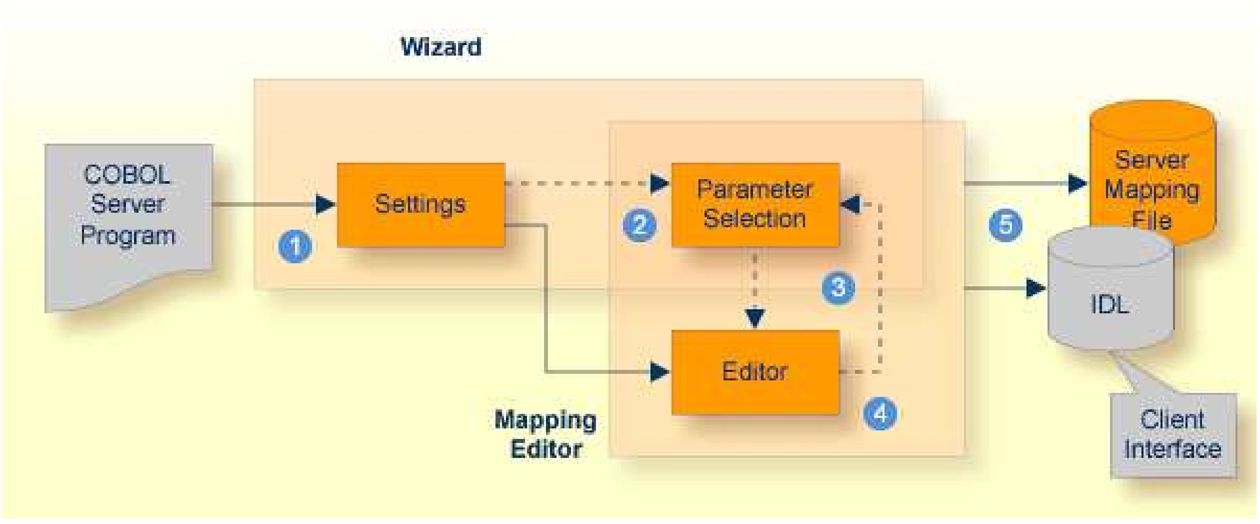
Introduction to the IDL Extractor for COBOL

- Introduction
- Extractor Wizard
- Mapping Editor
- Supported COBOL Interface Types
- Usage of Server Mapping Files

Introduction

The Software AG IDL Extractor for COBOL inspects a COBOL source and its copybooks for COBOL data items to extract. It can also extract directly from copybooks. In a user-driven process supported by an *Extractor Wizard*, the interface of a COBOL server is extracted and - with various features offered by a *Mapping Editor* - modelled to a client interface.



1. Start the wizard, select your server program and make COBOL-specific settings.
2. Optional. This step is not always necessary: it is possible that parameters have already been selected, for example as a result of the COBOL `USING` clause.
3. Optional. If necessary, you can modify the parameter selection from the Mapping Editor.
4. Fine-tune the COBOL to IDL mapping.
5. Generate an IDL file and a server mapping file. These two related files map the client interface to the COBOL server program and are described below:

- **IDL File**
  The Software AG IDL file (interface definition language) contains the modelled interface of the COBOL server. In a follow-up step the IDL file is the starting point for the RPC client-side wrapping generation tools to generate client interface objects. See *EntireX Wrappers*.

- **Server Mapping File**
  A server mapping file to complete the mapping is generated only if it is required by the RPC server during runtime to call the COBOL server. See *Usage of Server Mapping Files*.

Extractor Wizard

The extractor wizard guides you through the extraction process. The wizard supports the following tasks:

- Accessing COBOL source files, either in the local file system where the EntireX Workbench runs or remotely from the host computer with the RPC server extractor service. The wizard supports the following: z/OS partitioned data sets and CA Librarian data sets (including member archive levels) as well as BS2000/OSD LMS libraries. See *Extractor Service* in the z/OS Batch | IMS | BS2000/OSD Batch RPC Server documentation. For this purpose, define a local or remote COBOL extractor environment. See *IDL Extractor for COBOL Preferences*.

- Resolving of COBOL copybooks. If a relevant copybook from the COBOL `DATA DIVISON` is missing, a browse dialog is offered where you can locate the copybook - either a folder (local extractor environment) or data set (remote extractor environment) - interactively. Copybook folder or data sets can also be predefined in the COBOL extractor environment. See *IDL Extractor for COBOL Preferences*.

- Resolving of COBOL copybooks with the `REPLACE` option.

- CA Librarian (`-INC`) and CA Panvalet (`++INCLUDE`) control statements are supported. They are handled in a similar way to copybooks.

- Various COBOL server interface types, such as standard CICS `DFHCOMMAREA`, CICS with different structures on input and output, CICS with a large buffer compatible to webMethods WMTLSRVR, standard batch, Micro Focus standard calling conventions, and IMS BMP server with PCB pointers. See *Supported COBOL Interface Types*.

- Selecting the COBOL server interface manually within the *COBOL Mapping Editor* page. This allows you to extract from a COBOL server where the interface definition is not completely given by the parameters provided in the *`PROCEDURE DIVISION` Mapping*, making it impossible to detect the parameters automatically.

- Defining the default COBOL to IDL mapping in the *IDL Extractor for COBOL Preferences* for the following fields:

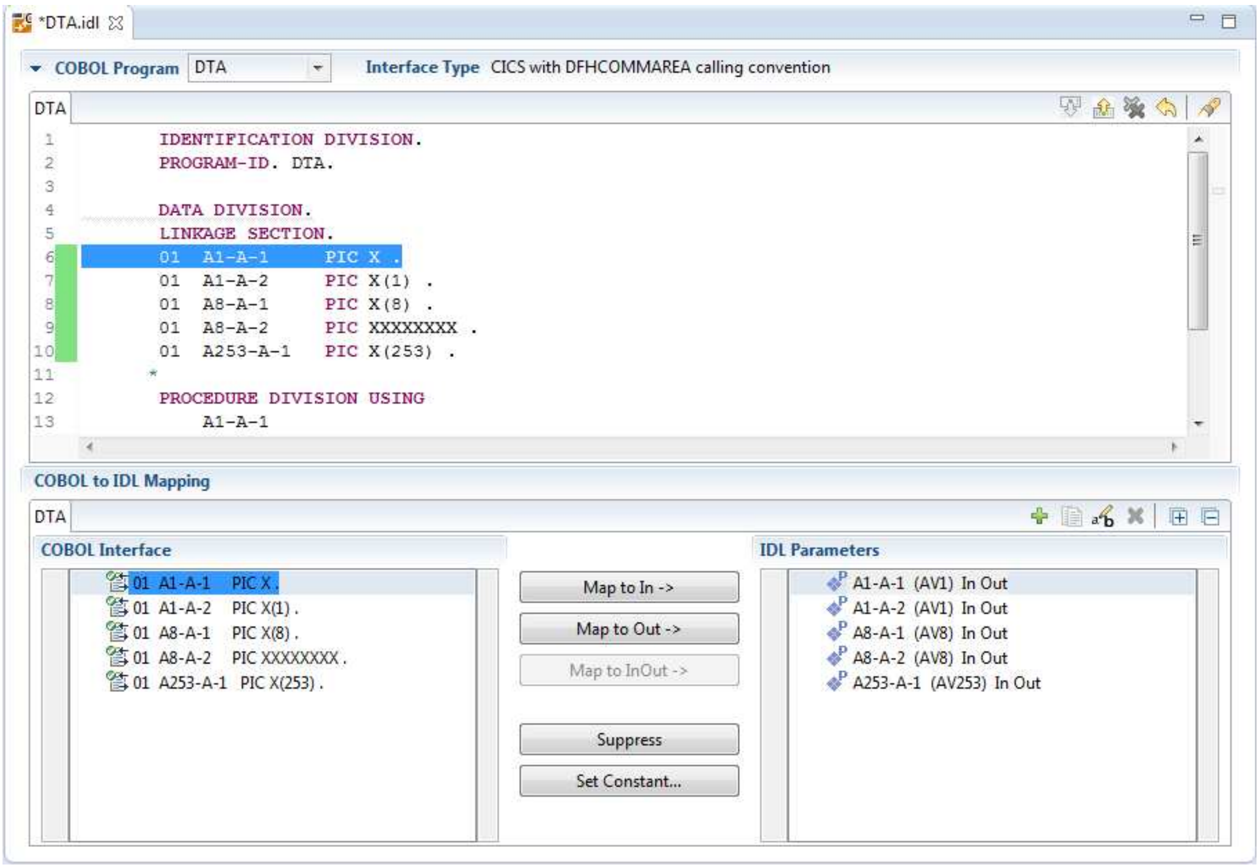
  - COBOL pseudo-parameter `FILLER` fields. You can define whether they should be part of the RPC client interface or not. By default, they are not contained in the IDL.

  - The name prefix for `FILLER` and anonymous groups used for IDL parameters.

  - COBOL alphanumeric fields (`PICTURE X`, `A`, `G`, `N`). They can be mapped either to variable-length or fixed-length strings in the IDL. This option is provided for modern RPC clients that support variable-length strings, and also for legacy RPC clients that support fixed-length strings only.

The extractor wizard is described in a step-by-step tutorial; see *Using the IDL Extractor for COBOL - Overview*.

Mapping Editor



```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. DTA.

       DATA DIVISION.
       LINKAGE SECTION.
       01  A1-A-1       PIC X .
       01  A1-A-2       PIC X(1)  .
       01  A8-A-1       PIC X(8)  .
       01  A8-A-2       PIC XXXXXXXX  .
       01  A253-A-1     PIC X(253)  .
      *
       PROCEDURE DIVISION USING
           A1-A-1
```

```
01 A1-A-1    PIC X .
01 A1-A-2    PIC X(1) .
01 A8-A-1    PIC X(8) .
01 A8-A-2    PIC XXXXXXXX .
01 A253-A-1  PIC X(253) .
```

```
A1-A-1  (AV1) In Out
A1-A-2  (AV1) In Out
A8-A-1  (AV8) In Out
A8-A-2  (AV8) In Out
A253-A-1  (AV253) In Out
```

The *COBOL Mapping Editor* is the tool to select and map the COBOL server interface to IDL. This section gives a short overview of the mapping features provided. These features are described in more detail in the documentation section for the respective interface type.

- Add and remove the parameters of the COBOL server in the top window of the COBOL Mapping Editor page. The current selection is shown in the bottom window for fine tuning.
- Provide IDL directions for parameters of the COBOL server. A COBOL server does not contain IDL direction information, so you can add this information manually in the Mapping Editor.
- Select `REDEFINE` paths used in the IDL. The Mapping Editor allows you to select a single `REDEFINE` path for every `REDEFINE` unit (all redefine paths addressing the same storage location).
- Suppress unneeded fields in the IDL. This keeps the IDL client interface lean and also minimizes the amount of data transferred during runtime.
- Define parameter constants as input for the COBOL server. Constant parameters are not contained in the IDL file, which means they are invisible for RPC clients. This makes the IDL client interface easier and safer to use, minimizing improper usage.
- For one COBOL server program, you can create and model multiple interfaces. If the IDL is processed further with a wrapper of the EntireX Workbench, the business functions are provided as

  - Web service operations if exposed as a Web service instead of a Web service with a single operation
  - methods if wrapped with the Java Wrapper or .NET Wrapper instead of a Java class with a single method
  - etc.

See *COBOL Mapping Editor* for more information.

Supported COBOL Interface Types

The IDL Extractor for COBOL supports as input a COBOL server with various interface types. This section covers the following topics:

- Supported CICS COBOL Interface Types
- Micro Focus with Standard Linkage Calling Convention
- Batch with Standard Linkage Calling Convention
- IMS MPP Message Interface (IMS Connect)
- IMS BMP with Standard Linkage Calling Convention
- What to do with other Interface Types?
- Compatibility between COBOL Interface Types and RPC Server

The interface type you are mostly working with can be set in the preferences. See *IDL Extractor for COBOL Preferences*.

Supported CICS COBOL Interface Types

Analyzing the technique used to access the interface with COBOL and CICS statements is the safest way to determine the interface type. The following CICS COBOL interface types are supported:

- *CICS with `DFHCOMMAREA` Calling Convention*
- *CICS with Channel Container Calling Convention*
- *CICS with `DFHCOMMAREA` Large Buffer Interface*

There is no clear and easy indication how to identify the interface type of a CICS COBOL server without COBOL and CICS knowledge. Below are some criteria that might help to determine the interface type. If you are unsure, consult a CICS COBOL specialist.

- The payload size of the CICS COBOL server is greater than 32 KB:
  - In this case it is *not* a DFHCOMMAREA interface, because the DFHCOMMAREA is limited to 32 KB.

    - It could be a large buffer or channel container interface, which are only limited by the storage (memory) available to them.

- The CICS COBOL server is located in a remote CICS region:

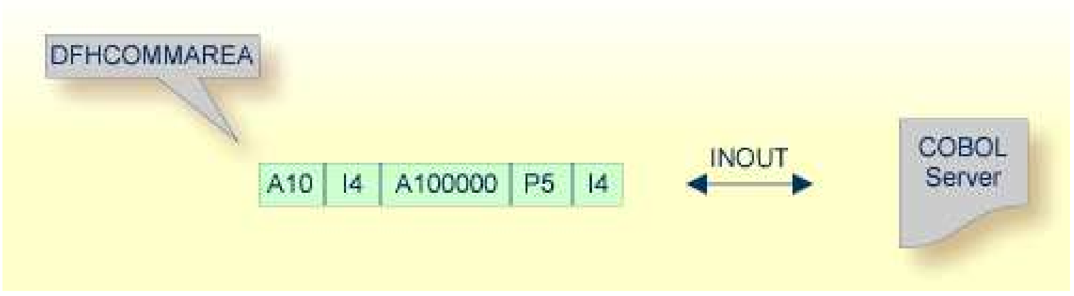
    - In this case it is *not* a large buffer interface (designed to assist with webMethods mainframe migration), because large buffer programs must reside on the same CICS region as the caller, that is, the CICS RPC Server (z/OS | z/VSE).

    - It could be a DFHCOMMAREA or channel container interface, which can reside in a remote CICS region.

Note:
The most used interface type is the DFHCOMMAREA interface. Large buffer and channel container interfaces are used much less frequently.

CICS with DFHCOMMAREA Calling Convention

The IDL Extractor for COBOL supports CICS programs using the standard `DFHCOMMAREA` calling convention.



The following illustrates roughly how you can determine whether a COBOL server follows the `DFHCOMMAREA` calling convention standard:

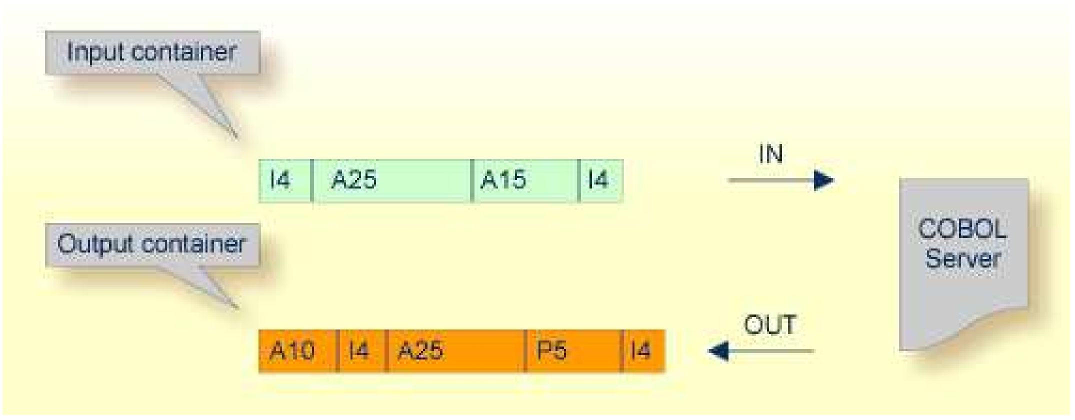
```
LINKAGE SECTION.
01 DFHCOMMAREA.
   02 OPERATION                      PIC X(1).
   02 OPERAND-1                      PIC S9(9) BINARY.
   02 OPERAND-2                      PIC S9(9) BINARY.
   02 FUNCTION-RESULT                PIC S9(9) BINARY.

PROCEDURE DIVISION USING DFHCOMMAREA.
   . . .
```

Most `DFHCOMMAREA` programs have a `DFHCOMMAREA` data item in their `LINKAGE SECTION` and may address this item in the `PROCEDURE DIVISION` header. If you find this in your COBOL source it's a clear indication it is a `DFHCOMMAREA` server program. But even if this is missing, it can be a `DFHCOMMAREA` program, because there are alternative programming styles. If you are unsure, consult a COBOL CICS specialist or see *Supported CICS COBOL Interface Types* for more information.

See *Step 4: Define the Extraction Settings and Start Extraction* for more information on extracting COBOL servers with this interface type.

CICS with Channel Container Calling Convention

The IDL Extractor for COBOL supports CICS programs using the channel container calling convention.



The following illustrates roughly how you can determine whether a COBOL server follows the Channel Container standard.

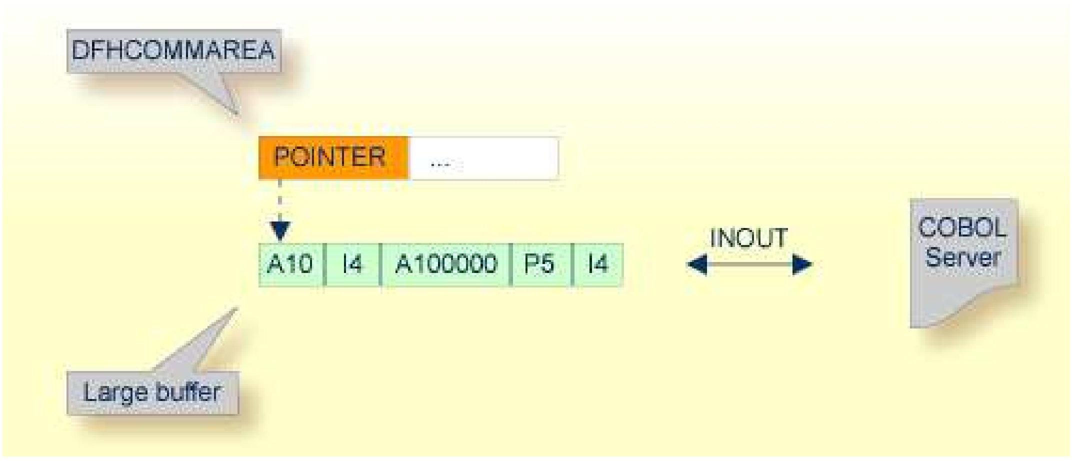
```
WORKING-STORAGE SECTION.
01 WS-CONTAINER-IN-NAME              PIC X(16) VALUE "CALC-IN".
01 WS-CONTAINER-OUT-NAME             PIC X(16) VALUE "CALC-OUT".
. . .
LINKAGE SECTION.
01 LS-CONTAINER-IN-LAYOUT.
   02 OPERATION                      PIC X(1).
   02 OPERAND1                       PIC S9(9) BINARY.
   02 OPERAND2                       PIC S9(9) BINARY.
01 LS-CONTAINER-OUT-LAYOUT.
   02 FUNCTION-RESULT                PIC S9(9) BINARY.

PROCEDURE DIVISION.
   . . .
   EXEC CICS GET CONTAINER (WS-CONTAINER-IN-NAME) SET (ADDRESS OF LS-CONTAINER-IN-LAYOUT) ...
   . . .
   EXEC CICS PUT CONTAINER (WS-CONTAINER-OUT-NAME) FROM (ADDRESS OF LS-CONTAINER-OUT-LAYOUT) ...
   . . .
```

Channel Container programs use `EXEC CICS GET CONTAINER` in their program body (`PROCEDURE DIVISION`) to read their input parameters. Output parameters are written using `EXEC CICS PUT CONTAINER`. There is no clear indication in the linkage or working storage section to identify a channel container program. If you are unsure, consult a COBOL CICS specialist for clarification.

See *Step 4: Define the Extraction Settings and Start Extraction* for more information on extracting COBOL servers with this interface type.

CICS with DFHCOMMAREA Large Buffer Interface

This type of program has a defined `DFHCOMMAREA` interface to access more than 31 KB of data in CICS. The interface is the same as the webMethods WMTLSRVR interface. This enables webMethods customers to migrate to EntireX.



Technically,

- the `DFHCOMMAREA` layout contains a structure with a *length* and a *pointer* to a large buffer. The following illustrates this:

  ```
  LINKAGE SECTION.
  01 DFHCOMMAREA.
     10 WM-LCB-MARKER                         PIC X(4).
     10 WM-LCB-INPUT-BUFFER                   POINTER.
     10 WM-LCB-INPUT-BUFFER-SIZE              PIC S9(8) BINARY.
     10 WM-LCB-OUTPUT-BUFFER                  POINTER.
     10 WM-LCB-OUTPUT-BUFFER-SIZE             PIC S9(8) BINARY.
     10 WM-LCB-FLAGS                          PIC X(1).
        88 WM-LCB-FREE-OUTPUT-BUFFER              VALUE 'F'.
     10 WM-LCB-RESERVED                       PIC X(3).
  01 INOUT-BUFFER.
     02 OPERATION                             PIC X(1).
     02 OPERAND-1                             PIC S9(9) BINARY.
     02 OPERAND-2                             PIC S9(9) BINARY.
     02 FUNCTION-RESULT                       PIC S9(9) BINARY.

  PROCEDURE DIVISION USING DFHCOMMAREA.
     . . .
     SET ADDRESS OF INOUT-BUFFER TO WM-LCB-INPUT-BUFFER.
     . . .
     SET ADDRESS OF INOUT-BUFFER TO WM-LCB-OUTPUT-BUFFER.
  ```

  The fields subordinated under `DFHCOMMAREA` prefixed with `WM-LCB` describe this structure. The field names themselves can be different, but the COBOL data types must match exactly.

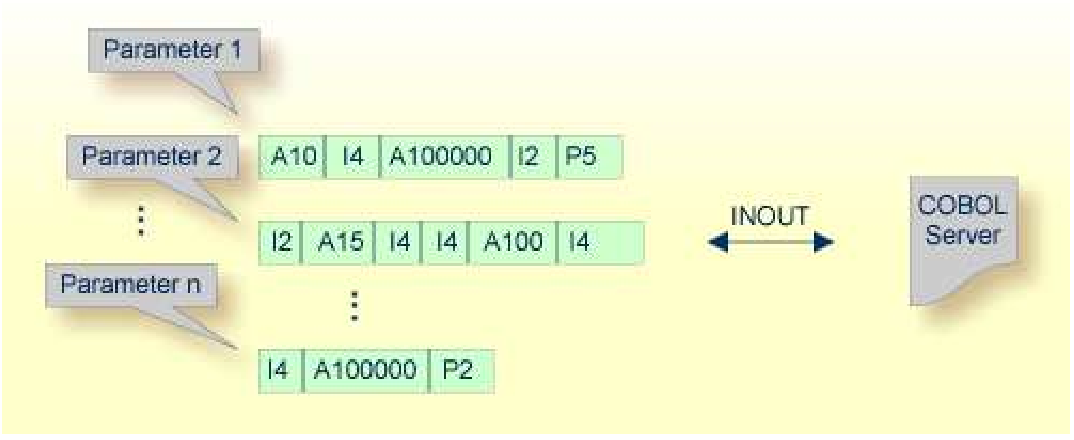
- data is described by separate structures, here `INOUT-BUFFER` in the linkage section.

  If you find this in your COBOL source, it's a clear indication it is a large buffer program. If you are unsure, consult a COBOL CICS specialist for clarification.

See *Step 4: Define the Extraction Settings and Start Extraction* for more information on extracting COBOL servers with this interface type.

Micro Focus with Standard Linkage Calling Convention

Standard call interfaces with a given number of parameters are supported. Every parameter addresses a fixed COBOL structure.



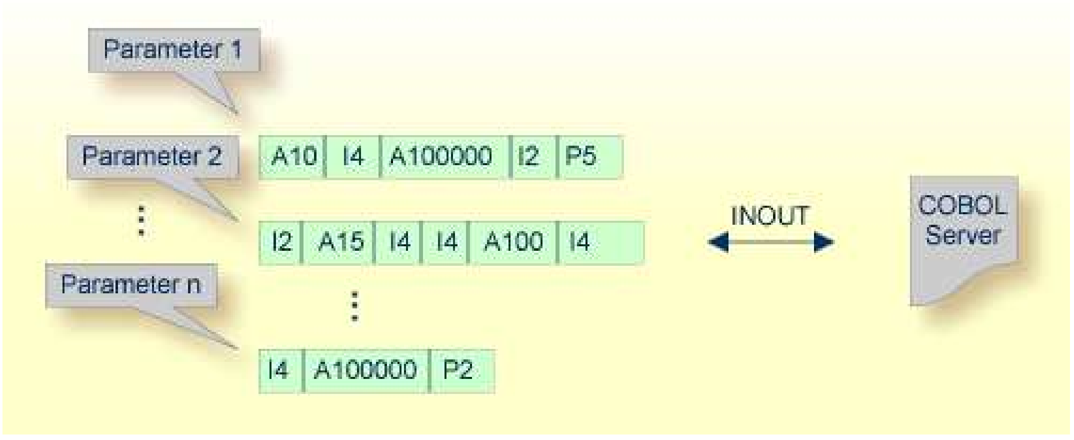
Technically, the generated COBOL server skeleton contains

- a parameter list: `PROCEDURE DIVISION USING PARM1 PARM2 ... PARMn`
- the parameters in the linkage section as COBOL data items on level 1

See *Step 4: Define the Extraction Settings and Start Extraction* and *Micro Focus with Standard Linkage Calling Convention* for more information on extracting COBOL servers with this interface type.

Batch with Standard Linkage Calling Convention

Standard call interfaces with a given number of parameters are supported. Every parameter addresses a fixed COBOL structure.

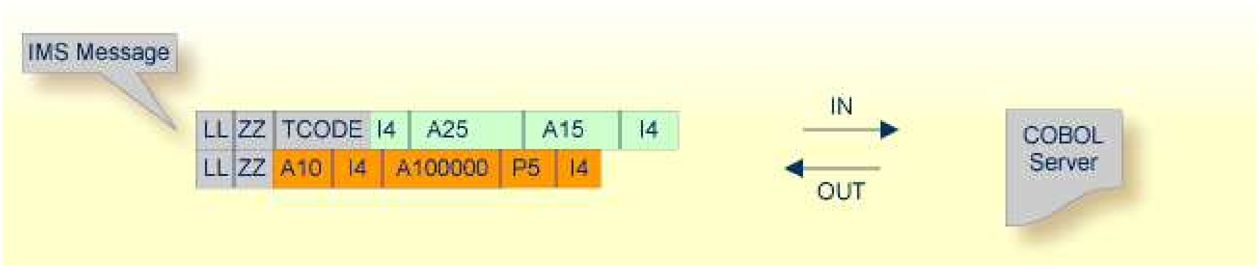


Technically, the COBOL server contains

- a parameter list: `PROCEDURE DIVISION USING PARM1 PARM2 ... PARMn`
- the parameters in the linkage section as COBOL data items on level 1

See *Step 4: Define the Extraction Settings and Start Extraction* and *Batch with Standard Linkage Calling Convention* for more information on extracting COBOL servers with this interface type.

IMS MPP Message Interface (IMS Connect)



IMS message processing programs (MPP) get their parameters through an IMS message and return the result by sending an output message to IMS. The IDL Extractor for COBOL enables extractions from such programs.

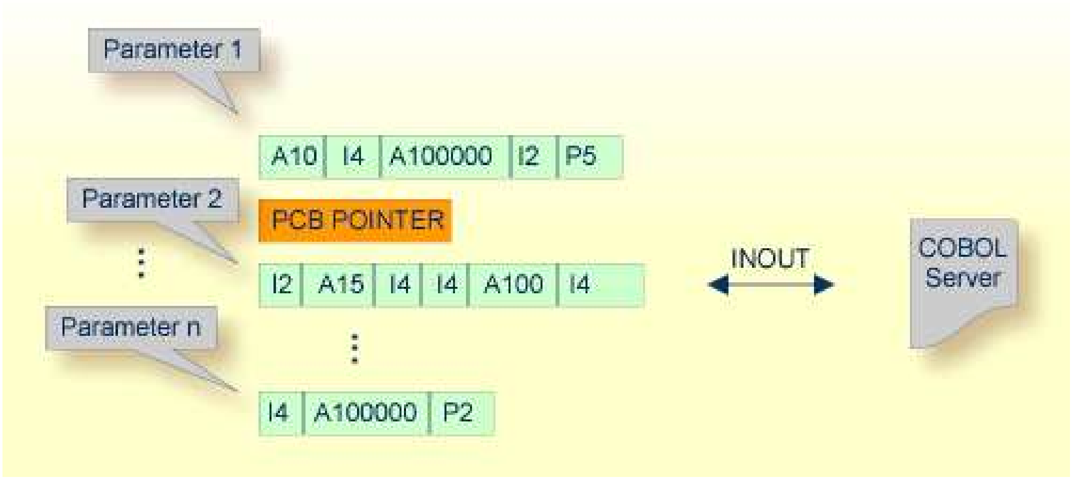
The COBOL server contains:

- a structure in the working storage section for the input and the output message.
- an IOPCB in the linkage section used to read input messages and write output messages using an IMS system call (i.e. `CALL "CBLTDLI"`).
- The message contains also technical fields specific to IMS (see fields `LL`, `ZZ` and `TRANCODE` in the picture above).

See *Step 4: Define the Extraction Settings and Start Extraction* and *IMS MPP Message Interface (IMS Connect)* for more information on extracting COBOL servers with this interface type.

IMS BMP with Standard Linkage Calling Convention

IMS batch message processing programs (BMP) with PCB parameters are directly supported. You have the option to specify a PSB list as input to the extractor to locate PCB parameters.



Technically, the COBOL server contains

- a parameter list: `PROCEDURE DIVISION USING PARM1 PCB PARM2 ... PARMn`
- IMS-specific *PCB pointers* within the parameter list
- the parameters in the linkage section as COBOL data items on level 1

See *Step 4: Define the Extraction Settings and Start Extraction* and *IMS BMP with Standard Linkage Calling Convention* for more information on extracting COBOL servers with this interface type.

What to do with other Interface Types?

Other interface types, for example CICS with non-DPL-enabled `DFHCOMMAREA`, can be supported by means of a custom wrapper. If you have to extract from such a COBOL server, proceed as follows:

1. Implement a custom wrapper, providing one of the supported interface types above.
2. Extract from this custom wrapper.

Compatibility between COBOL Interface Types and RPC Server

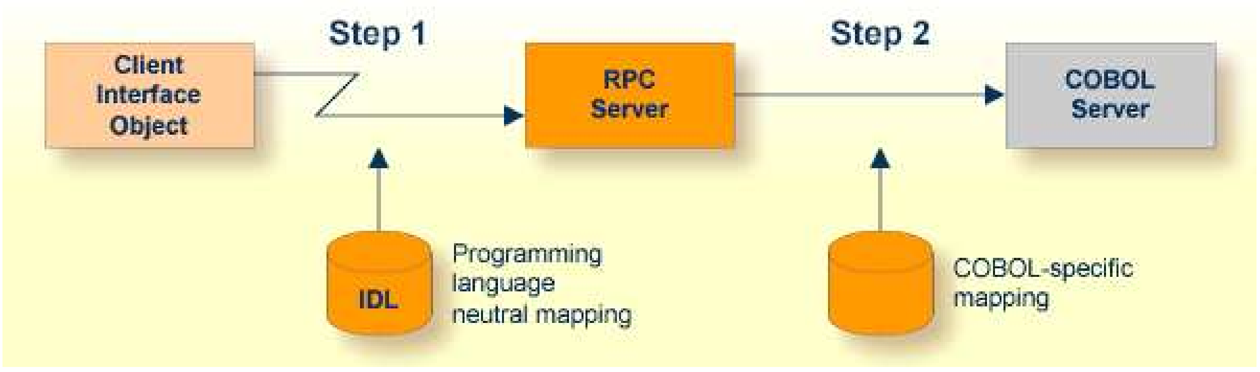
To call a server successfully, the RPC server used must support the interface type of the COBOL server. The table below gives an overview of possible combinations of an interface type and a supporting RPC server:

		Supported by RPC Server									
		z/OS			**UNIX/Windows**			**BS2000/OSD**	**z/VSE**	
Interface Type	**Supported by EntireX Adapter**	**CICS**	**Batch**	**IMS**	**CICS ECI**	**Micro Focus**	**IMS Connect**	**Batch**	**CICS**	**Batch**
CICS with `DFHCOMMAREA` Calling Convention (Extractor | Wrapper)	x	x			x				x	
CICS with `DFHCOMMAREA` Large Buffer Interface (Extractor | Wrapper)		x							x	
CICS with Channel Container Calling Convention (Extractor | Wrapper)		x								
Batch with Standard Linkage Calling Convention (Extractor | Wrapper)			x	x				x		x
Micro Focus with Standard Linkage Calling Convention (Extractor | Wrapper)						x				
IMS BMP with Standard Linkage Calling Convention (Extractor | Wrapper)				x						
IMS MPP Message Interface (IMS Connect) (Extractor)	x						x			

Usage of Server Mapping Files

There are many situations where the RPC server requires a server mapping file to correctly support special COBOL syntax such as `REDEFINES`, `SIGN LEADING` and `OCCURS DEPENDING ON` clauses, `LEVEL-88` fields, etc.

Server mapping files contain COBOL-specific mapping information that is not included in the IDL file, but is needed to successfully call the COBOL server program.



The RPC server marshals the data in a two-step process: the RPC request coming from the RPC client (Step 1) is completed with COBOL-specific mapping information taken from the server mapping file (Step 2). In this way the COBOL server can be called as expected.

The server mapping files are retrieved as a result of the *IDL Extractor for COBOL* extraction process and the *COBOL Wrapper* if a COBOL server is generated. See *When is a Server Mapping File Required?*.

There are *server*-side mapping files (*EntireX Workbench* files with extension .svm) and *client*-side mapping files (Workbench files with extension .cvm). See *Server Mapping Files for COBOL* and *How to Set the Type of Server Mapping Files*.

If you are using server-side mapping files, perform the following tasks:

- Customize the server-side mapping container. See *Server-side Mapping Files in the RPC Server* in the RPC server documentation for z/OS (CICS, Batch, IMS) | Micro Focus | CICS ECI | IMS Connect | BS2000/OSD | z/VSE (CICS, Batch).
- Deploy the files to the RPC server. See *Deploying Server-side Mapping Files to the RPC Server* in the RPC server documentation for z/OS (CICS, Batch, IMS) | Micro Focus | CICS ECI | IMS Connect | BS2000/OSD | z/VSE (CICS | Batch).

Note:
For IMS Connect and CICS ECI connections with the webMethods EntireX Adapter, server-side mapping files are not deployed. They are wrapped into the Integration Server connection - the same as client-side mapping files. For RPC connections, deployment to the target RPC server is mandatory. See the EntireX Adapter documentation under http://documentation.softwareag.com > *webMethods Product Line*.