flat IDoc" and converts it into an R/3 Link format IDoc.

In this context, a flat IDoc is one that is saved to the file system by SAP—for example, when an IDoc is sent to a Port that has a type of "File". Its records are delimited by carriage returns, the DC record always being 524 bytes long and the DD segment records being variable length.

Conversely, an R/3 Link format IDoc is a fixed-length structure containing a 524-byte DC record followed by one or more 1063-byte DD records. It contains no carriage returns or other delimiting characters.

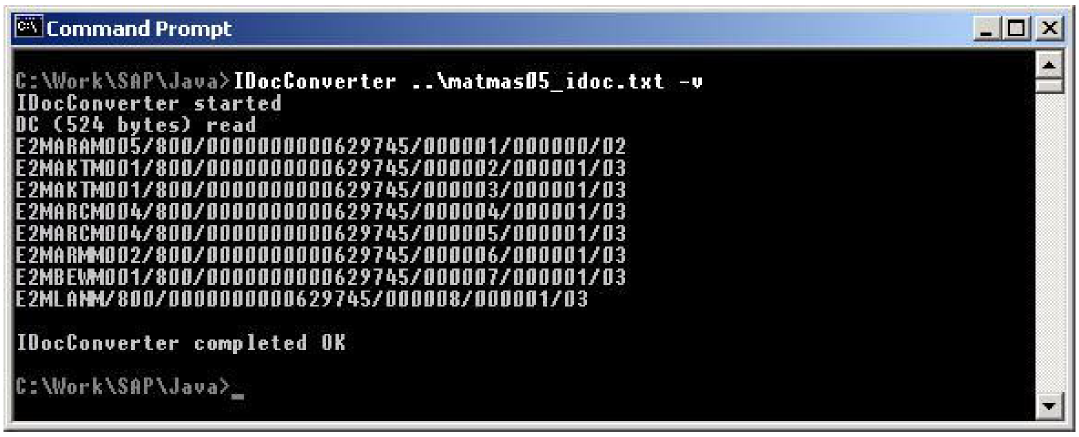
Both formats are textual, so the algorithm for the conversion is fairly simple:

- For each line in the input file, remove the carriage return
- If it is the first line, check that it is 524 bytes long and write it to the output file
- If it is a subsequent line, pad it with spaces to 1063 bytes long and write it to the output file
- Any discrepancies (incorrect DC record length or DD record 'overflow') result in an exception being thrown and the utility terminating; a partial output file may be written

The tool's command-line syntax is as follows:

```
IDocConverter inputFilename [-v]
```

The *inputFilename* is the pathname of the flat input IDoc, and the optional *–v* flag indicates that verbose output should be sent to the console. Here is an example screenshot of the tool in action:



```
C:\Work\SAP\Java>IDocConverter ..\matmas05_idoc.txt -v
IDocConverter started
DC (524 bytes) read
E2MARAM005/800/000000000000629745/000001/000000/02
E2MAKTM001/800/000000000000629745/000002/000001/03
E2MAKTM001/800/000000000000629745/000003/000001/03
E2MARCM004/800/000000000000629745/000004/000001/03
E2MARCM004/800/000000000000629745/000005/000001/03
E2MARMM002/800/000000000000629745/000006/000001/03
E2MBEWM001/800/000000000000629745/000007/000001/03
E2MLANM/800/000000000000629745/000008/000001/03

IDocConverter completed OK

C:\Work\SAP\Java>_
```

Note that the *–v* option has caused details about the input IDoc to be displayed. You can see that the DC is of the correct length and is followed by 8 DD segments. The tool displays the parsed content of the 6 header fields in each DD.

The output file will be written to the same directory as the input file, the filename having the `.out` extension added. In the above example, a file called `matmas05_idoc.txt.out` was written to the parent directory.

IDocHeaderTweak

Use *IDocHeaderTweak* if you are using Message Broker Version 6.0.

This utility takes a C header file containing IDoc segment definitions (such as the one exported from SAP using transaction WE60), and modifies it so that it is suitable for import into Message Broker.

There are a number of reasons why the exported C header file will not import successfully as-is:

- Use of the Char C++ type, which isn't supported by the Broker's C importer
- Use of preprocessor directives such as #define, not supported by the importer
- Use of reserved C++ keywords as field names (for example, compl) which causes the import to fail

The utility takes the original C header file and removes or replaces the unsupported features. In addition, it adds an extra field to each segment structure definition to bring its total size up to 1000 bytes—this field is called pad*nnn*, where *nnn* is the size of the padding field.

Those readers familiar with the IDOC parser available in SupportPac IA0F will be aware that the package included a Perl script called hdrFiddle.pl. IDocHeaderTweak provides the same functionality as the Perl script but has the following advantages:

- It does not prereq a Perl environment, and can use the same JRE that is shipped with Message Broker
- It adds functionality to replace field names that are reserved C++ keywords (currently the only mapping implemented is of compl to compll)
- It allows you to split a single input file into multiple output files, each containing a single segment definition; you may wish to do this for housekeeping purposes

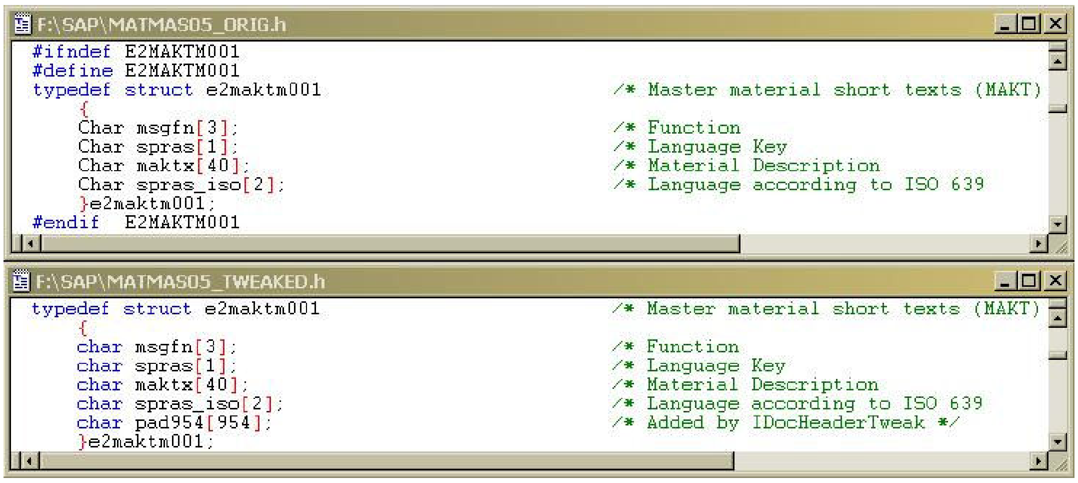
The tool's command-line syntax is as follows:

```
IDocHeaderTweak inputFilename [-v] [-s]
```

The *inputFilename* is the pathname of the SAP-exported C header file, and the optional *–v* and *–s* flags enable verbose output and output file splitting respectively.

If you have not supplied the split option, then a single new file called *X*_TWEAKED.h will be created (where *X*.h was the original filename). If you have supplied the split option, then multiple files named *S*.h will be created, where S represents each segment name.

The picture below shows a "before and after" view of a single segment definition:



```
#ifndef E2MAKTM001
#define E2MAKTM001
typedef struct e2maktm001                          /* Master material short texts (MAKT)
    {
    Char msgfn[3];                                 /* Function
    Char spras[1];                                 /* Language Key
    Char maktx[40];                                /* Material Description
    Char spras_iso[2];                             /* Language according to ISO 639
    }e2maktm001;
#endif  E2MAKTM001
```

```
typedef struct e2maktm001                          /* Master material short texts (MAKT)
    {
    char msgfn[3];                                 /* Function
    char spras[1];                                 /* Language Key
    char maktx[40];                                /* Material Description
    char spras_iso[2];                             /* Language according to ISO 639
    char pad954[954];                              /* Added by IDocHeaderTweak */
    }e2maktm001;
```

Note the removal of the preprocessor directives, the replacement of the Char type, and the addition of the pad954 field.

Finally, you should delete or move the original header file so that it doesn't get picked up during a subsequent invocation of *mqsicreatemsgdefs*.

IDocMsgSetTweak

Use *IDocMsgSetTweak* if you are using Message Broker Version 6.0.

This utility takes a Message Set definition file—as seen in the Broker Toolkit with a .mxsd extension—and modifies the names of the Message definitions it contains.

In order for the IDOC parser to parse an entire IDoc at runtime, it takes the segment name contained in each DD header (in the *segnam* field) and uses this as the Message name for the rest of the segment data. For example, if the field InputRoot.IDOC.DD[1].segnam contains 'E2MARAM005', then the contents of the InputRoot.IDOC.DD[1].sdatatag element is subsequently parsed as an MRM Message of type E2MARAM.

Unfortunately, the default and unalterable behaviour of the Message Broker Version 6.0 importers is to create Message definitions called msg_*xxx*, where *xxx* is the name of the structure being imported. Thus, at runtime, the IDOC parser will be unable to find a matching Message definition and will fail.

Thus, the user needs to modify the names of the Message definitions created so that they match the corresponding segment names. This can be done manually through the Toolkit, or can be automated using this utility.

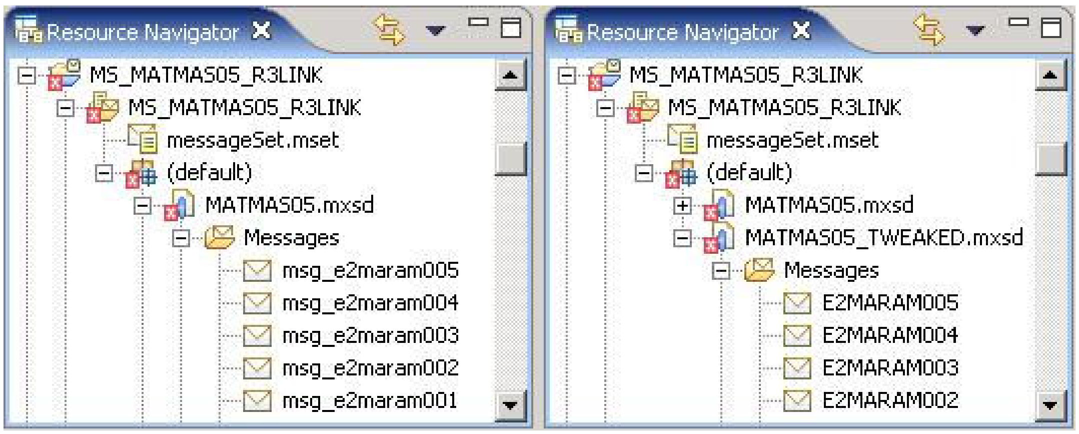
The tool's command-line syntax is as follows:

```
IDocMsgSetTweak inputFilename [-v]
```

The *inputFilename* is the pathname of the MXSD file created in the workspace by the *mqsicreatemsgdefs* tool, and the optional *–v* option produces verbose output.

Running this tool will create a second MXSD file called *X*_TWEAKED.mxsd, where *X*.mxsd is the original name. Refreshing the Toolkit view of the Message Set project in question will show both message sets, along with lots of error messages indicating duplicate definitions.

Below you can see an example of this:



You can see that the old Message names (left) have been replaced with ones that the IDOC parser will be able to resolve (right). You can delete the old MXSD file, the error messages will disappear and you will be able to save and use the Message Set.

When to use the utilities

The table summarises the circumstances when it is necessary to use the utility programs:

	IDOC parser	***MRM parser***
Message Broker 6.0	IDocConverter * IDocHeaderTweak IDocMsgSetTweak	<Not supported>
Message Broker 6.1	IDocConverter *	<No utilities needed>

* If you have ‘flat’ IDocs.

Chapter 4. Understanding the R/3 Link Format

The format of R/3 Link messages is shown in Figure 1 below:

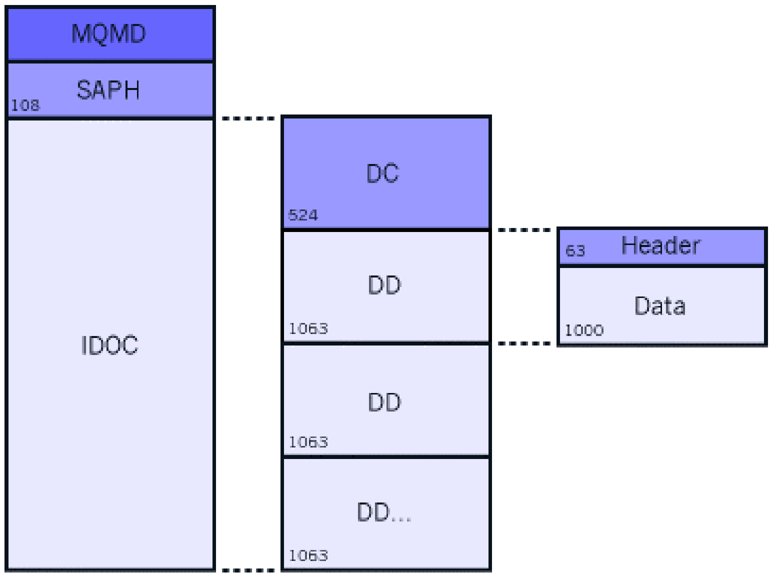


Figure 1. *Composition of an MQSeries Link for R/3 Message*

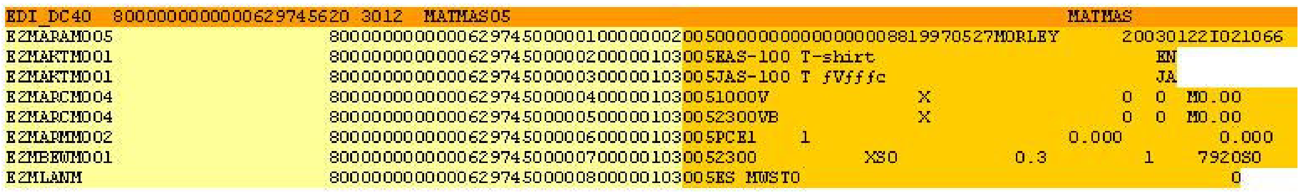
On the left-hand side you can see a logical breakdown of the WebSphere MQ message produced and consumed by the R/3 Link. It consists of an MQ Message Descriptor followed by a 108-byte SAP header and then the IDoc data itself. Note that the SAP header is not part of the IDoc — it is used exclusively by the R/3 Link and a dissection of this header is out of the scope of this Guide.

Breaking down the IDoc data we can see it consists of a number of fixed-length data blocks. *DC* is the IDoc Control Header, which is always 524 bytes long. Next comes 1 or more *DD* blocks. These represent the IDoc segments that make up the IDoc.

The DD blocks can be further broken down into a 63-byte header containing details about the segment such as its name and level in the document tree, and a 1000-byte data segment. Note that this data segment is always 1000 bytes in length and you will often see a lot of blank padding in segments on the wire.

The message format described above very closely matches the format of an IDoc as sent by SAP as a 'flattened' textual document.

A fragment of a flat MATMAS05 IDoc is shown below:



```
EDI_DC40  8000000000000629745620 3012  MATMAS05                                                MATMAS
EZMARAM005                    80000000000006297450000010000000200500000000000000000008819970527MORLEY    20030122I021066
EZMAKTM001                    800000000000062974500000200000103005EAS-100 T-shirt                                  EN
EZMAKTM001                    800000000000062974500000300000103005JAS-100 T ƒVƒƒc                                  JA
EZMARCM004                    8000000000000629745000004000001030051000V               X                 0  0  M0.00
EZMARCM004                    8000000000000629745000005000001030052300VB              X                 0  0  M0.00
EZMARMM002                    800000000000062974500000600000103005PCE1     1                      0.000           0.000
EZMBEWM001                    8000000000000629745000007000001030052300             XS0           0.3          1    7920S0
EZMLANM                       800000000000062974500000800000103005ES MWST0                                        0
```

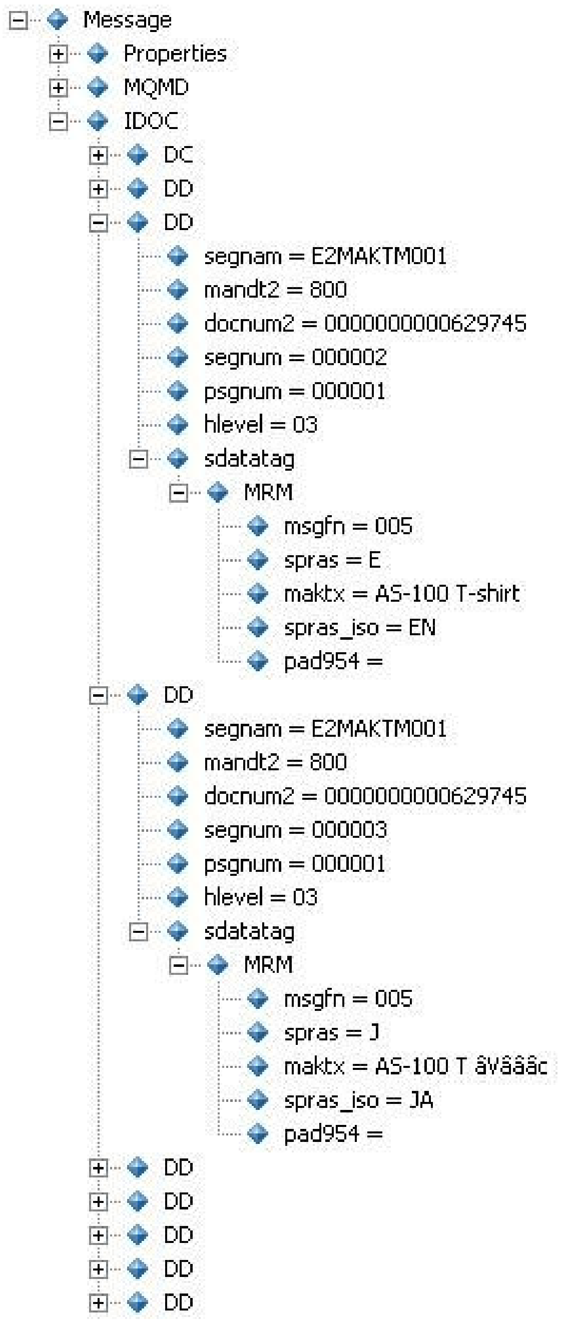
The various parts of the IDoc have been shaded. The top line shows the IDoc Control Header (DC) and is followed by the IDoc Data Segments (DDs), one per line. The lighter shading shows the DD headers containing the segment name, client, IDoc number, segment number, parent segment number and hierarchy level. The medium shading shows the DD data segments whose format depends on the segment type.

The only difference between this format and the one expected by the R/3 Link is regarding record delimiting: Segments within flat text IDocs are delimited by carriage returns whereas R/3 Link IDoc Segments are length-delimited. Thus, segments in the latter structure are always padded with spaces to 1063 bytes in length and do not contain carriage returns.

If you are using WebSphere Message Broker Version 6.0, the *IDocConverter* application is provided to create an R/3 Link format data file from a flat IDoc file.

The IDOC Message Tree

When the IDOC parser encounters a R/3 Link IDoc it attempts to parse it into DC and DD segments. An example of the resultant message tree is shown below:



```
Message
  Properties
  MQMD
  IDOC
    DC
    DD
    DD
      segnam = E2MAKTM001
      mandt2 = 800
      docnum2 = 0000000000629745
      segnum = 000002
      psgnum = 000001
      hlevel = 03
      sdatatag
        MRM
          msgfn = 005
          spras = E
          maktx = AS-100 T-shirt
          spras_iso = EN
          pad954 =
    DD
      segnam = E2MAKTM001
      mandt2 = 800
      docnum2 = 0000000000629745
      segnum = 000003
      psgnum = 000001
      hlevel = 03
      sdatatag
        MRM
          msgfn = 005
          spras = J
          maktx = AS-100 T âVâââc
          spras_iso = JA
          pad954 =
    DD
    DD
    DD
    DD
    DD
```

Note that the message tree's body consists of a top-level element called IDOC (as opposed to MRM or XML) and within that element there is a single DC child and a number of DD children.

A couple of the DD subtrees have been expanded, and here you can see the header fields containing important information such as the name of the segment and its position in the IDoc, followed by the *sdatatag* element. This is the element, common to all segments, under which the segment data itself lives. Because the IDOC parser invokes the MRM parser 'under the covers' to parse the segment data, the element in the message tree at the root of the segment data is called *MRM* and its immediate children represent the fields in the structure.

Some example ESQL field references for the IDOC built message tree are shown below:

```
SET idocNum   = InputRoot.IDOC.DC.docnum;
SET firstSeg  = InputRoot.IDOC.DD[1].segnam;
SET thirdLang = InputRoot.IDOC.DD[3].sdatatag.MRM.spras;
SET thirdText = InputRoot.IDOC.DD[3].sdatatag.MRM.maktx;
```

Note the slightly unusual use of the MRM tag in the middle of the last two field references, corresponding to the root of the segment data.

The MRM Message Tree

If you are using WebSphere Message Broker Version 6.1 and you have chosen to use the MRM parser, the resultant message tree is very similar to the message tree created by the IDOC parser above. In fact, the only differences are:

- The top-level element is instead called *MRM*.
- The element at the root of the segment data is no longer called *MRM*, but instead obtains its name from the value of the preceding DD *segnam* field.

Some example ESQL field references for the MRM built message tree are shown below:

```
SET idocNum   = InputRoot.MRM.DC.docnum;
SET firstSeg  = InputRoot.MRM.DD[1].segnam;
SET thirdLang = InputRoot.MRM.DD[3].sdatatag.E2MAKTM001.spras;
SET thirdText = InputRoot.MRM.DD[3].sdatatag.E2MAKTM001.maktx;
```

Chapter 5. Scenario walk through

In this Chapter we will walk through the steps needed to get a set of IDoc segment definitions out of SAP and into broker, and show them in use in a simple message flow.

We will use the following overall procedure (details below):

1. **Extract segment definitions from SAP**

Transaction WE60 can be used to extract all the segment definitions for a given IDoc to a C header file.

2. **Prepare C header for Message Broker Toolkit import**

If you are using WebSphere Message Broker Version 6.0, unfortunately the C header that results from Step 1 above is not in a format suitable for importing into a message set for use by the IDOC parser. The two main problems are:

- The header contains structures not supported by the Message Broker C importer (for example, preprocessor instructions and use of the C++ `Char` type instead of the `char` type)
- The structures need padding to 1000 bytes

The Java application *IDocHeaderTweak* provided with this package translates field names that have a reserved meaning in C++ and pads fields to the required length.

3. **Import the tweaked C header into the Message Broker Toolkit**

Now that the C header has been tweaked, we can import it into a message set with a CWF (IDOC parser) or TDS (MRM parser) physical format using a wizard or the *mqsicreatemsgdefs* command.

4. **Modify the Message Set's Message type names**

If you are using WebSphere Message Broker Version 6.0, when the C importer creates Message types from the C header we supply, it gives them names like *msg_e2maram005*, where *e2maram005* is the segment definition name. However, when the IDOC parser encounters a segment called *E2MARAM005*, it expects to find a matching Message type to help it parse the segment data.

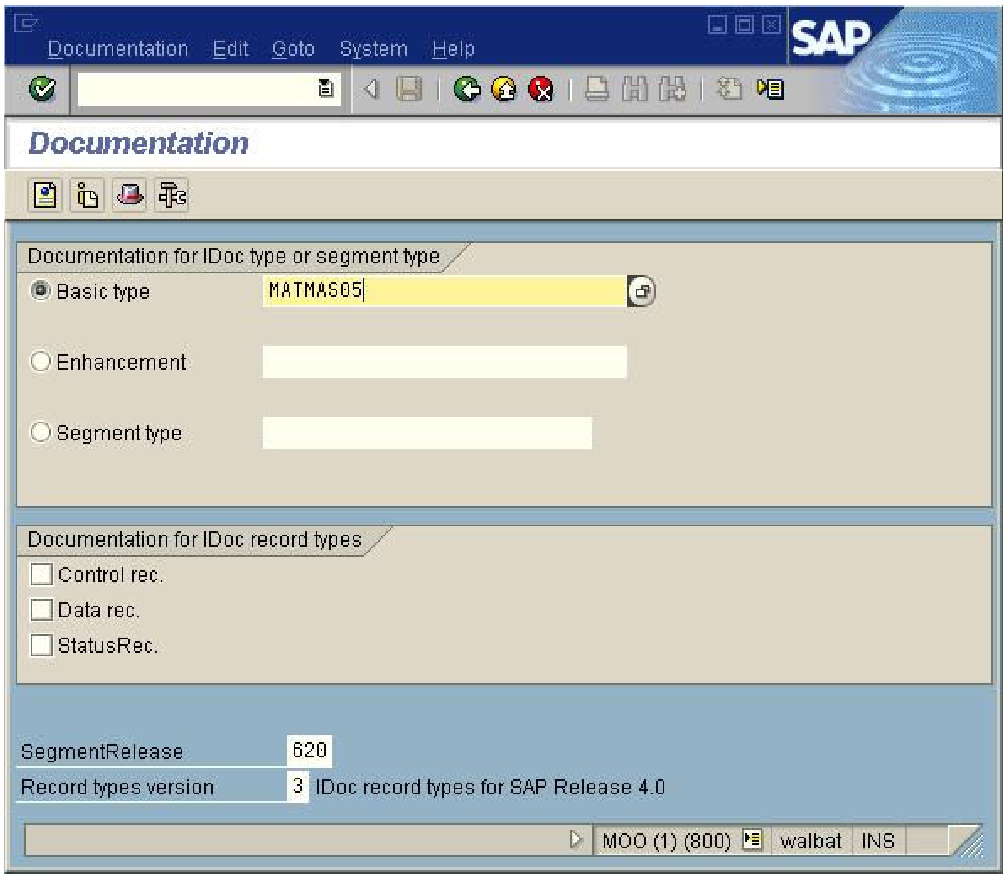
Thus, in order for the runtime parsing to work correctly, we need to change all Message types in our imported message set from *msg_e2xxx* to *E2XXX*. You can do this manually from within the toolkit, or use the supplied *IDocMsgSetTweak* application.

5. **Add the IDoc message definition skeleton**

Finally we add the IBM-supplied skeletal IDoc message definition to the message set using a wizard, and we are ready to use the message set with the IDOC parser or MRM parser in a message flow.

Detailed procedure

Log onto SAP using the SAP Logon GUI and run transaction **WE60**. Fill in the name of the basic IDoc type we wish to extract segment definitions for (in this example *MATMAS05*):



Documentation for IDoc type or segment type

Basic type [✓] MATMAS05

Enhancement

Segment type

Documentation for IDoc record types

☐ Control rec.

☐ Data rec.

☐ StatusRec.

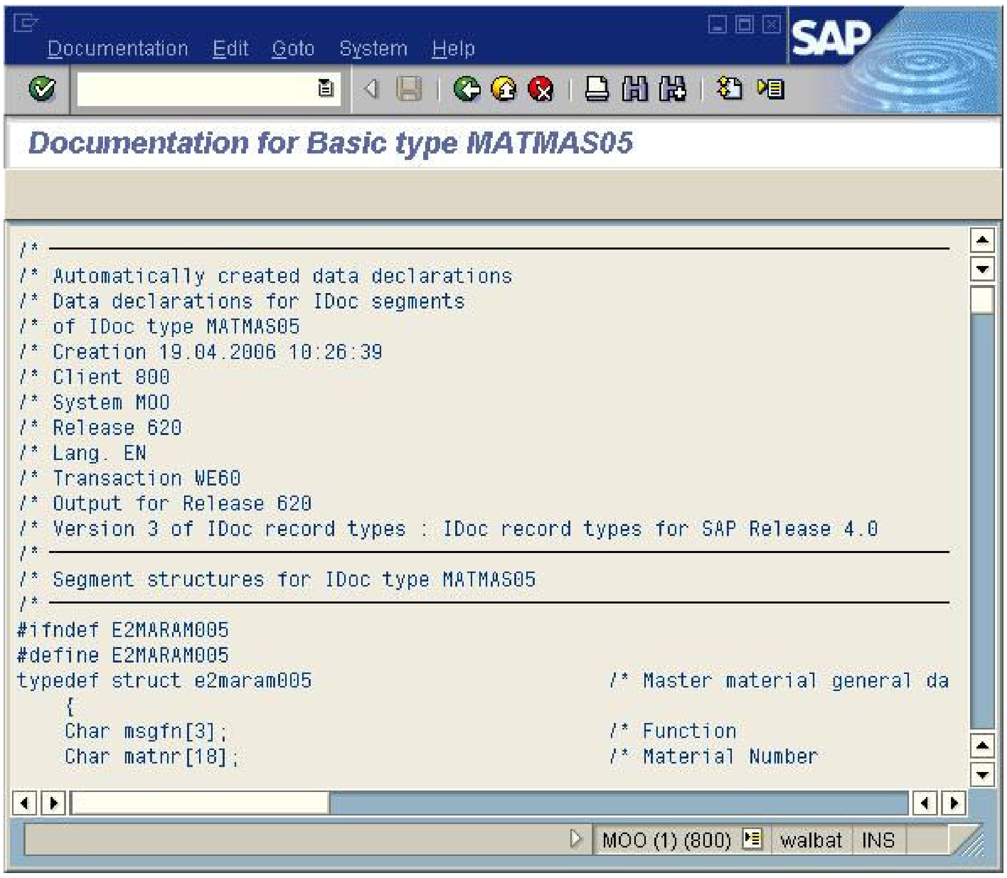
SegmentRelease 620

Record types version 3 IDoc record types for SAP Release 4.0

MOO (1) (800) walbat INS

Since the IDOC parser in WebSphere Message Broker Version 6.0 or 6.1 already knows about the Control and Data (i.e. segment header) records, we do not need to check any of the boxes in the *Documentation for IDoc record types* dialog.

Click on the hat icon or hit **F7** and after a short while a C header containing the structure definitions will be displayed:

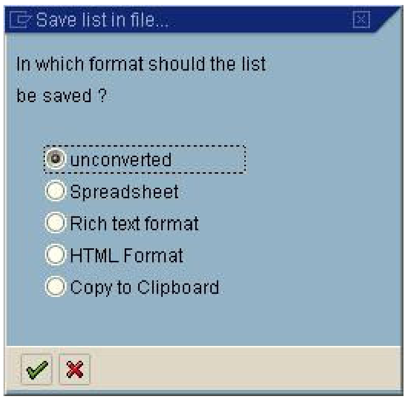


Documentation for Basic type MATMAS05

```
/* ─────────────────────────────────────────────
/* Automatically created data declarations
/* Data declarations for IDoc segments
/* of IDoc type MATMAS05
/* Creation 19.04.2006 10:26:39
/* Client 800
/* System MOO
/* Release 620
/* Lang. EN
/* Transaction WE60
/* Output for Release 620
/* Version 3 of IDoc record types : IDoc record types for SAP Release 4.0
/* ─────────────────────────────────────────────
/* Segment structures for IDoc type MATMAS05
/* ─────────────────────────────────────────────
#ifndef E2MARAM005
#define E2MARAM005
typedef struct e2maram005                         /* Master material general da
    {
    Char msgfn[3];                                /* Function
    Char matnr[18];                               /* Material Number
```

MOO (1) (800) walbat INS

Select the menu item **Systems–>List–>Save–>Local File** and a formatting dialog will be displayed:

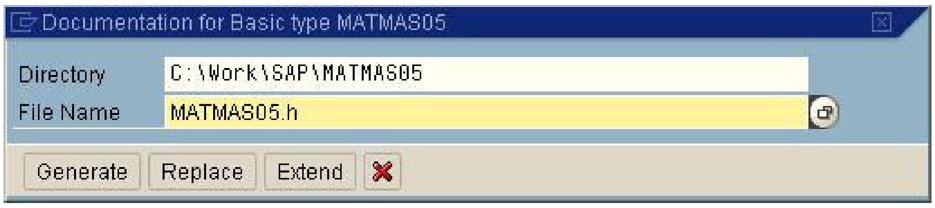

Save list in file...

In which format should the list be saved ?

[✓ unconverted]
Spreadsheet
Rich text format
HTML Format
Copy to Clipboard

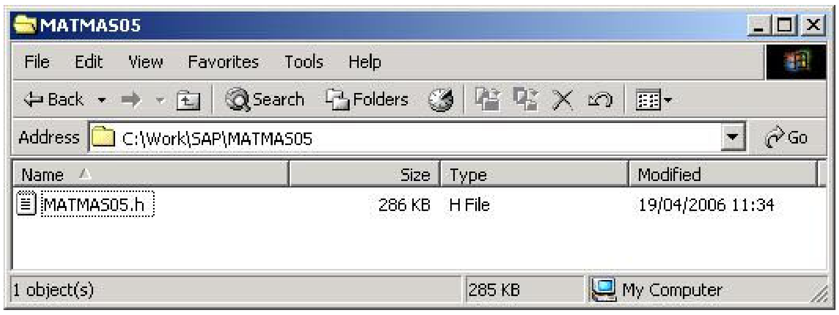
Accept the default option of *unconverted* and click on **Continue**. A dialog will be displayed in which you should enter the name of a directory and filename you wish to save the C header to.

Since the import we perform later uses an entire directory as its input, it is recommended that you supply a new directory or the name of an existing, empty one. You should also give the file a meaningful name and an extension of **.h**:


Documentation for Basic type MATMAS05

Directory	C:\Work\SAP\MATMAS05
File Name	MATMAS05.h

Confirm that the file has been saved to disk:

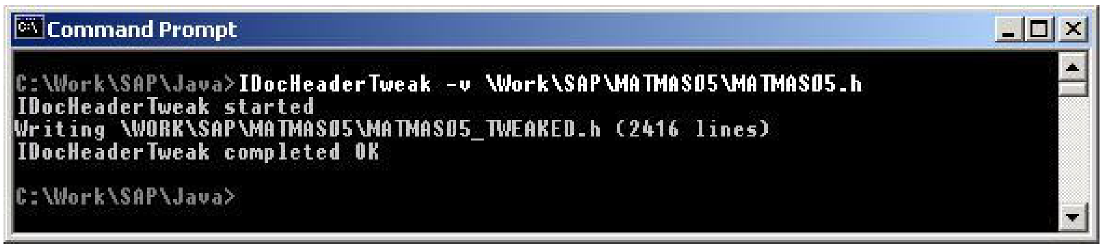

MATMAS05
File Edit View Favorites Tools Help
Back Search Folders
Address C:\Work\SAP\MATMAS05
Go

Name	Size	Type	Modified
MATMAS05.h	286 KB	H File	19/04/2006 11:34

1 object(s) 285 KB My Computer

If you are using WebSphere Message Broker Version 6.0, you are now ready to run the *IDocHeaderTweak* utility. This takes two parameters: the fully-qualified pathname of the C header file and the *–v* option which produces verbose output.

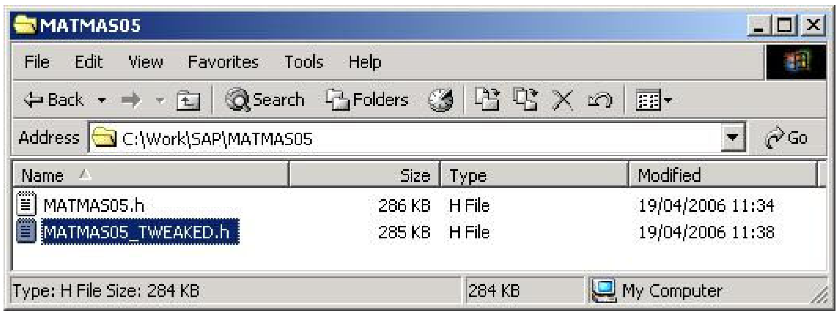
Here is an example of running the tool:



```
C:\Work\SAP\Java>IDocHeaderTweak -v \Work\SAP\MATMAS05\MATMAS05.h
IDocHeaderTweak started
Writing \WORK\SAP\MATMAS05\MATMAS05_TWEAKED.h (2416 lines)
IDocHeaderTweak completed OK

C:\Work\SAP\Java>
```

You will now see a new file created in the source directory, which has the name of the original with *_TWEAKED* appended:


MATMAS05
File Edit View Favorites Tools Help
Back Search Folders
Address C:\Work\SAP\MATMAS05
Go

Name	Size	Type	Modified
MATMAS05.h	286 KB	H File	19/04/2006 11:34
MATMAS05_TWEAKED.h	285 KB	H File	19/04/2006 11:38

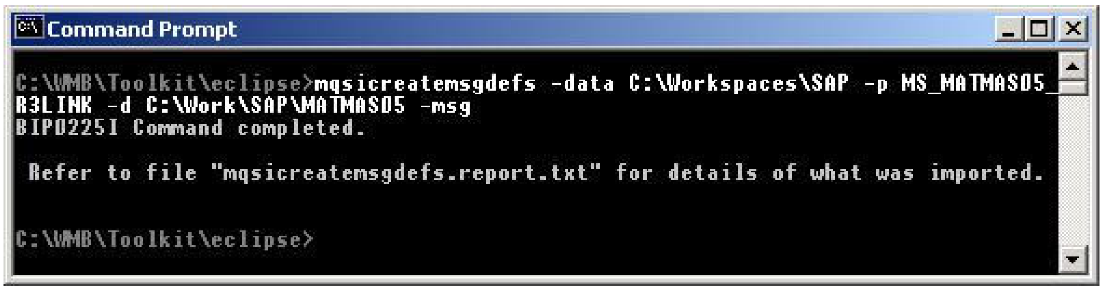
Type: H File Size: 284 KB 284 KB My Computer

Move or delete the original and, if you want, rename the new file to the name of the original.

If you are using WebSphere Message Broker Version 6.1, you do not need to run *IDocHeaderTweak*.

Before importing the tweaked header into the Toolkit workspace, it is necessary to create a message set with an appropriate physical format either to receive the definition directly, or to act as a template for a new message set that will receive the definition. If you are using the IDOC parser then a Custom Wire Format (CWF) is needed. If you are using the MRM parser (6.1 only) then a Tagged/Delimited String (TDS) format called 'Text_IDoc' is needed. The physical format is necessary in order to set the padding character to SPACE for all of the fields in the imported definition. Use the New Message Set wizard to create the message set and add the physical format. Then set the message set's 'Message domain' property to IDOC or MRM as required.

You are now ready to import the tweaked C header into the message set you have just created using either the New Message Definition File From C Header File wizard, or the *mqsicreatemsgdefs* utility. For example:



Command Prompt

```
C:\WMB\Toolkit\eclipse>mqsicreatemsgdefs -data C:\Workspaces\SAP -p MS_MATMAS05_
R3LINK -d C:\Work\SAP\MATMAS05 -msg
BIP0225I Command completed.

 Refer to file "mqsicreatemsgdefs.report.txt" for details of what was imported.


C:\WMB\Toolkit\eclipse>
```

If you use the *mqsicreatemsgdefs* command you must specify the –msg argument so that messages are created from imported structures.

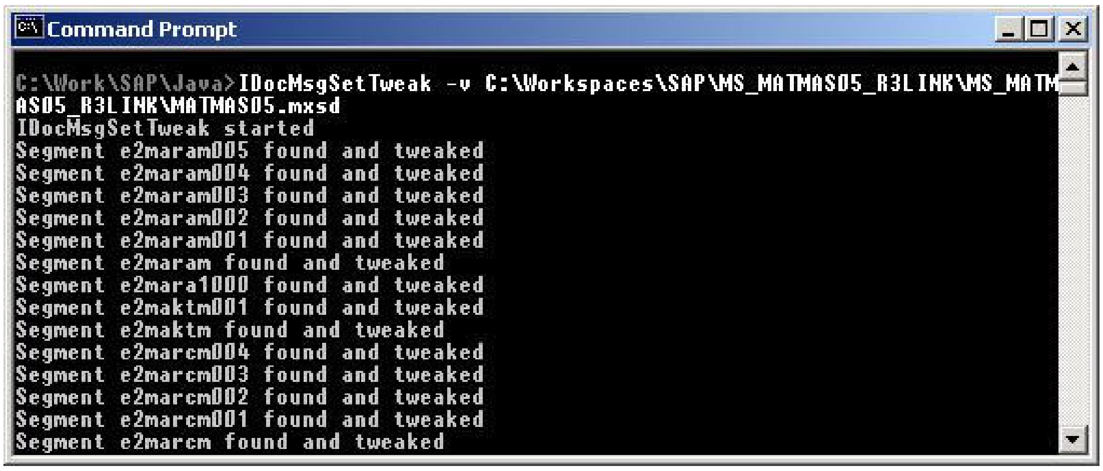
If you use the wizard you must check the boxes to create messages from imported structures, and on the last page of the wizard set the Padding Char option to ‘SPACE’ and select the String Encoding radio button ‘Fixed Length’.

Additionally if you are using WebSphere Message Broker Version 6.1:

If you use the *mqsicreatemsgdefs* command you must set the –opt argument to point at the IDocOptions610.xml file supplied with this SupportPac. If you have “flat” IDocs you must edit the options file and change the C PRE_PROCESSING_OPTION from “ale_idoc” to “file_idoc”.

If you use the wizard you must ensure that the pre-processing option on the first page is set to ‘SAP ALE IDoc’ for R/3 format IDocs or to ‘SAP File IDoc’ for “flat” IDocs.

If you are using WebSphere Message Broker Version 6.0, you now need to run the *IDocMsgSetTweak* tool to modify the Message type names in the message set generated in the previous step. This takes two parameters: the fully-qualified pathname of the message set (.mxsd) and the *–v* option.

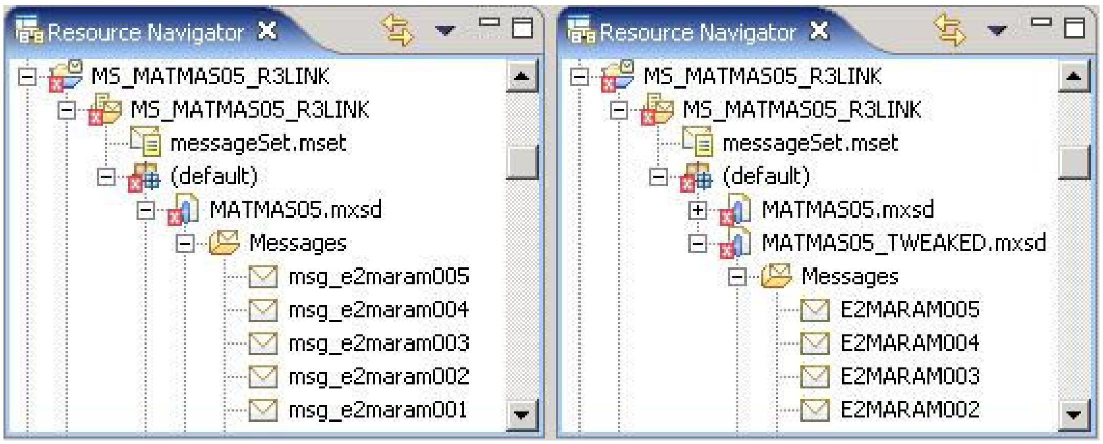


Command Prompt

```
C:\Work\SAP\Java>IDocMsgSetTweak -v C:\Workspaces\SAP\MS_MATMAS05_R3LINK\MS_MATM
AS05_R3LINK\MATMAS05.mxsd
IDocMsgSetTweak started
Segment e2maram005 found and tweaked
Segment e2maram004 found and tweaked
Segment e2maram003 found and tweaked
Segment e2maram002 found and tweaked
Segment e2maram001 found and tweaked
Segment e2maram found and tweaked
Segment e2mara1000 found and tweaked
Segment e2maktm001 found and tweaked
Segment e2maktm found and tweaked
Segment e2marcm004 found and tweaked
Segment e2marcm003 found and tweaked
Segment e2marcm002 found and tweaked
Segment e2marcm001 found and tweaked
Segment e2marcm found and tweaked
```

As you can see above, the tool generates messages as it finds and tweaks Message type definitions. Refresh the Message Set and you will see two message set files in your new project, along with a lot of errors about multiple message type definitions.

Before deleting the original MXSD file, take a look at the message type definitions in the two files:



You can see that the tweaking application has modified Message types called *msg_e2xxx* to *E2XXX*. You can now delete the original MXSD file, rename the new one to the original name, and all the errors should disappear.

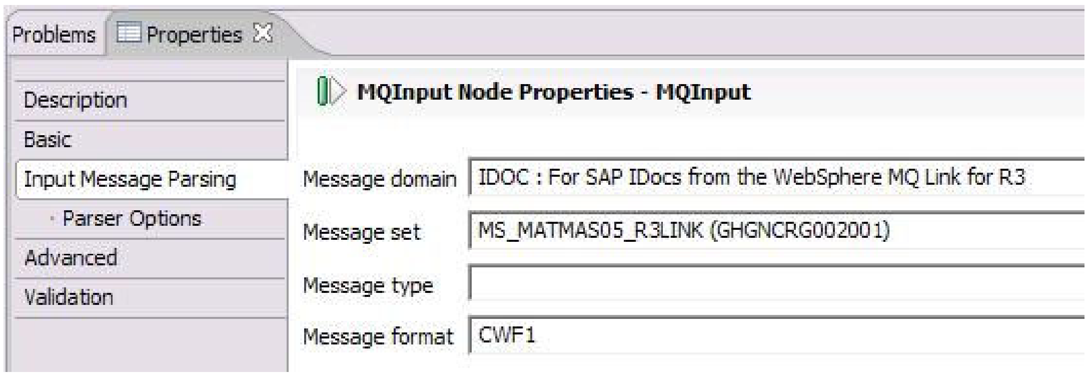
If you are using WebSphere Message Broker Version 6.1, you do not need to run *IDocMsgSetTweak*.

The message set now contains the correct MXSD for the application-specific parts of the IDoc structure. To complete the IDoc structure, you need to import the generic DC and DD parts. Use the New Message Definition File From IBM Supplied Message wizard to locate and import a skeletal IDoc message definition that models the DC and DD. If you are using WebSphere Message Broker Version 6.0 select file idoc.xsd. If you are using WebSphere Message Broker Version 6.1 select 'SAP ALE IDoc' or 'SAP File IDoc' as appropriate.

You can now use your new message set with the IDOC parser or (for MB Version 6.1 only) the MRM parser in a message flow. The simplest way to demonstrate this is to create a message flow consisting of an *MQInput* node wired to a *Trace* node which traces out *${Root}* to the filesystem.

The default properties of the MQInput node should be configured to expect IDocs and to use the message set you created above to resolve segment definitions:

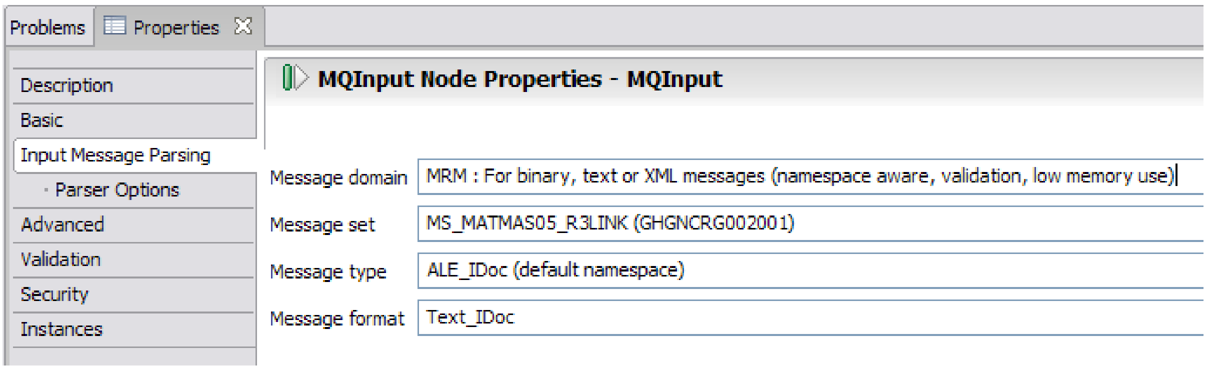
Example MQInput node settings for IDOC parser. Note that it is not necessary to set the Message type property:



MQInput Node Properties - MQInput

Message domain	IDOC : For SAP IDocs from the WebSphere MQ Link for R3
Message set	MS_MATMAS05_R3LINK (GHGNCRG002001)
Message type	
Message format	CWF1

Example MQInput node settings for MRM parser. The Message type property must be set to 'ALE_IDoc' or 'File_IDoc' as appropriate, and the Message format property must be set to 'Text_IDoc':



MQInput Node Properties - MQInput

Message domain	MRM : For binary, text or XML messages (namespace aware, validation, low memory use)
Message set	MS_MATMAS05_R3LINK (GHGNCRG002001)
Message type	ALE_IDoc (default namespace)
Message format	Text_IDoc