IBM Software Group

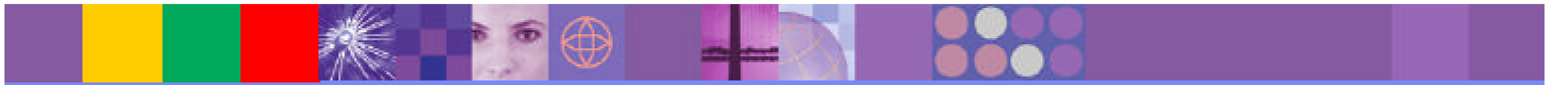
2005 B2B Customer Conference

Pioneering New Horizons – Solutions that Evolve

Using WDI in a Real Time Environment

David Hixon

WebSphere. software



ON DEMAND BUSINESS

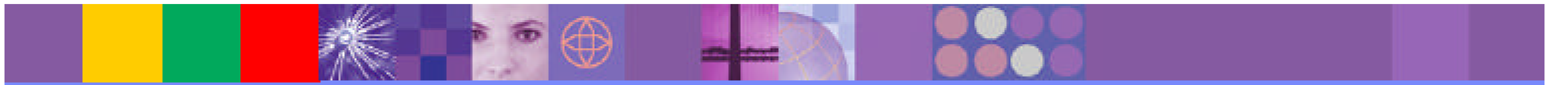
Agenda

- **Methods of using WDI in a real time environment**
  - **MQ Triggering**
    - **WDI Adapter and the multi-threaded adapter**
  - **CICS**
    - **WDI as a transaction**
    - **Hot DI**
  - **APIs**
    - **C++ and Java**
    - **C**
- **Case study – Danny Robbins of BNSF**

IBM Software Group

Real Time Options for WebSphere MQ

WebSphere software

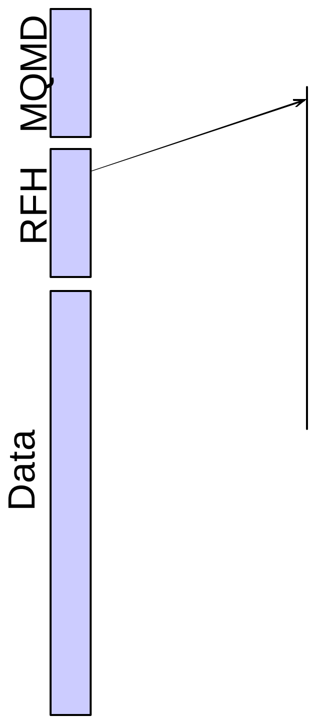


ON DEMAND BUSINESS

Command Substitution

- Information from the RFH2 header can be used to specify parameters in a PERFORM command in a Service Profile.
  - Perform command arguments are put into the <usr> folder of the RFH2 header.
  - &TagName is used on the PERFORM COMMAND to substitute the value from the RFH2 header.

Command Substitution



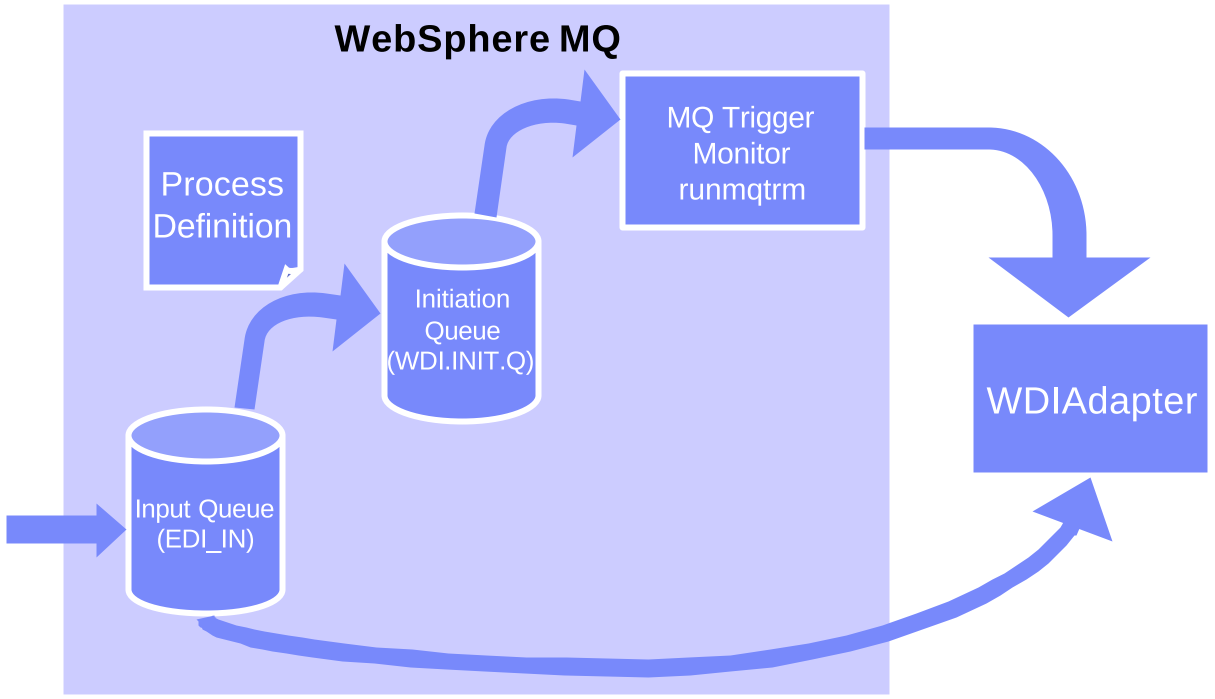
The following folder names are defined for use by WebSphere products:

<mcd>	Message content descriptor
<psc>	Publish/subscribe command
<pscr>	Publish/subscribe command response
<usr>	**<fmtid>POHD1</fmtid>**

Command Substitution

- PERFORM TRANSFORM WHERE DOCID(&fmtid)
  - &fmtid will be replaced with the value from the element in the user folder

WDIAdapter Overview

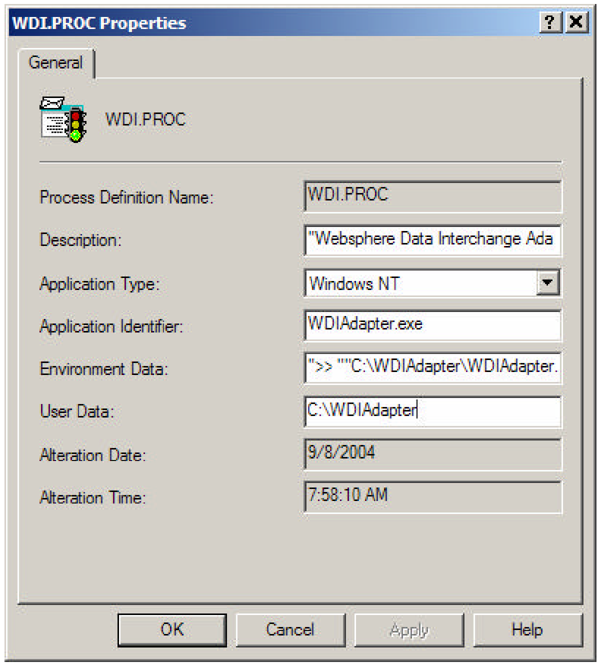


Setting up MQ Objects for WDIAdapter

- Setup Script “wdimqcommands.txt”
  - Process Definition – WDI.PROC
  - Initiation Queue – WDI.INIT.Q
  - Default Input/Output queues
- Execute script with runmqsc command
  - runmqsc < wdimqcommands.txt

Process Definition

- WDI.PROC
  - ➢Environment Data
  - ➢User Data



Process Definition Name:	WDI.PROC
Description:	"Websphere Data Interchange Ada
Application Type:	Windows NT
Application Identifier:	WDIAdapter.exe
Environment Data:	">> ""C:\WDIAdapter\WDIAdapter.
User Data:	C:\WDIAdapter
Alteration Date:	9/8/2004
Alteration Time:	7:58:10 AM

Configuring WDIAdapter with “wdi.properties”

- The “wdi.properties” file specifies:
  - Directories to be used
  - Database Name
  - Language Code
  - Wait Time

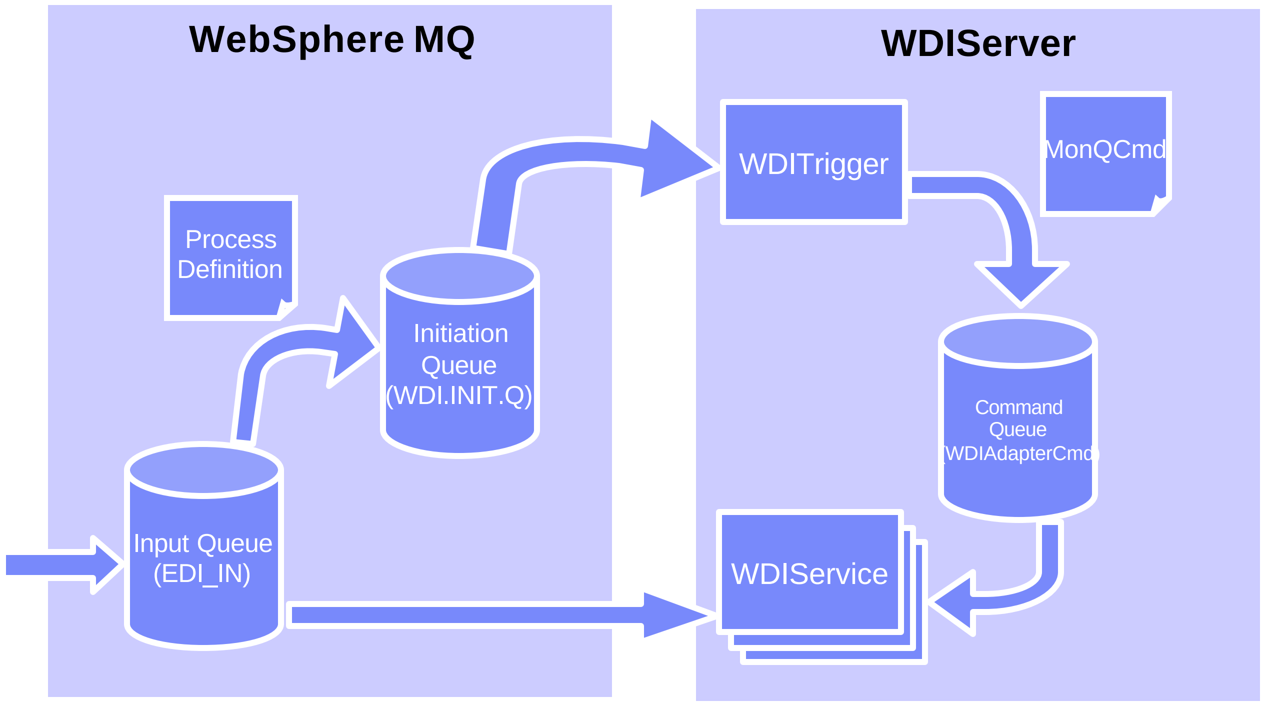
Directories to be used

- runtimedirectory – location of WDI executables
- datadirectory – working directory for WDIAdapter
- rcvdirectory – Receive file directory
- prtdirectory – Print file directory
- dtddirectory – directory for DTD files

Database Name, Language Code, Wait Time

- Database Name “EDIEC32E”
  - User id and password can be specified
- Language code
  - Identifies the Language Profile to be used
- Wait Time
  - Indicates how long WDIAdapter will wait for a message

WDIServer Overview



Setting up MQ Objects

- AdvAdapterMQ.txt
  - Creates WDI.TRANSLATOR.PROC
  - Creates WDIAdapterCmd Queue

WDI.TRANSLATOR.PROC

- New process definition for WDIServer
- Parameters can be passed on “Environment Data”
  - NumThreads – number of translators from pool to assign
  - Timeout – amount of time in microseconds to wait
  - ReqId – value for REQID keyword
  - FileId – value for FILEID keyword

Input Queue - Trigger Data

- Trigger Data can pass parameters
- Environment Data can pass parameters
- Not on both

Large Queue Names

- Long queue names supported with WDIServer
  - Passing ReqId and FileId allows this
  - Default is to use Queue Name for both

Configuring WDIServer

- qmgrname
- initq
- numtranslators
- genprtfile
  - Always/onerror – on error is default
- Keeparchive
  - Always/onerror – never is default

Starting WDIServer

- WDIServer will look for properties when started
  - Current working directory
  - WDISERVER_PROPERITES environment variable
- Initializes translator pool (WDIService processes)
  - Each translator creates a directory
  - WDITransCmdQ_XXXX
- Creates archive directory

Starting WDIServer on AIX

- Extended Shared Memory
  - export EXTSHM=ON

Added Benefits of WDIServer

- Multiple Translators per queue
- The ability to use long queue names

IBM Software Group

Real Time Options for CICS

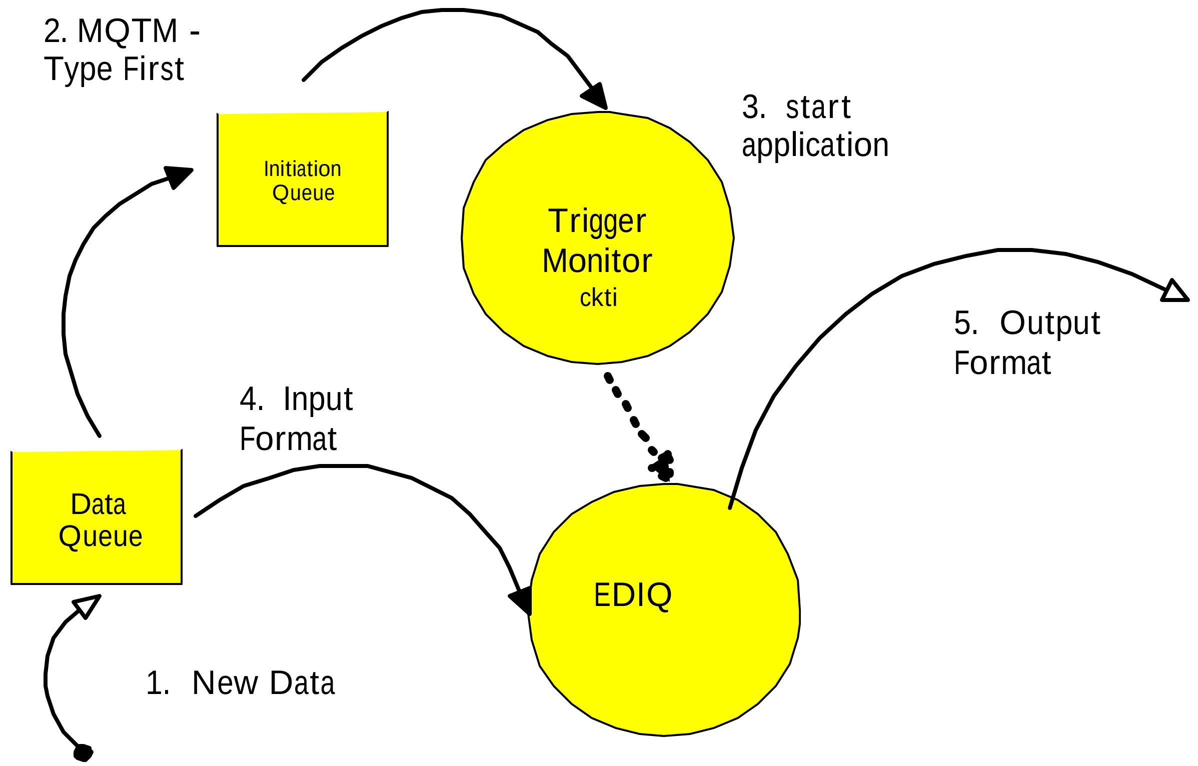
WebSphere. software

ON DEMAND BUSINESS

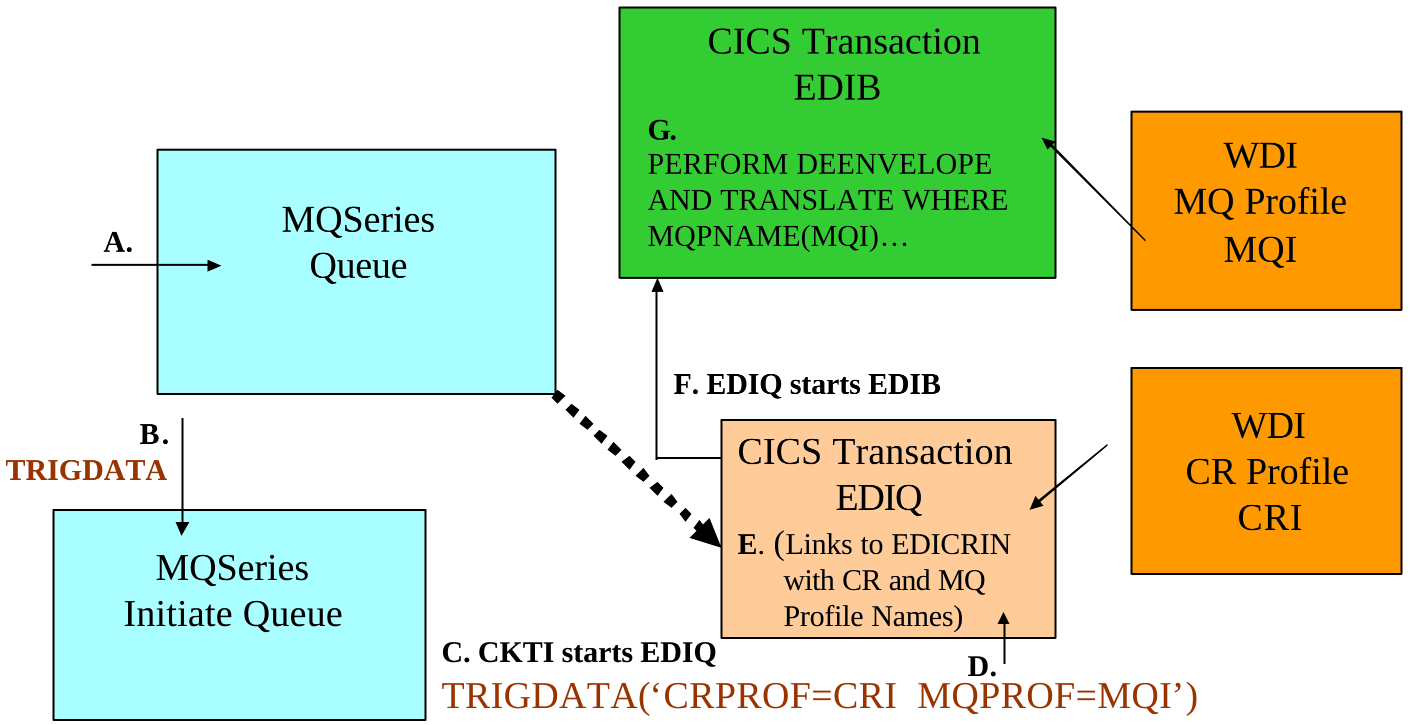
EDIQ

- Provides dynamic behavior in CICS
- Uses triggering
- Based on Continuous Receive

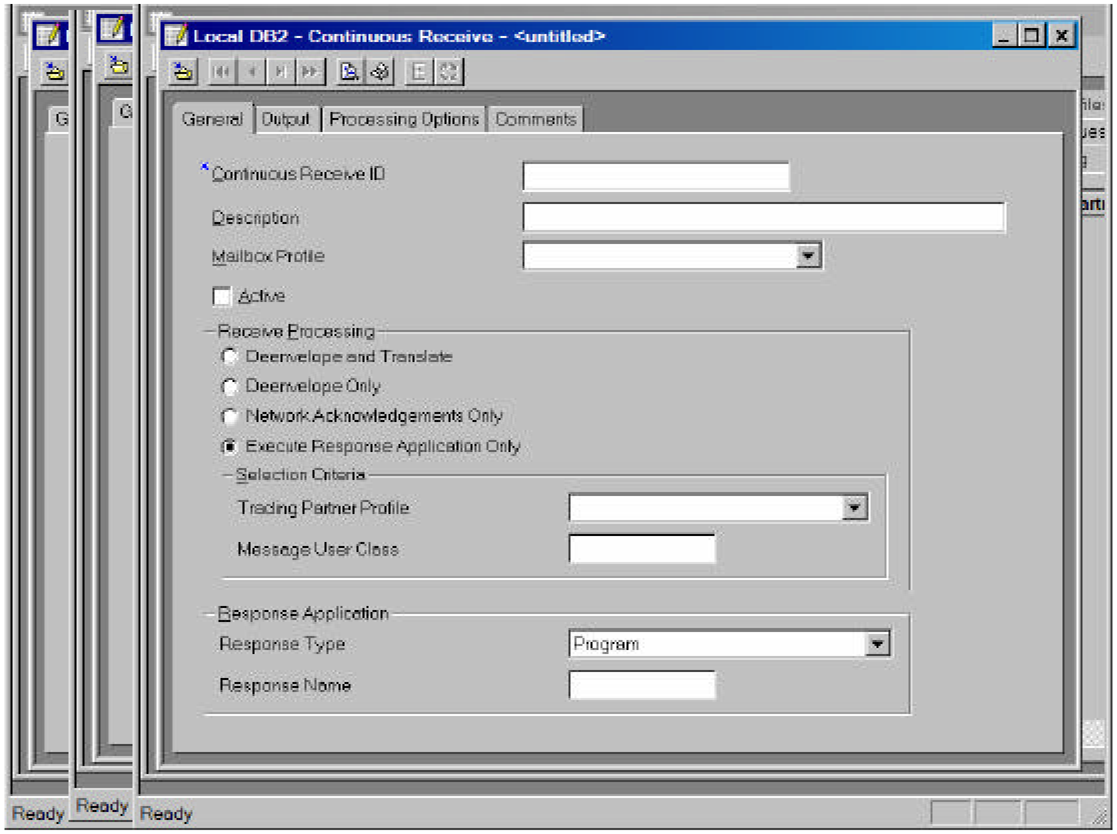
EDIQ


2. MQTM - Type First
Initiation Queue
Trigger Monitor ckti
3. start application
5. Output Format
4. Input Format
Data Queue
EDIQ
1. New Data

Transaction overview



CR Profile


Local DB2 - Continuous Receive - <untitled>

*Continuous Receive ID	
Description	
Mailbox Profile	

☐ Active

Receive Processing
( ) Deenvelope and Translate
( ) Deenvelope Only
( ) Network Acknowledgements Only
(●) Execute Response Application Only

Selection Criteria

Trading Partner Profile	
Message User Class	

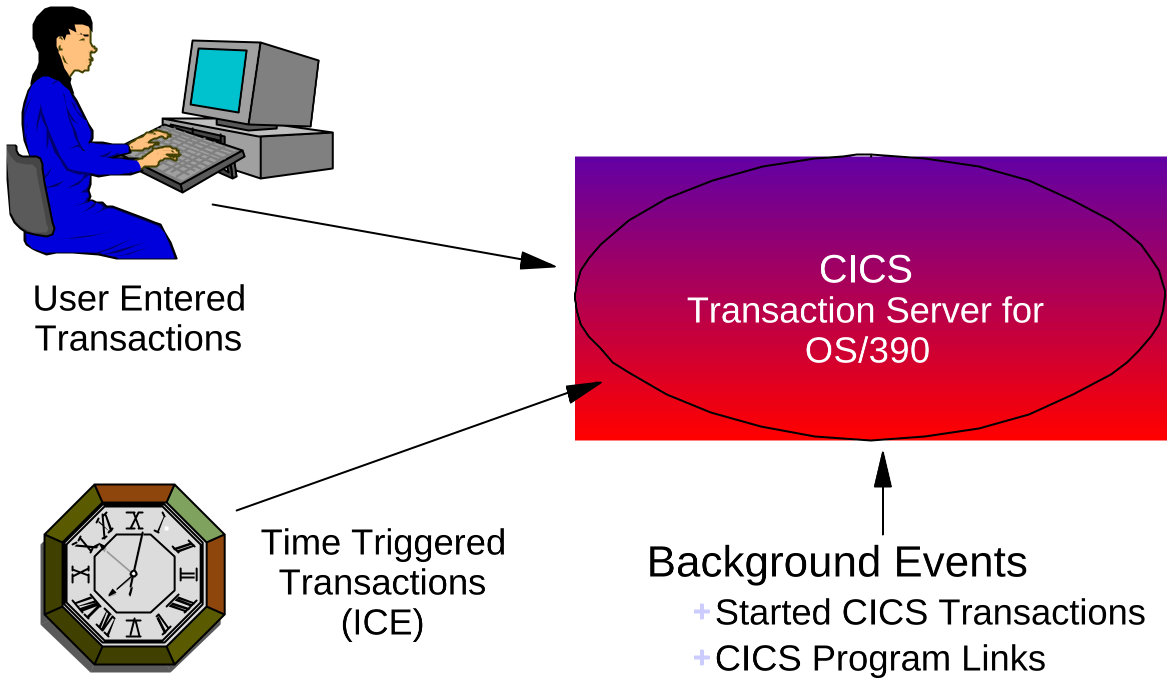
Response Application

Response Type	Program
Response Name	

WDI CICS Specific Functions

- Real time processing
  - WDI Continuous Receive
    - CICS Response programs
    - Update Status
    - Expedite CICS interface to Information Exchange mailbox
  - Hot DI
- CICS preserves and allocates MVS resources in behalf of WDI
- CICS encourages Multiple Concurrent Threads (Processing)
  - WDI has resource contention techniques
    - TSLT DB2 Lock
    - ENQ/DEQ requirements
    - WDI "files" required for concurrency

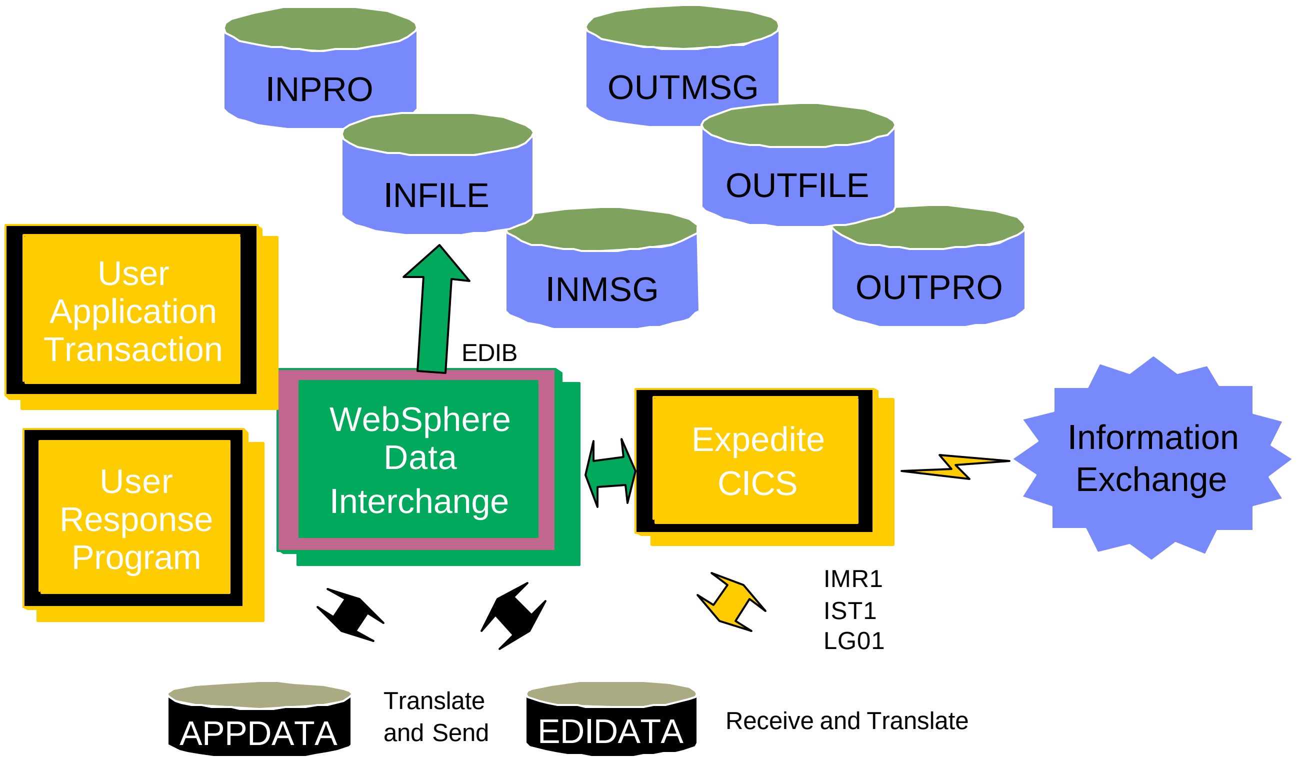
WDI Initiation in CICS


User Entered Transactions
CICS
Transaction Server for OS/390
Time Triggered Transactions (ICE)

Background Events

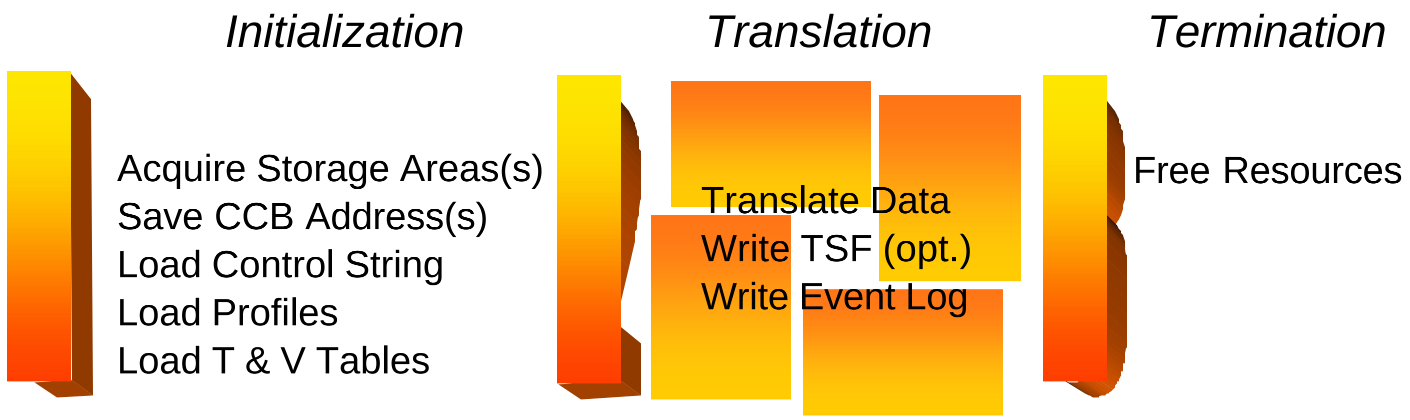
- Started CICS Transactions
- CICS Program Links
- TDQ ATI Triggers

Continuous Receive

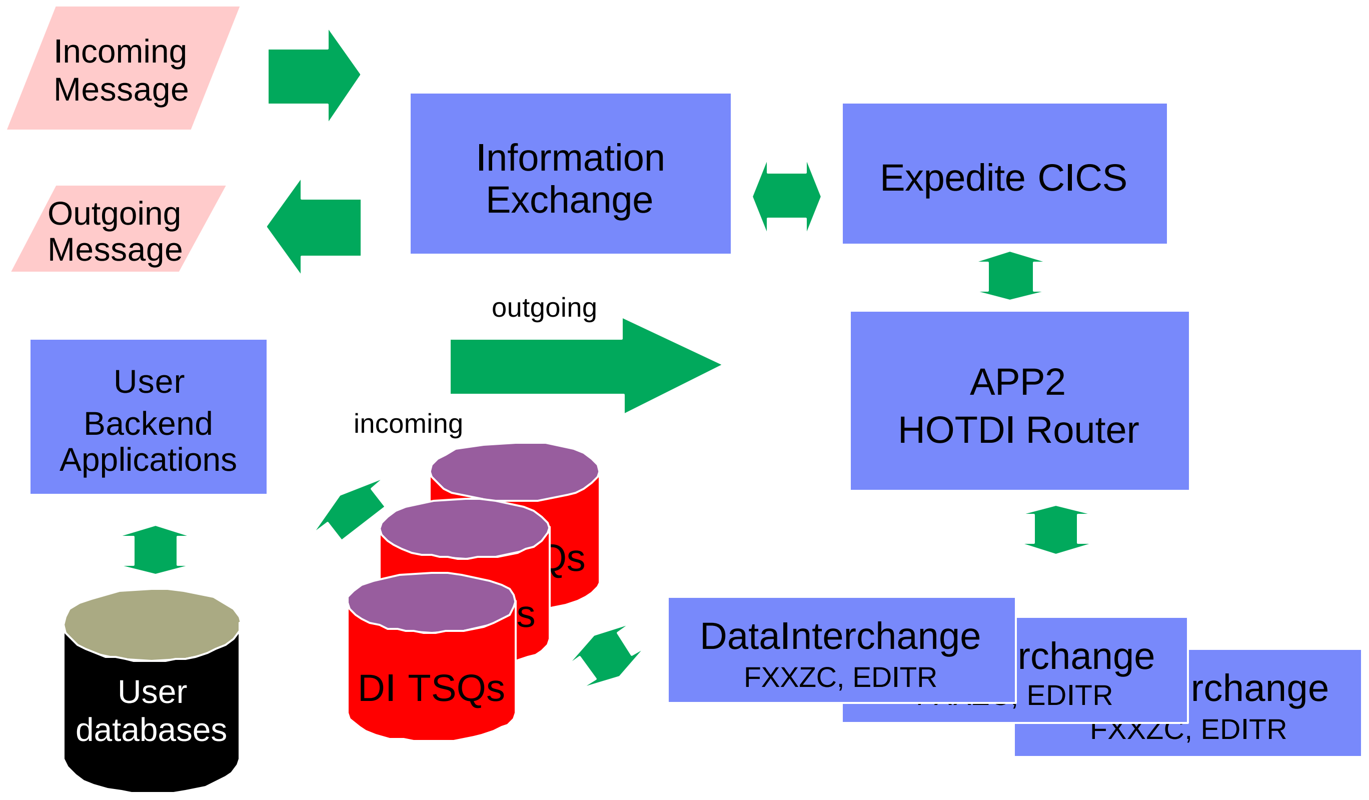


HOT-DI

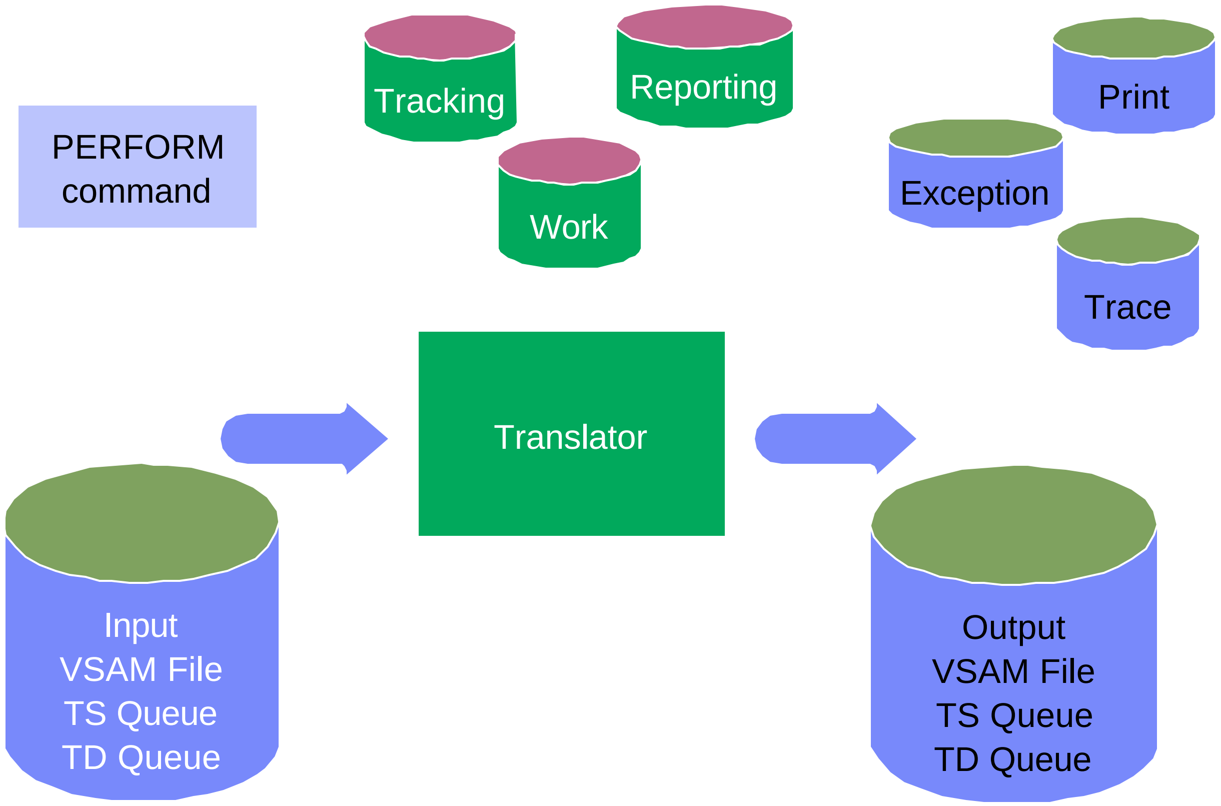
"HOT-DI" Translation Process



Hot DI Example



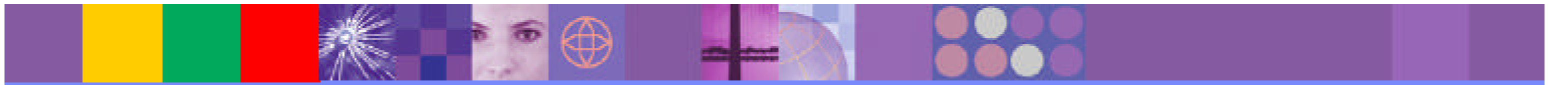
WDI MVS Basic Flow



IBM Software Group

Real Time Options using the Various APIs

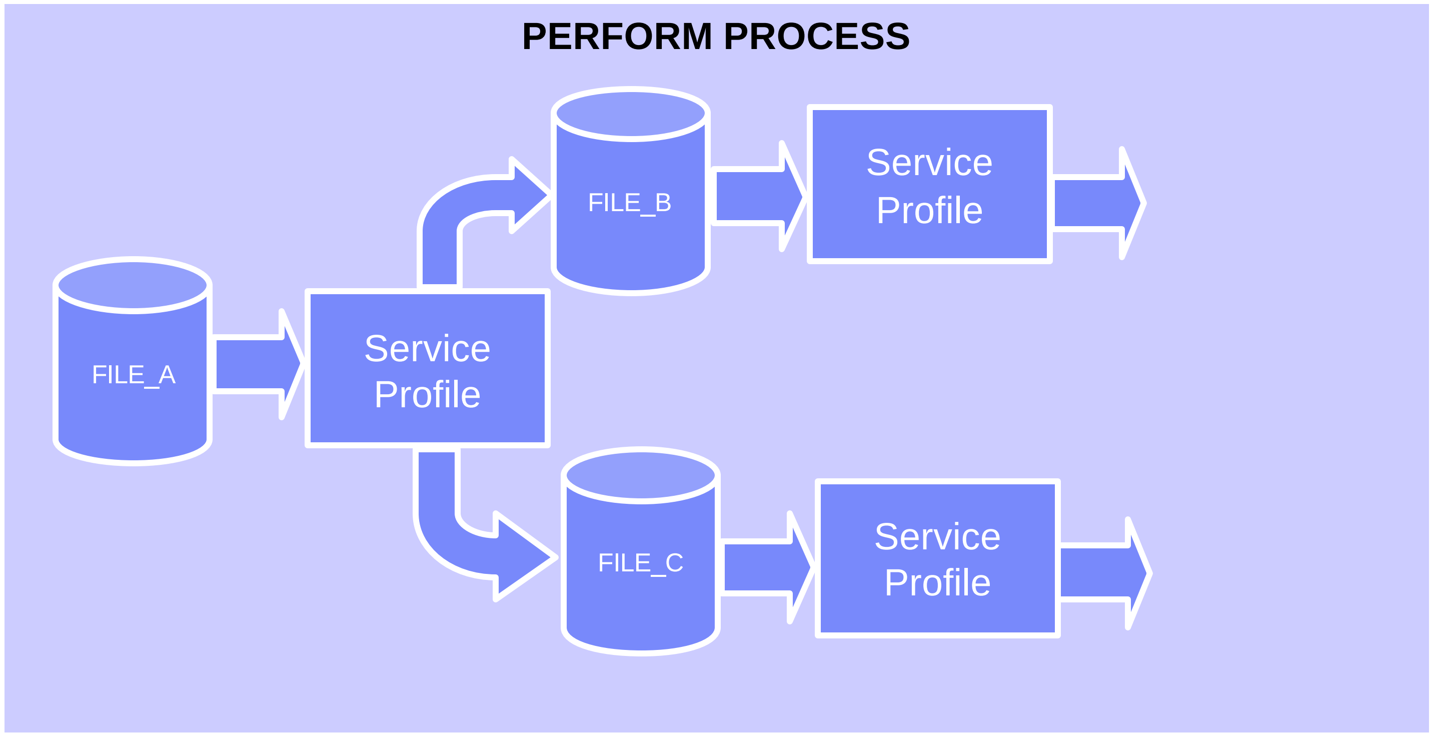
WebSphere. software



What is Command Chaining?

- Introduced with the PROCESS keyword
- Uses PERFORM command templates
- Allows dynamic command substitutions
- Invokes commands when logical files are available
- Used by WDIAdapter and WDIServer

Command Chaining Overview



What functionality does it provide?

- New level of flexibility for solutions.
  - Appropriate action can be taken for each input source.
  - The output of one command becomes input for additional commands.
- Provides the foundation upon which the WDIAdapter and WDIServer programs are built.

What is a Service Profile?

- Service Profiles provide the link between a data source and the specific PERFORM command that will be used to process data found on that data source. The service profile also defines all the files required by the PERFORM command.

Elements of a Service Profile

- Service Profile name
- PERFORM command
- Criteria for continuing the chain
- Input and output file names

What determines which Service Profile is used?

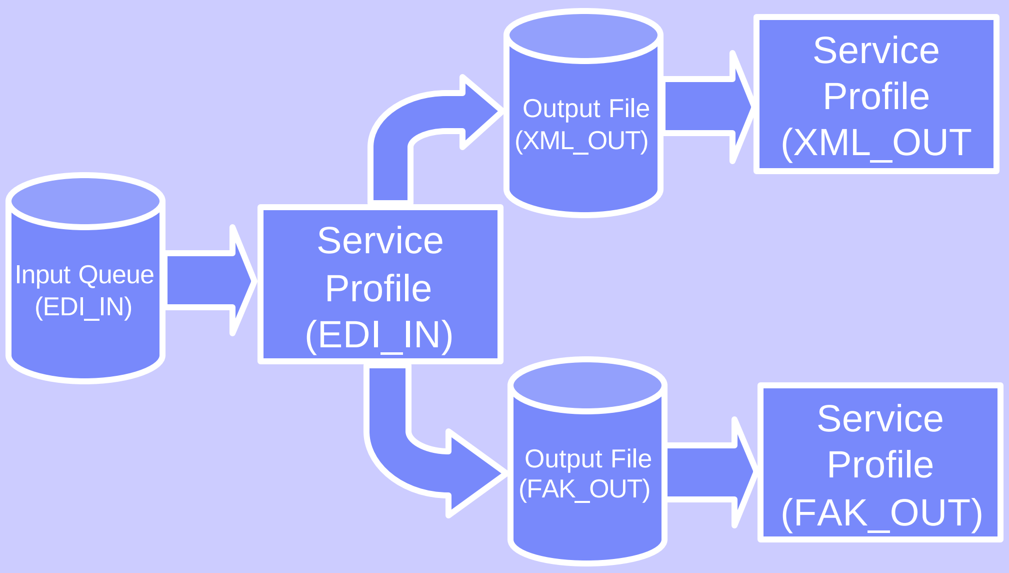
- Process to select a service profile:
  - Logical file has been closed
  - Logical file name matches a Service Profile
  - The current error level matches Service profile criteria

How does command chaining work?

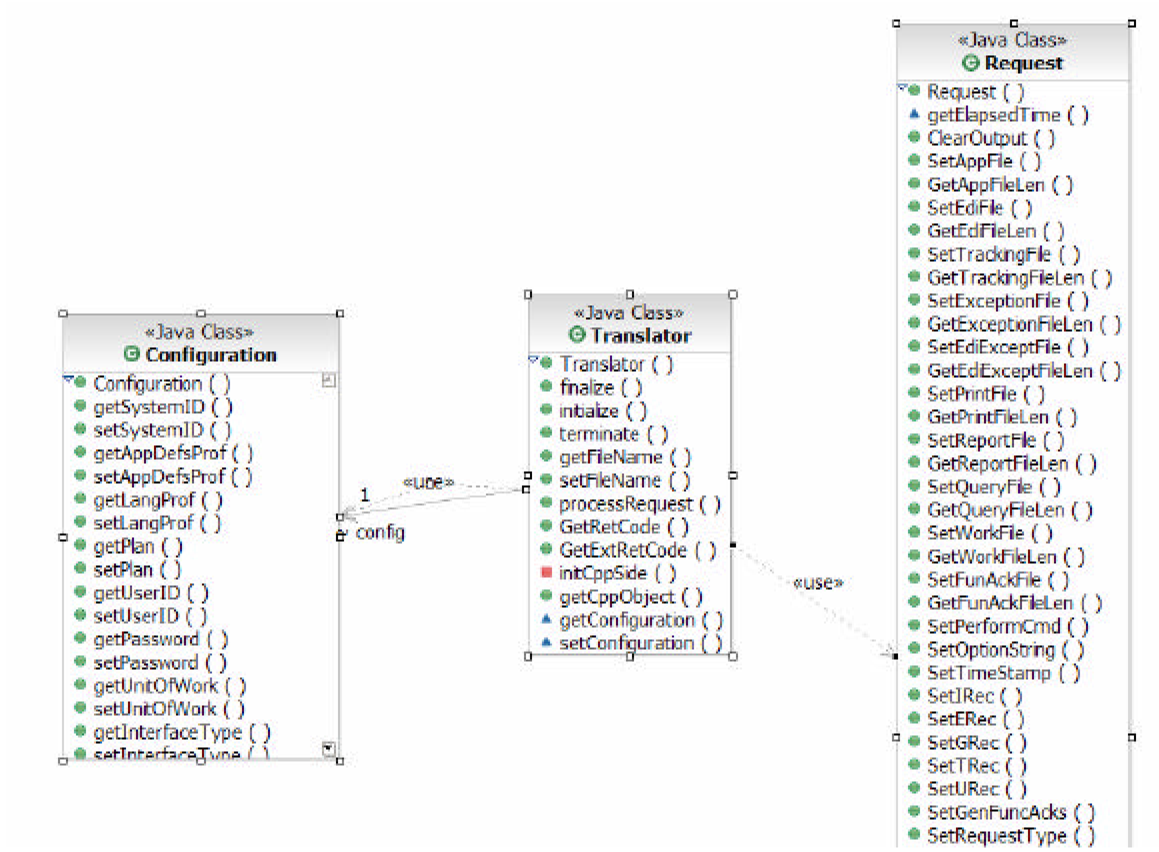
- PERFORM RECEIVE AND PROCESS
  - REQID – Names a Mailbox Profile
  - FILEID – Logical filename
- PERFORM PROCESS
  - FILEID – Logical filename

EDI to XML Example

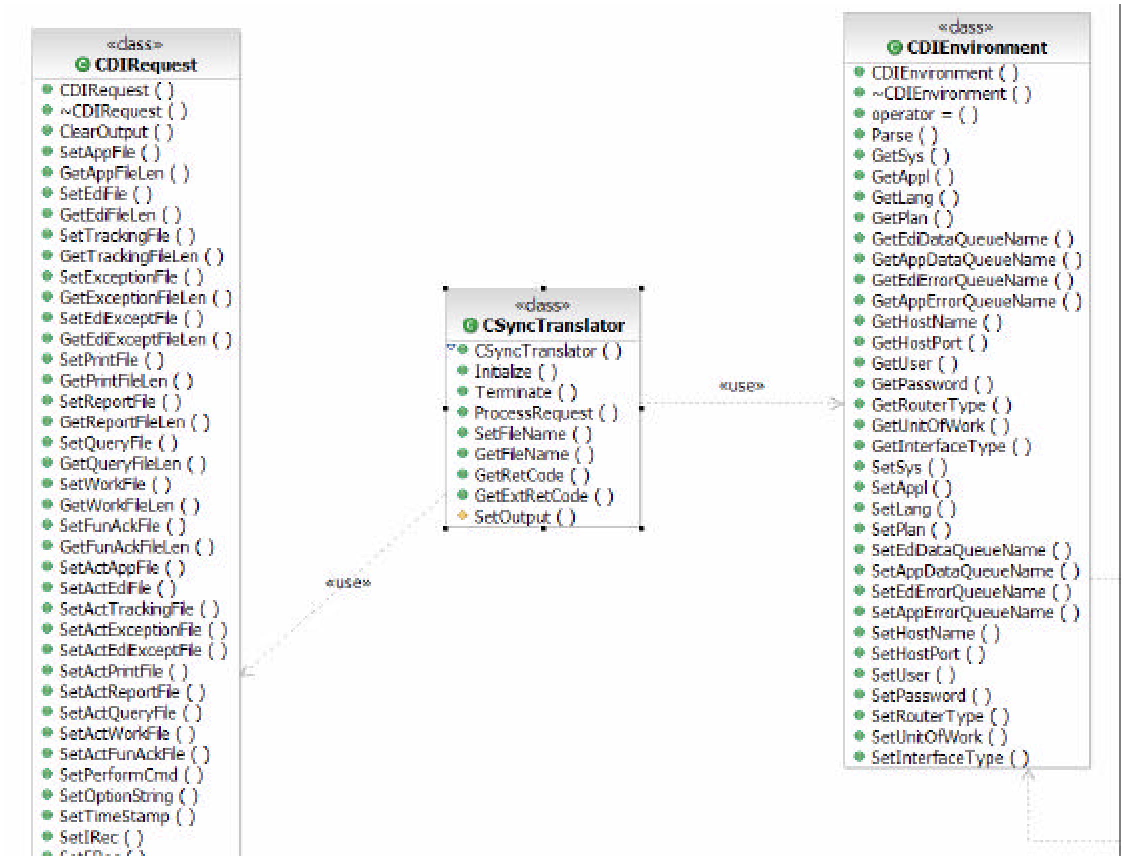
PERFORM RECEIVE AND PROCESS WHERE REQID(EDI_IN) FILEID(EDI_IN)



Real Time Java API Definition



Real Time C++ API Definition



Real Time C++ API Program Example

```
#include <string.h>
#include "diapi.h"
int main ()
{
   CDIEnvironment aCDIEnvironment;
   CDIRequest       aTransformRequest;
   CSyncTranslator aCSyncTranslator;
   enum eResult     rc;
   bool                   bShutdown = 0;

   //Define the Data Interchange Environment
   aCDIEnvironment.SetPlan("EDIEC32E");
   aCDIEnvironment.SetLang("ENU     ");
   aCSyncTranslator.SetFileName("INFILE",   "sample.dat");
   aCSyncTranslator.SetFileName("OUTFILE", "sample.out");
   aCSyncTranslator.SetFileName("FFSEXCP", "sample.aex");
   aCSyncTranslator.SetFileName("PRTFILE", "sample.prt");

   // Initialize the translator:
   rc = aCSyncTranslator.Initialize(aCDIEnvironment);

   while (!bShutdown)    {
      // TODO: Get data to be processed and write to INFILE:

      // Set the perform commands to be executed:
     aTransformRequest.SetPerformCmd
        ("PERFORM PROCESS WHERE FILEID(INFILE);"

      // Ask the synchronous translator to process the EDI to AD F Request:
      rc = aCSyncTranslator.ProcessRequest(aTransformRequest);
   }

   // If we are shutting down, then terminate:
   rc = aCSyncTranslator.Terminate();

   // Terminate and go home:
   return(0);
}
```

Questions and Discussion