

E61

OTMA API: An Alternative Access to IMS from S/390

Jack Yuan



Anaheim, California

October 23 - 27, 2000

Agenda

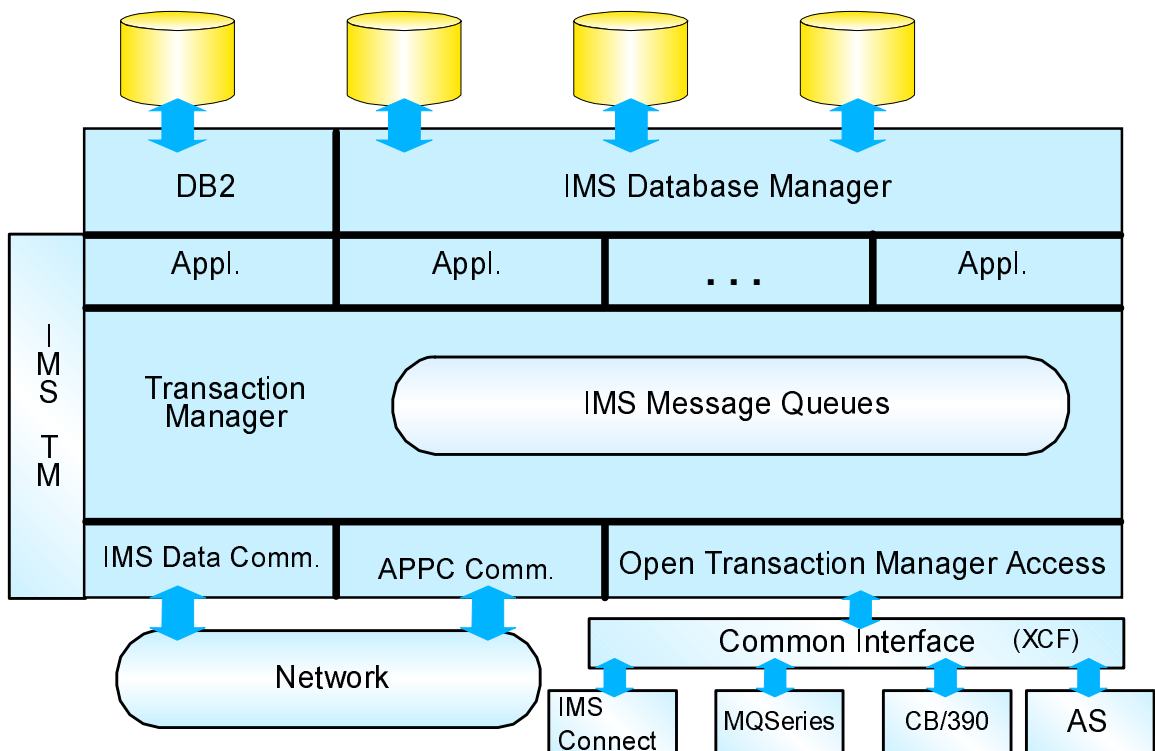
- What is OTMA?
- Problem/Solution
- OTMA C/I
- Functions & Restrictions
- Applications using C/I
- C/I Initialization and Stub
- API Parameters
- For C/C++ users
- For Cobol and Assembler users
- JAVA Wrapper
- REXX Wrapper
- Invocation Samples
- Security
- Tips & Considerations
- APARs needed
- Summary

Introduction

What is OTMA?

OTMA is a transaction based, connectionless client/server protocol.

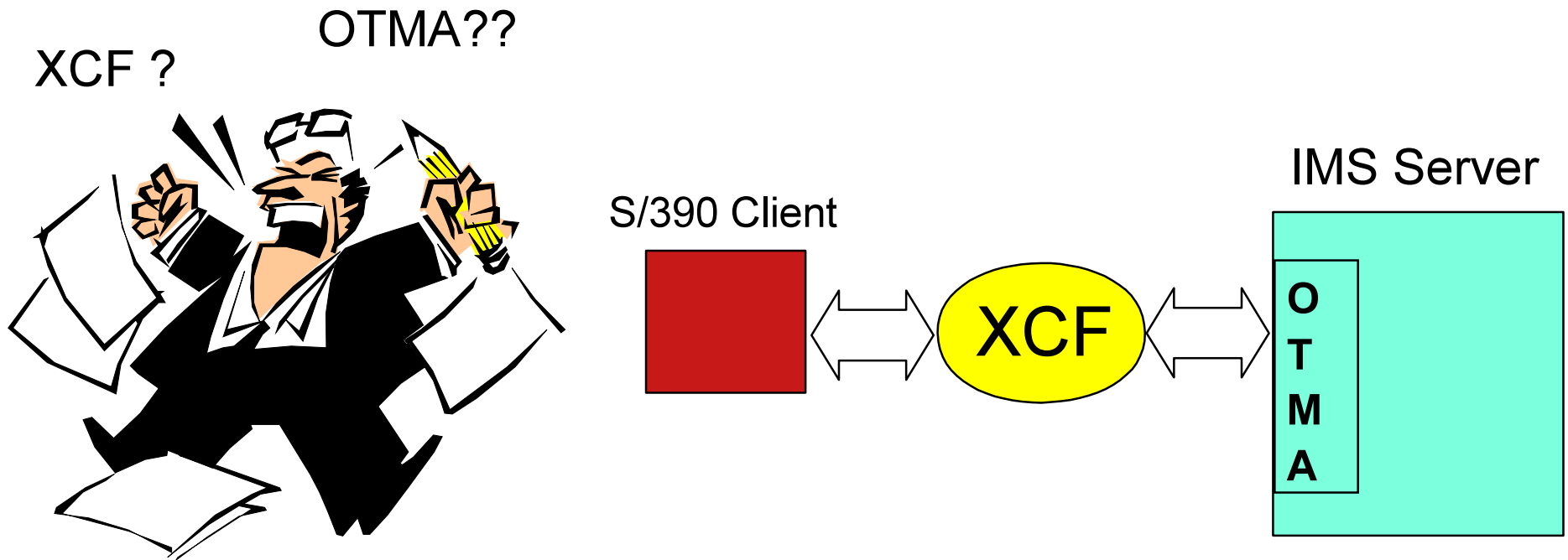
Communication to OTMA is over XCF.



Introduction...

Problem?

Problem: It has been difficult for customers to write their own programs into IMS via OTMA.



Introduction...

Solution

Solution: IMS V6 introduces the OTMA Callable Interface which is a high-level interface for easy access to IMS transactions and commands from other OS/390 subsystems.

No need to know
OTMA protocol!

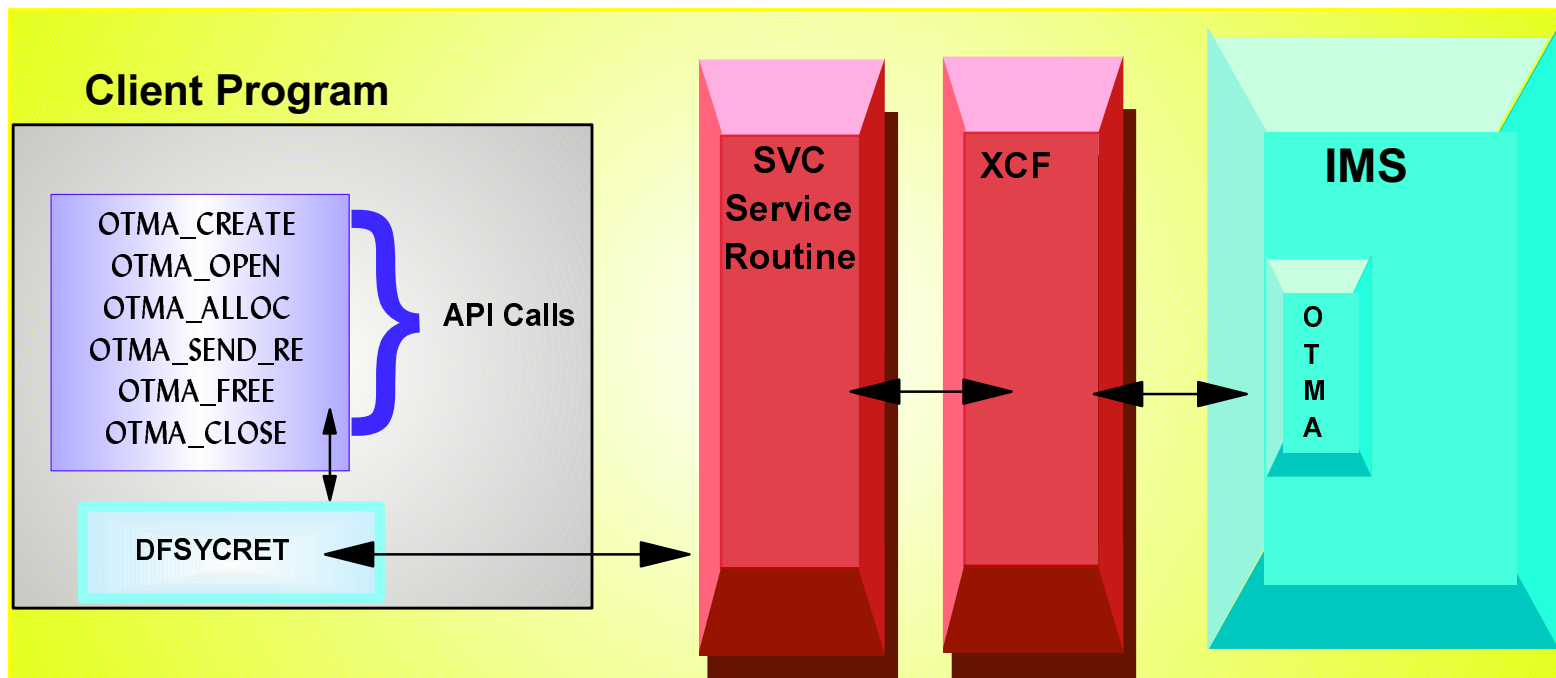


No need to understand
XCF macros!

Introduction...

OTMA Callable Interface

*OS/390
Environment*



OTMA Callable Interface API

- **otma_create** : create storage structures
- **otma_open** : establish a connection with IMS
- **otma_openx** : otma_open + IMS DFSYDRU0 exit name
- **otma_alloc** : create an independent session
- **otma_send_receive** : send data to IMS and receive output
- **otma_send_receivex** : otma_send_receive + otma user data
- **otma_free** : release the independent session

OTMA Callable Interface API...

- **otma_send_async** : send data to IMS W/O receiving IMS reply.
- **otma_receive_async** : receive an asynchronous or unsolicited message from IMS
- **otma_close** : end the connection with IMS

OTMA Callable Interface

Functions Supported

- Support execution of IMS transactons and commands
- Support unsolicited messages from IMS
- Support multi-segment messages in both directions
- Support message segments up to 32K
- Pass RRS context token to IMS
- Connect to multiple IMSs
- Support calls from authorized or unauthorized library
- Support preemptive abort of an uncompleted transaction

OTMA Callable Interface

Restrictions

- Need IMS V6 with APARs or IMS V7
- Can only run in OS/390 1.3 or above
- Does not support IMS OTMA resynchronization protocol and synchronous Tpipe
- Run IMS commands that OTMA supports
- No time-out feature. The user must create its own.

Potential Applications using C/I

- A monitor program sending messages to multiple IMS using C/I??
- A tool application sending IMS commands across SYSPLEX environment using C/I??
- A S/390 adapter communicating with IMS using C/I??
- Web serving from MVS web server??
- Volume testing without TPNS??
- Use like APPC to run a transaction on another IMS system without an MSC link??

OTMA C/I Initialization & Object Stub

- DFSYSVI0 must be run after MVS IPL to initialize OTMA Callable Interface SVC service.
 - ◆ Need to add an entry in MVS program property table (PPT) for the DFSYSVI0. Runs DFSYSVI0 as a stand-alone job.
- DFSYCRET, a load module containing entry points for each API call, needs to be linked with the application program.
 - ◆ Located in the V6 LOAD or V7 ADFSLOAD dataset.

For C and C++ Users

- DFSYC0.H header file defines the API calls.
 - ◆ Needs to be included in the C/C++ application program.
 - ◆ Located in the V6 GENLIB or V7 SDFSMAC dataset.
- Sample Programs are provided as follows:
 - ◆ A sample C/C++ program invoking C/I APIs
 - ◆ A sample wait routine
 - ◆ A sample JCL to compile/link the C/C++ program
 - ◆ A sample JCL to run the sample C/C++ program
 - ◆ A sample JCL to initialize the OTMA C/I

For Cobol and Assembler Callers

- Need to create an Assembler macro or Cobol procedure to map each API calls to an entry point routine in the OTMA C/I object stub, DFSYCRET (IMS does not provide the BAL macro or Cobol procedure yet.)
- `otma_create` -> `dfsycrct`,
- `otma_open`->`dfsyopn1`, `otma_openx` -> `dfsyopn2`,
- `otma_alloc`->`dfsyaloc`, `otma_free`->`dfsyfree`,
- `otma_send_receive`->`dfsysend`,
`otma_send_receivex`->`dfsysndx`,
- `otma_send_asnc`->`dfsysnda`,
`otma_receive_async`->`dfsyrcva`, `otma_close`->`dfsyclse`
(ignore `qinit` and `qget` API)
- See the DFSYC0 header file for C/C++ programs

JAVA Wrapper for C/I

- www.s390.ibm.com/nc/sntc/JAVAOTMA.html contains the thin Java/JNI wrapper around the OTMA Callable Interface
- Provides access to IMS through OTMA for Java applications and servlets with the OTMA C/I capabilities and semantics preserved
- OTMA Java classes: `otmaAdaptor()`, `otmaConnection()`, `otmaSession()`, `otmaInteraction()`, and `otmaException()`

JAVA Wrapper for C/I...

- JAVAOTMA Package
- www.s390.ibm.com/nc/sntc/JAVAOTMA.html
 - ▶ otmaXample.java is a sample application that illustrates the use of the Java OTMA classes; with some editing, it can be compiled into a working program. otmaHttplet.java contains the framework of a servlet program that uses Java OTMA.
- Featured in Redbook SG24-5619
- Java Programming Guide for OS/390
- Java OTMA servlet
demodemomvs.demopkg.ibm.com/jotma_inv.html

REXX Wrapper for C/I

- OTMA for OS/390 REXX
 - ▶ www.s390.ibm.com/nc/sntc/samples.html
 - ▶ OTMA WWWX routine in DB2IMSRX package

API Parameters for otma_create

anchor	output, the anchor word needed for all the subsequent API calls
return/reason code	output, return and reason code structure
ecb	input, event control block
group_name	input, XCF group name for IMS OTMA
member_name	input, XCF member name for your program
partner_name	input, XCF member name for IMS OTMA
session	input, number of sessions for the connection
tpipe_prefix	input, any 4-char (a-z, A-Z)

API Parameters for otma_open

anchor	output if create was not invoked first. It is input if create was called and anchor was returned.
return/reason code	output, return and reason code structure
ecb	input, event control block same as otma_create
group_name	input, XCF group name for IMS
member_name	input, XCF member name for your program
partner_name	input, XCF member name for IMS
session	input, number of sessions for the connection
tpipe_prefix	input, any 4-char (a-z, A-Z)

All the parameters for otma_open and otma_create are the same.

API Parameters for otma_openx

anchor	output if create was not invoked first. It is input if create was called and anchor was returned.
return/reason code	output, return and reason code structure
ecb	input, event control block
group_name	input, XCF group name for IMS
member_name	input, XCF member name for your program
partner_name	input, XCF member name for IMS
session	input, number of sessions for the connection
tpipe_prefix	input, any 4-char (a-z, A-Z)
IMS_dru_name	input, name of IMS OTMA destination resolution exit routine if any
special option	input, specify NULL

API Parameters for otma_alloc

anchor	input, the anchor word
return/reason code	output, return and reason code structure
session handle	output, session handle
special options	input, special options for the subsequent send API
IMS tran_code	input, IMS transaction name or first 8 bytes of IMS cmd with a "/"
RACF userid	input, RACF userid for the IMS input trans or command
RACF group name	input, RACF group name for the IMS input trans or command, blanks otherwise.

API Parameters for otma_send_receive

anchor	input, the anchor word
return/reason code	output, return and reason code structure
ecb	input, event control block
session handle	input, session handle from otma_alloc
lterm name	input, lterm override if any, blanks otherwise
mod name	input, modname if any, blanks otherwise
send buffer	input, input buffer
send length	input, length of send data in the input buffer
send segment list	input, input array for multi-segment input
receive buffer	output, output buffer
receive length	input, length of output buffer
receive segment list	output, output array for multi-segment output
received length	output, length of actual data received
context id	input, RRS context token
DFS message buffer	output, IMS DFS message received

API Parameters for otma_send_receive

anchor	input, the anchor word
return/reason code	output, return and reason code structure
ecb	input, event control block
session handle	input, session handle from otma_alloc
lterm name	input, lterm override if any, blanks otherwise
mod name	input, modname if any, blanks otherwise
send buffer	input, input buffer
send length	input, length of send data in the input buffer
send segment list	input, input array for multi-segment input
receive buffer	output, output buffer
receive length	input, length of output buffer
receive segment list	output, output array for multi-segment output
received length	output, length of actual data received
context id	input, RRS context token
DFS message buffer	output, IMS DFS message received
otma user data	input, 1K of any user data

API Parameters for otma_free

anchor	input, the anchor word
return/reason code	output, return and reason code structure
session handle	input, session handle


API Parameters for otma_send_async

anchor	input, the anchor word
return/reason code	output, return and reason code structure
ecb	input, event control block
tpipe name	input, 8-byte tpipe name. The first 4-char can not be the same as tpipe-prefix specified in otma_create api.
tran_name	input, 8-bytes trancode or IMS command
RACF userid	input, RACF userid for input
RACF group name	input, RACF groupname for input
lterm name	input, lterm override if any
modname	input, modname if any
optional user data	input, up to 1K of user data
input buffer	input, input data buffer
input data buffer length	input, length of input data
input segment list	input, input segment array
DFS message	output, IMS DFS message if any
special option	input, NULL

API Parameters for otma_receive_async

anchor	input, the anchor word
return/reason code	output, return and reason code structure
ecb	input, event control block
tpipe name	input, 8-byte tpipe name. The first 4-char can not be the same as tpipe-prefix specified in otma_create api.
lterm name	output, lterm override if any
modname	output, modname if any
optional user data	output, up to 1K of any user data
output buffer	output, output data buffer
output data buffer length	input, length of output data buffer
received data length	output, actual length of output data received
output segment list	output, output segment array
special option	input, NULL

API Parameters for otma_close



anchor	input, the anchor word
return/reason code	output, return and reason code structure

OTMA Callable Interface Invocation Samples

The basic usage

```
otma_create (...)  
Check RC  
otma_open (...)  
wait_on_ecb  
Check RC or Pose Code  
otma_alloc (...)  
Check RC  
otma_send_receive (...)  
wait_on_ecb  
Check RC or Pose Code  
otma_free (...)  
Check RC  
otma_close (...)  
Check RC
```

OTMA Callable Interface Invocation Samples

Connect to one IMS

```
otma_open (IMS)
```

```
otma_close (IMS)
```

Connect to two IMSs

```
otma_open (IMS5.1)  
otma_open (IMS6.1)
```

```
otma_close(IMS5.1)  
otma_close(IMS6.1)
```

OTMA Callable Interface

Invocation Samples

Connect to one IMS
send non-IMS conv trans

```
otma_open (IMS....)
otma_alloc (...)
otma_send_receive(...)
otma_free (...)

otma_close (IMS..)
```

Connect to one IMS
send two non-IMS conv.
trans to IMS

```
otma_open (IMS....)
otma_alloc (...)
otma_send_receive(...)
otma_free (...)

otma_alloc (...)
otma_send_receive(...)
otma_free (...)

otma_close (IMS..)
```

OTMA Callable Interface Invocation Samples

Send IMS conversational
transaction

```
otma_open (IMS....)
otma_alloc (...)
otma_send_receive(...)
otma_send_receive(...)
otma_send_receive(...)
otma_free (...)
otma_close (IMS..)
```

OTMA Callable Interface

Invocation Samples

Send one trans. input to IMS and receive output. Also receive async. output from IMS.

```
otma_open (IMS....)
otma_alloc (...)
otma_send_receive(...)
otma_free (...)

otma_receive_async(...)

otma_close (IMS..)
```

You need to change your IMS DFSYDRU0 user exit to specify the output TPIPE name so that the `otma_receive_async` API can obtain the IMS response from the TPIPE. (Invoking the `otma_receive_async` API requires that the user provides the TPIPE name in the parameter list.)

OTMA Callable Interface

Invocation Samples

Send-only to IMS

Method 1:

(using OTMA CM0)

```
otma_open (IMS....)
```

```
otma_send_async (...)
```

```
otma_close (IMS..)
```

Send-only to IMS

Method 2:

(using OTMA CM1)

```
otma_open (IMS....)
```

```
otma_send_receive(...)
```

Obtain RC=20,IMSDFS2082

```
otma_close (IMS..)
```

OTMA Callable Interface

Invocation Samples

Receive IMS queued output,
OTMA commit-then-send output,
or ALT-PCB output.

```
otma_open (IMS....)
Loop begin;
    otma_receive_async(...)
    wait for output
    process the data
Loop end;
otma_close (IMS..)
```

OTMA Callable Interface

Invocation Samples

Send one input and receive
3 IMS queued output.

```
otma_open (IMS....)
otma_send_async (...)
otma_receive_async (...)
otma_receive_async (...)
otma_receive_async (...)
otma_close (IMS..)
```

OTMA Callable Interface

Invocation Samples

Send one input and receive
a "BIG" output.

```
otma_open (IMS1....)
```

```
otma_alloc (...)
```

```
otma_send_receive (...)
```

```
ECB posted with a bad RC/RSN code
```

```
otma_free (...)
```

```
/* RSN(3) indicates the required storage */
```

```
/* for receiving the complete output */
```

```
Allocate storage for the output
```

```
/* Resubmit the message to IMS */
```

```
otma_alloc (...)
```

```
otma_send_receive (...)
```

```
otma_free (...)
```

```
otma_close (IMS1..)
```

OTMA Callable Interface

Tips

- **Errors related to OTMA C/I initialization:**
 - If C/I has not been installed on the MVS system, a F92 abend will occur when any API is issued.
 - A DFS3911E error message will occur if C/I is not installed properly.

OTMA Callable Interface

Tips

- `otma_open`, `otma_send_receive`, `otma_send_async`, and `otma_receive_async` each has an ECB parameter. This ECB is posted by the function or by an SRB routine that the function precipitates. The caller must check the ECB and wait for it to be posted before inspecting the return code and output fields.
- The ECB parameter needs to be initialized with 0 before passing to the API.

OTMA Callable Interface

Tips

- `otma_open` : establish the XCF connection with IMS
 - ▶ The parameters for `otma_open` and `otma_create` are identical.
 - ▶ Limit the number of client XCF names created in your program to save storage. You could re-use the same name.
 - ▶ If `otma_open` is invoked without issuing `otma_create`, ensure that the input anchor word is initialized with 0.
 - ▶ You may use our sample wait routine, `DFSYCWAT`, to wait on ECB.
 - ▶ You need to specify a large session number to increase the session usage.

OTMA Callable Interface

Tips

Try NOT to issue open and close often to have better performance.

```
otma_open
...
otma_send_receive
...
otma_close

otma_open
...
otma_send_receive
...
otma_close
```

OK

```
otma_open
..
otma_send_receive
..
otma_send_receive
..
otma_close
```

Better

OTMA Callable Interface

Tips

- **otma_alloc** : create an independent session for the subsequent **otma_send_receive** call.
 - ▶ Specify the name of IMS transaction or command to be sent to the IMS. The maximum length of the name is 8 characters. If blanks or NULL is specified in the input parameter, the transaction name/command needs to be in the beginning of the send buffer of **otma_send_receive** API.
 - ▶ Retry can be made for receiving the "session limit reached" error.

OTMA Callable Interface

Tips

- **otma_free : free the independent session.**
 - ▶ otma_free is needed for every otma_send_receive invocation except for the IMS conversation transaction.
 - ▶ invoke otma_free to perform the pre-emptive abort function for a long-running transaction.

OTMA Callable Interface

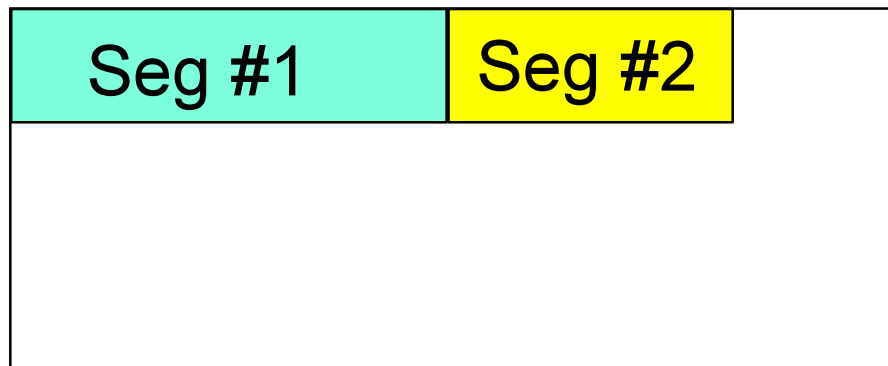
Tips

- The standard IMS LLZZ + trancode + one blank + data will be built by OTMA C/I. OTMA C/I obtains the trancode from `otma_alloc` call. The "data" is obtained from the `otma_send_receive` API.
- If no trancode is provided in the `otma_alloc` API, C/I expects that the input data parameter of the `otma_send_receive` API includes the trancode + (blank(s)) + data.
- The IMS LLZZ is always built by OTMA C/I.

OTMA Callable Interface

Tips

- The multi-segment input/output in the send/receive data buffer for `otma_send_receive` API:



no LLZZ for segments in the buffer, but LL needs to be included in the input/output array.

OTMA Callable Interface

Tips

For `otma_send_receive` API, the multi-segment input/output array specifies the # of the segments and the length of each segment.

Input/output array

of segments
Length of seg #1
Length of seg #2

Example

2
10
20

OTMA Callable Interface

Tips

For multi-segment **input** message to IMS

input buffer

seg 1
seg 2

input array

2
5
15

output array

5

The first element needs to be initialized with
(the size of array - 1)

OTMA Callable Interface

Tips

For multi-segment **output** message from IMS

output buffer

seg 1
seg 2
seg 3
seg 4
seg 5

input array

2
5
15

output array

5
6
18
10
3
8

Input array will not be changed.

If more than 5 segments received, rest of the segments will be shoved into the segment 5.

OTMA Callable Interface Tips

Review the sample programs first to help you use the C/I APIs.

Save/log the bad return/reason codes from C/I for easy debugging the problem.

After a C/I fix is installed, C/I needs to be refreshed by running DFSYSVI0. It would be nice to reset the client program too.

OTMA Callable Interface

Security

- **For callers from authorized library:**
 - ▶ Input will be passed to IMS OTMA for security checking.
- **For callers from unauthorized library:**
 - ▶ To protect access to the XCF group from any unauthorized callers, RACF facility resource class IMSXCF.OTMACI can be defined. During otma_open call, RACF RACHECK will be performed.
 - ▶ The input userid will be ignored by OTMA C/I for the IMS transaction/command input. OTMA C/I will invoke RACF routine to extract UTOKEN for the user and pass it to IMS OTMA for the security checking.

OTMA Callable Interface

Considerations

■ C/I initialization

- ▶ If the MVS is re-IPLed, the OTMA C/I needs to be re-initialized again by running the DFSYSVI0 JCL.

■ IMS Storage

- ▶ One C/I session results one tpipe in IMS. Tpipes consume IMS storage. Use `otma_free` to free the session so that the tpipe can be re-used.

■ SVC 146

- ▶ SVC 146 is used for unauthorized callers and authorized callers. SVC 146 itself issues several other MVS system calls.
- ▶ Additional validity checking for system integrity reasons and data copy from client storage to key 7 protected storage are performed.

OTMA Callable Interface

APAR needed

- For IMS V6,
 - ▶ Pre-conditioning APARs: V6 PQ19424 and PQ20680
 - ▶ Basic support: V6 PQ17203
 - ▶ Async support: V6 PQ32398
 - ▶ Recommended APAR level: V6 PQ39044
- For IMS V7, C/I is in the base.

OTMA Callable Interface Summary

- IMS OTMA Callable Interface provides a simple and easy-to-use API which abstracts out the details of OTMA and XCF.
- Invokers can submit IMS transactions or commands from within the OS/390 environment.
- It requires IMS V6 APARs, OS/390 1.3, and initialization procedure.
- www.software.ibm.com/data/ims/otmaci.html or Appendix D of the OTMA Guide and Reference