



e-business

MQSeries Integrator Agent for CICS Transaction Server

Introduction

Jane Doel-James



IBM

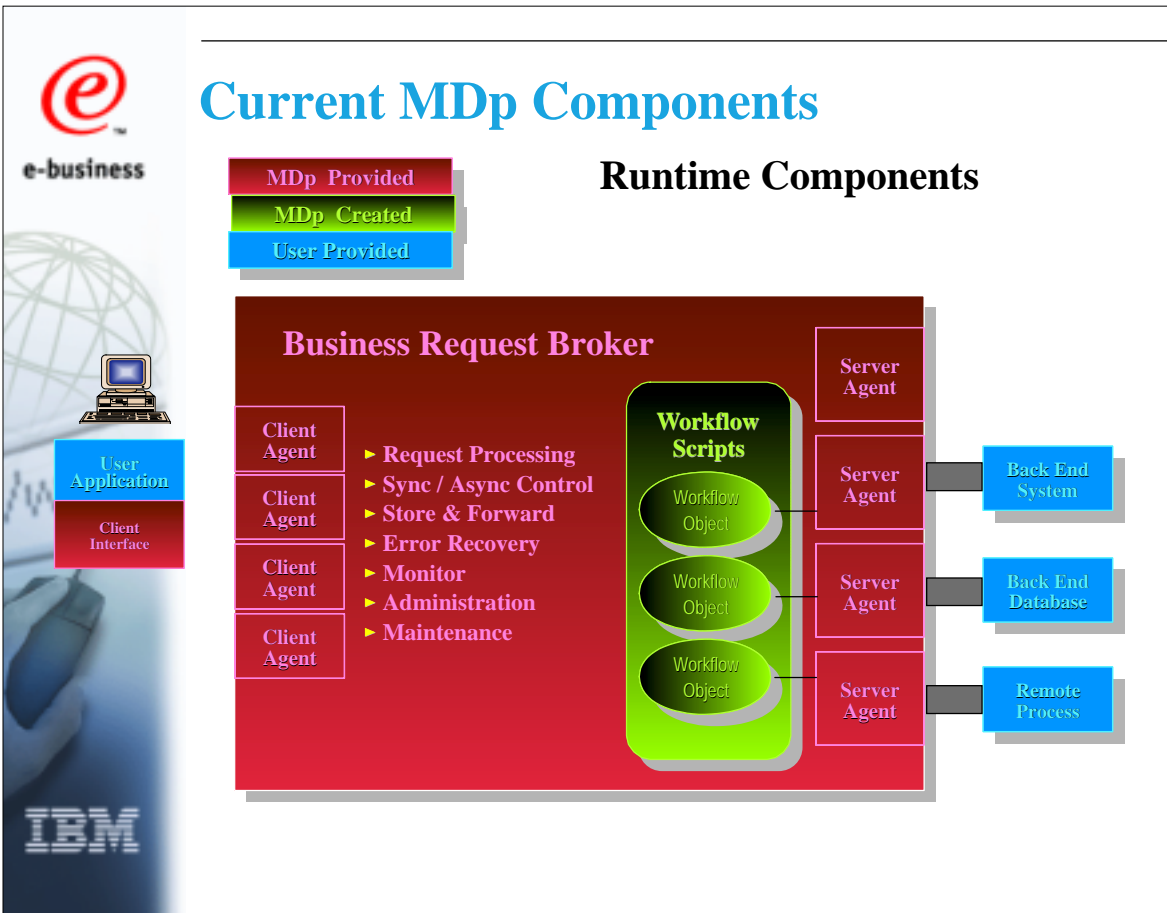


e-business



Agenda

- Why MQIAC?
- MQIAC concepts
- MQIAC adapter builder
- MQIAC runtime



MDp was produced by Early, Cloud. Consists of a runtime environment, and scripts produced by Services people to allow linking of customers' backend CICS and IMS transactions in a workflow. MDp was usually deployed by large IT Org's in CRM Apps that required a Single view of the Customer from multiple request Channels. (i.e Web, IVR, Call Ceter, etc.). Without MDp, this is difficult to implement over "Stove Pipe " delivery systems in CICS and IMS.

MDp is being withdrawn by IBM (who took over Early, Cloud some years ago)

MQSI Agent for CICS provides the first stage in a migration strategy from MDp to more strategic IBM products



e-business



User Application

Client Interface

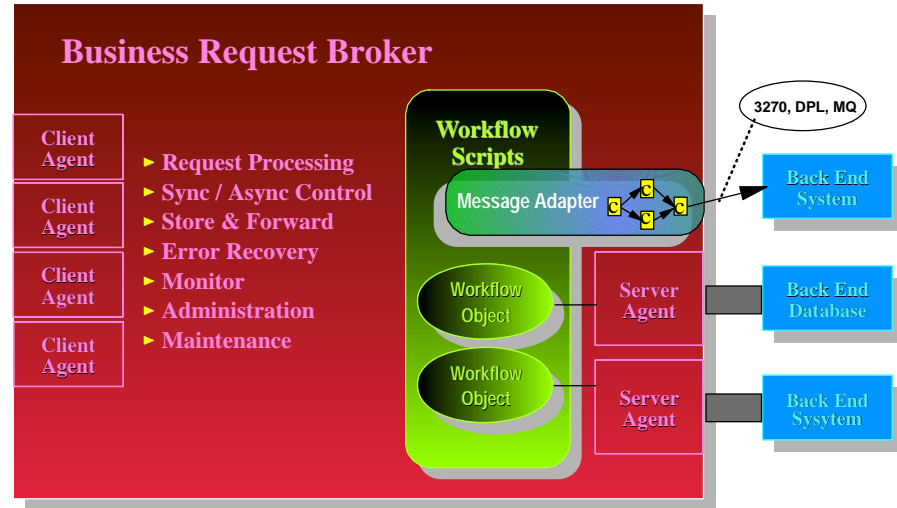


IBM

MDp and MQIAC

- MDp Provided
- MDp Created
- User Provided

Runtime Components

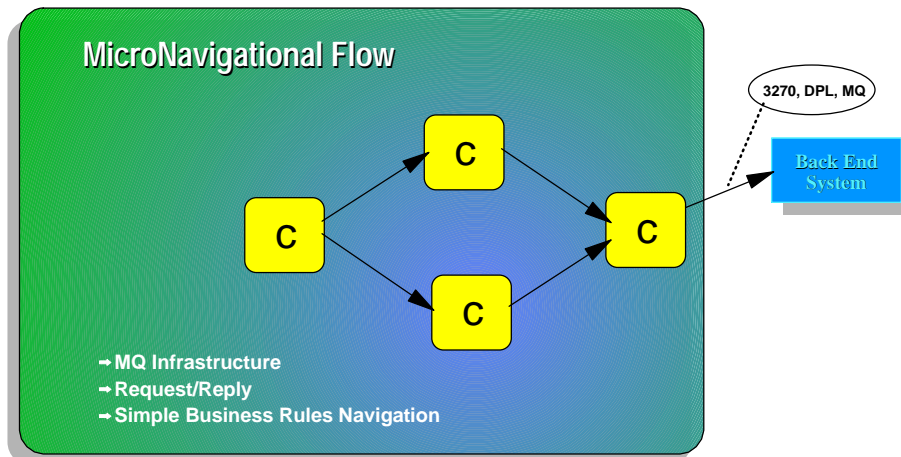


MQIAC adapters can be seen as a replacement for the Workflow scripts element of MDp
They are created by tooling on Windows NT, and run under a MQIAC runtime environment in CTS V1.3
They enable a logical flow to be created using existing user applications.



MQ Integrator Agent for CICS

MQSeries Tools
Generated



MQSI Agent for CICS adapters are generated using tooling based on the MQSI Control Center. This allows the user to specify simple business rules to control the flow between applications. For example, an inquiry application may be run, and depending on the output from this, an update application may then be invoked.

The basic model is a request/reply one - a controlling application starts an adapter with a request message, the adapter invokes the required applications according to the model and returns a reply message to the controlling application. The Adapter or "micro Flow" will behave synchronously in conjunction with the Request / Response boundary.



e-business

What can I use it for?

- Reuse old transactions from new front-end environments
 - ▶ 3270 transactions
 - ▶ CICS programs invoked with a COMMAREA
 - ▶ MQ-enabled transactions
- Run a series of these transactions in order to complete a business function
- Make decisions on next step based on output from transaction

The IBM logo is displayed in a white, blocky, sans-serif font against a dark blue background.

IBM



e-business



What can I use it for?

- Request MailCustomers
 - ▶ Eg Inquire on a Customer - **3270 transaction**
 - ▶ If the output indicates that the customer has a pet dog, mail info on veterinary insurance for dogs - **3270 transaction**
 - ▶ If he has a cat, send information on insurance for cats - **3270 transaction**
 - ▶ When all customers have been processed, output a report - **MQ enabled transaction**
- Response to requester

Requester starts the adapter passing a list of customers
all the following steps are done without further input from the requester
requester receives a response when all processing is done
Any hard, but recoverable, errors may also be returned as an error reply i.e 3270 link down



MQIAC Product Abstract

Product description:

IBM MQSeries Integrator Agent for CICS Transaction Server, known here also as MQSI Agent for CICS (MQIAC), consists of the MQSI Agent for CICS Adapter Builder and the MQSI Agent for CICS server run time. These components work together in a process that defines, models and executes "microflows" (business transactions) in a CICS / TS 1.3 environment.

Product Approach:

Using the MQSI Agent for CICS builder, a user can construct a microflow, in order to model all or part of the behavior necessary for executing a business transaction. Once a microflow is constructed, the user is then able to generate COBOL source code. This source code contains the microflow and static server run time information, that together make up the adapter. The adapter, which is specific to the business transaction, is deployed to an OS/390 environment, where it is hosted and executed as a CICS business transaction (adapter component).

'Business transaction' is the BTS concept of a transaction - this differs from the classic CICS transaction
CICS transaction

- normally short-lived - associated with a CICS task

- traditionally initiated from a 3270 terminal

- usually associated with a unit-of work

- can be linked together pseudo-conversationally to complete a piece of processing

BTS - a business transaction is the set of steps required to complete a piece of business - eg. book a holiday

- may be long-lived

- may be initiated by any CICS input

- may be made up of a number of CICS transactions

However in the current MQIAC deployment, we only drive an instance of the adapter, and hold any "micro flow" state in BTS containers only for Controlling app to compensate by potentially using connecting another Adapter or "micro flow" instance to that state for compensation.



e-business

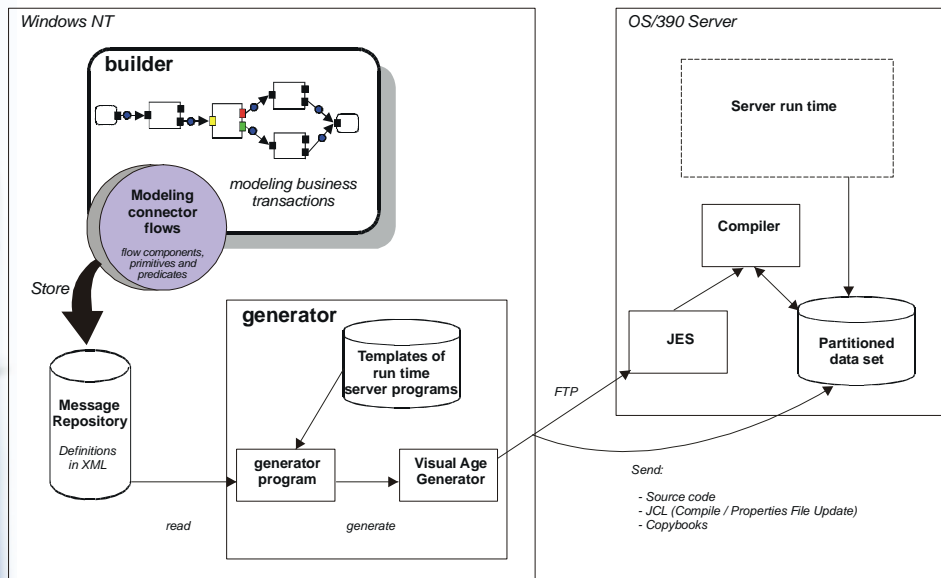


Development First Principles

- **Design in accordance with IBM MQ tooling architecture**
 - ▶ Logical message model (MRM)
 - ▶ Flow composition model / Flow composition builder
 - ▶ Adapter / connector architecture
- **Utilize / reuse MQSI v2 builder as tool foundation to extend**
 - ▶ Assume BIT packaging at some point
- **Utilize / reuse existing IBM tool products where appropriate**
 - ▶ Host on Demand components
 - ▶ Visual Age Gen components
 - ▶ EAB / CCF components
- **Follow roadmap / direction of eclipse tool framework**
 - ▶ Intercept Eclipse development rollout plan when feasible
- **Follow roadmap / direction of runtime evolution**
 - ▶ MQWF (staffless workflow)
 - ▶ WAS , J2EE / J2C, Message beans,
 - ▶ "Micro scripting" RT services progression



MQIAC Build time overview



Use the builder component to model a connector flow (i.e., microflow) that represents the required adapter behavior.

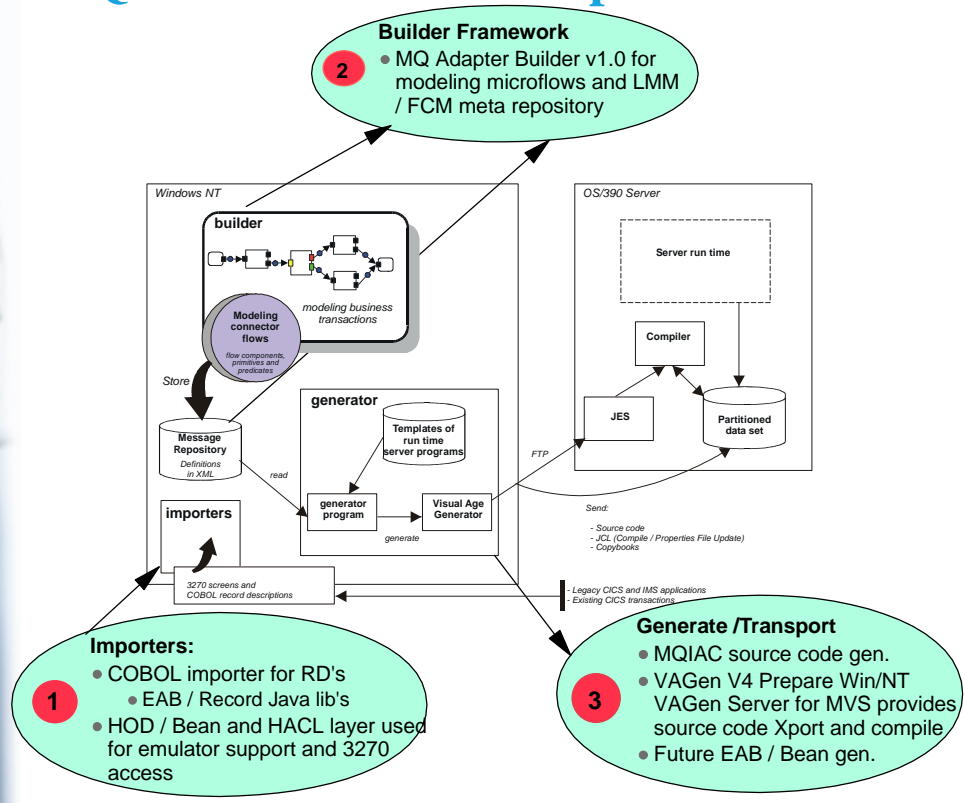
Use the generator facility to generate COBOL source code and JCL for the connector flow.

Move the COBOL source code to the OS/390 server for compilation. The compilation results in the adapter, which resides on the OS/390 as a CICS/BTS application.

Data structures and screen layouts from existing applications are imported into the MQSI Adapter Builder
The Adapter Builder tooling is used to model flows between the existing applications
Models are held as XML documents in a repository
The Adapter Builder tooling is used to generate CICS Cobol BTS programs to implement the models, and to
deploy the programs to an TS 1.3 OS/390 server and compile them
A controlling application invokes the Adapter, passing input in an architected message
The Adapter runtime executes, invoking user transactions as specified in the model

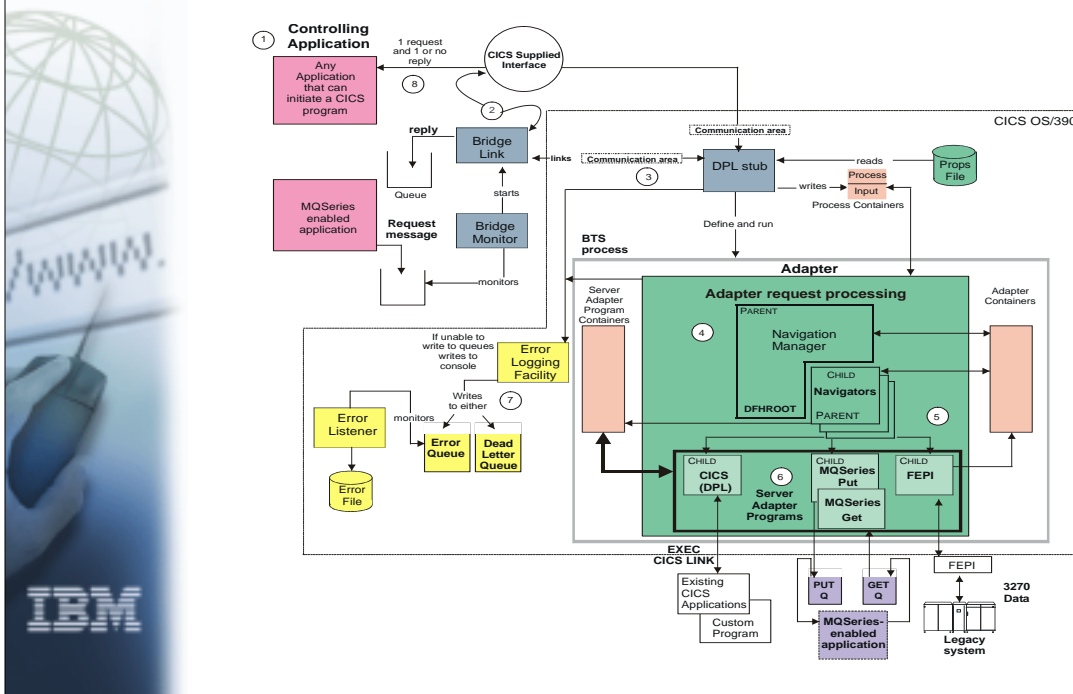


MQIAC Build time composition





Adapter run time processing



Controlling application can

- initiate the adapter runtime using a CICS LINK interface (DPL, EXCI, ECI)

- put a message on an MQ queue, using the MQ/CICS Bridge to initiate the adapter runtime - preferred - this method must be used for update transactions (i.e. if recoverability is required)

In either case the DPL stub is invoked and a COMMAREA is passed

DPL stub initiates the BTS process, running the Navigation Manager as DFHROOT

The Navigation manager passes control to the Navigator (this is the highest level of our modelled adapter and controls the adapter logic)

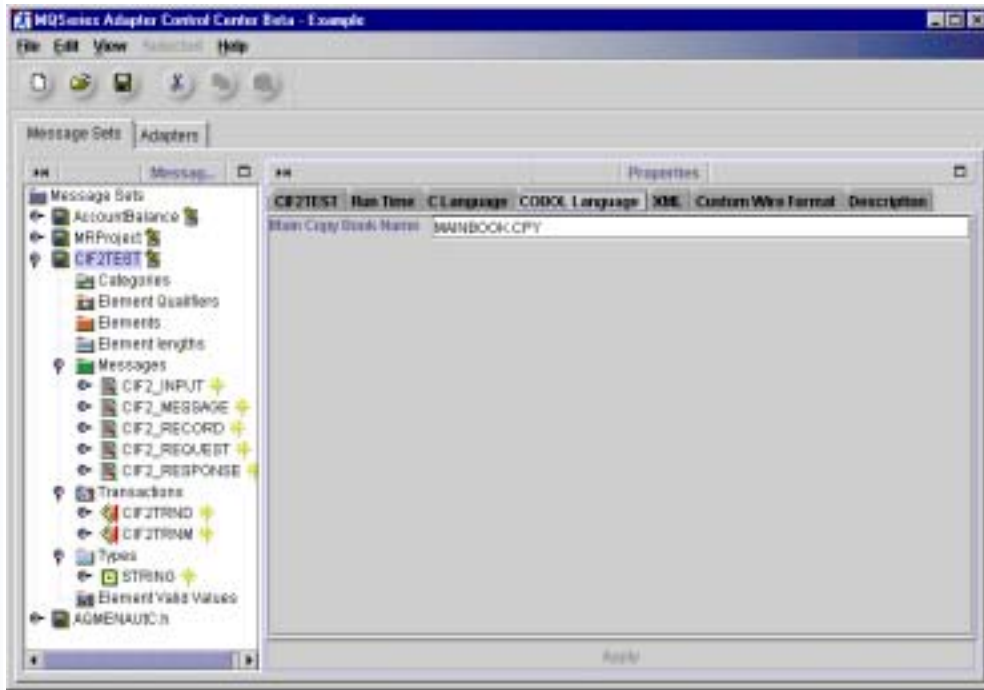
The Navigator executes the adapter control logic, defining and running activities to implement the specific calls to user transactions (adapter servers)

The adapter servers call the backend transactions using the appropriate interface (DPL, MQPUT/MQGET or FEPI screen navigation)

When all user transactions have been run as required, control is passed back through the Navigator and the Navigator Manager to the DPL Stub, which returns the reply to the application either directly in a COMMAREA or via the MQ/CICS Bridge



Importing COBOL Structured Data Types

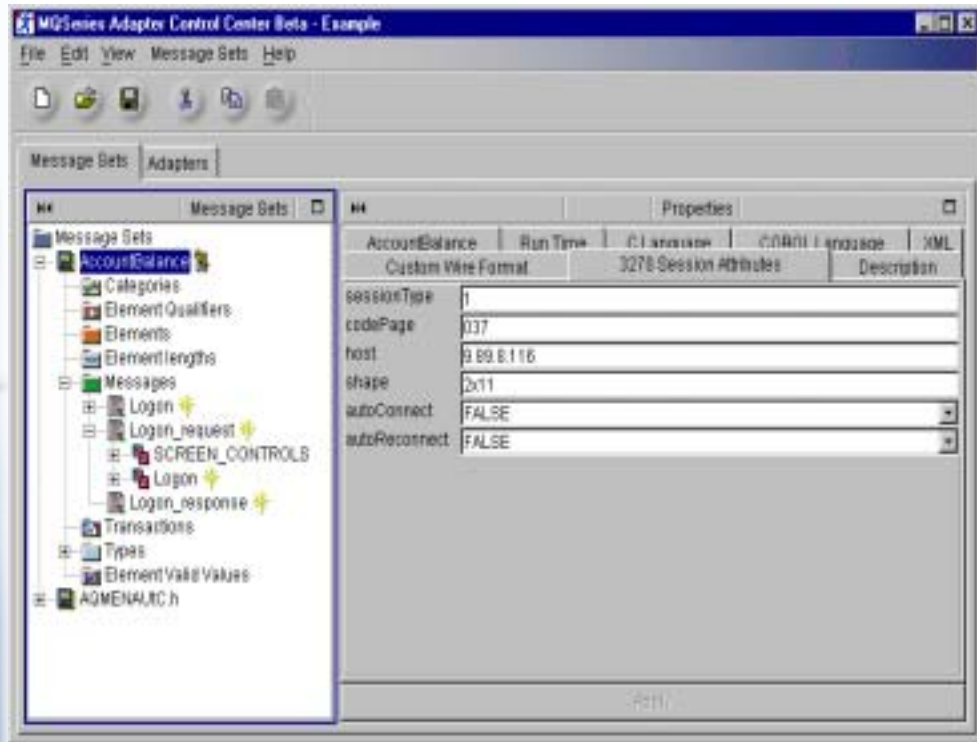


Control Center display divided into two areas - Message Sets and Adapter
Message Sets contain the Imported Cobol copybooks and 3270 screens. It is also possible to define messages directly within the Control Center



e-business

Importing 3270 Screens

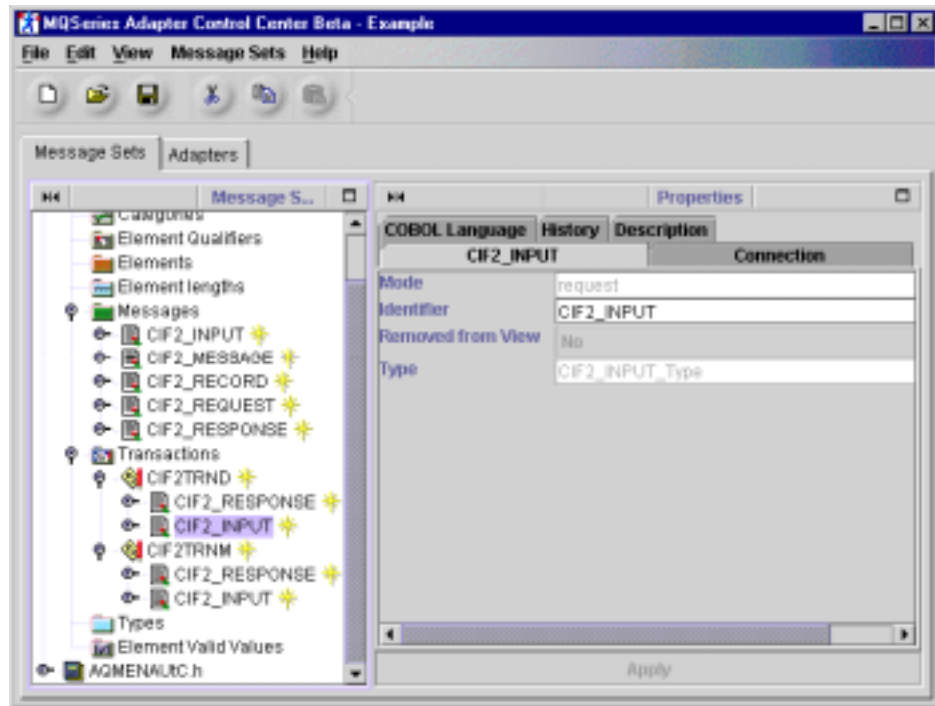


When 3270 screens are imported, their definition includes the IP address of the host they were imported from

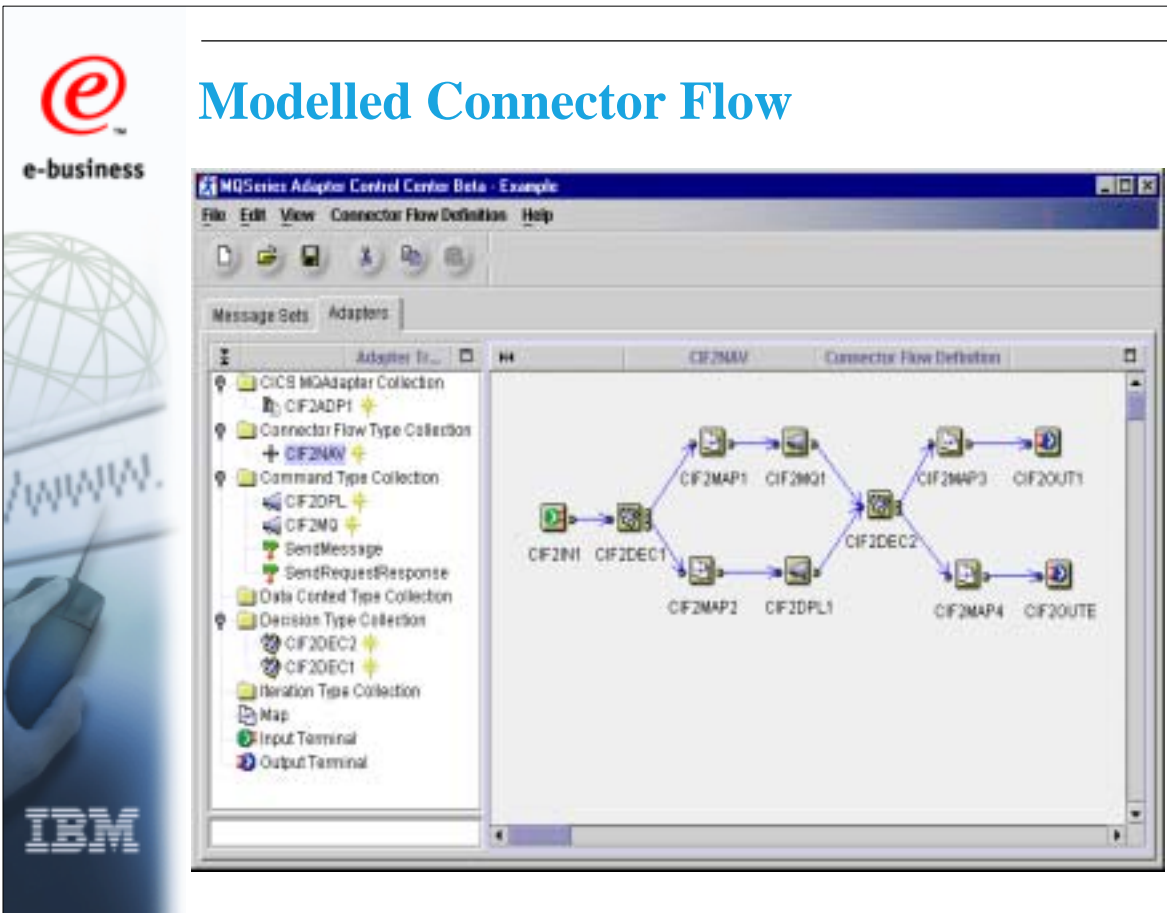
Screen Control fields are also generated on the 3270 import, and a Transaction (special parser) used in the Command to identify and wire the control flow for the screen.



Message model and transactions



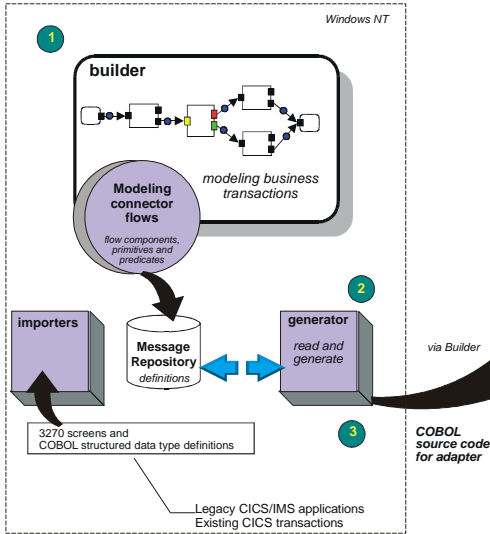
When messages have been imported, they can be associated with transactions to form the basic building blocks of the modelled flow



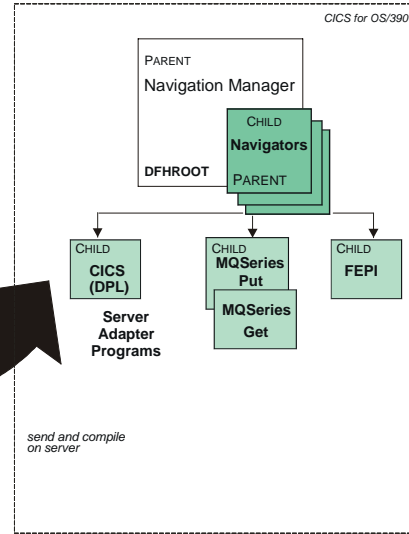
- A modelled flow (known as a Connector Flow Definition) can contain different nodes:
 - Input and Output terminals define the beginning and end points of the flow and the data associated with them
 - Map nodes allow the user to control the way data passes between nodes in the flow
 - Command nodes represent the logic to actually call the user transactions - these use the transaction definitions created earlier to define their inputs and outputs
 - Decision nodes control routing through the flow depending on simple business rules
 - Iteration nodes allow repeated passes through logic

Adapter: Model and Deploy

Modeling, generating and deploying



Deployed adapter



1. Use builder to model a microflow representing the required adapter behavior.
2. Generate COBOL source code for microflow.

3. Transport the COBOL source code to an OS/390 server for compilation.



Runtime parameters

- Parameters needed at run time held in Property file
- Populated from
 - ▶ Specification files created on NT - used as input to update job at deploy time
 - ▶ Update job submitted from TSO
- Contains program ids, transids, FEPI pool names, MQ queue names...
- Allows flexibility of runtime deployment

The IBM logo, consisting of the letters 'IBM' in a bold, white, sans-serif font, positioned at the bottom of a vertical blue gradient bar on the left side of the slide.

IBM

Link this to model and deploy screen before and runtime screens that follow



Specification file

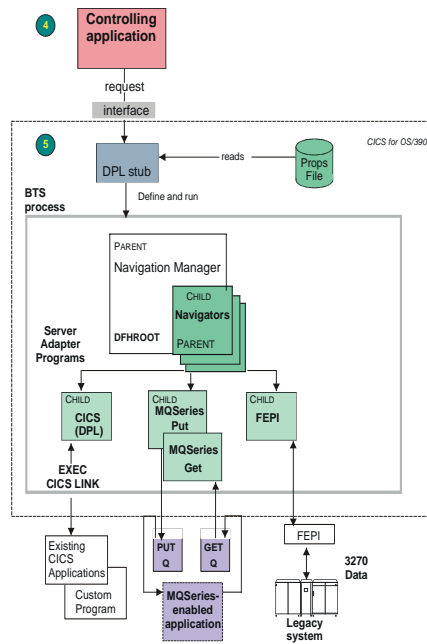
Specification file coded in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE AttributeGroup SYSTEM "mqsi.dtd" >
<AttributeGroup xmi.label="Connector Resource">
  <Attribute xmi.label="MAT_REQTYPE" type="" xmi.uuid="" valueMandatory="false" value="0" encoded="false"/>
  <Attribute xmi.label="MAT_NAVTYPE" type="" xmi.uuid="" valueMandatory="false" value="F" encoded="false"/>
  <Attribute xmi.label="MAT_LOGOFFTYPE" type="" xmi.uuid="" valueMandatory="false" value="R" encoded="false"/>
  <Attribute xmi.label="MAT_POOL" type="" xmi.uuid="" valueMandatory="false" value="CICSDEV2" encoded="false"/>
  <Attribute xmi.label="MAT_TARGET" type="" xmi.uuid="" valueMandatory="false" value="DFLTTARG" encoded="false"/>
  <Attribute xmi.label="MAT_CONVERSE" type="" xmi.uuid="" valueMandatory="false" value="Y" encoded="false"/>
  <Attribute xmi.label="MAT_TIMEOUT" type="" xmi.uuid="" valueMandatory="false" value="025" encoded="false"/>
  <Attribute xmi.label="MAT_MAXCALEN" type="" xmi.uuid="" valueMandatory="false" value="400" encoded="false"/>
</AttributeGroup>
```

The IBM logo, consisting of the letters 'IBM' in a bold, blue, sans-serif font, positioned at the bottom left of the slide.



Adapter: Execution



4. Controlling application initiates model execution (adapter request processing) at run time.

5. The DPL MQSI Agent for CICS Stub program uses information in the request message to:

- a. Read the Properties file
- b. Define the BTS process
- c. Write containers and run the BTS process
- d. Initiate the programmatic functions (adapter request processing) that enable the business transaction to be processed.

Link numbered points to previous Model and Deploy screen

Communication between levels of processing uses BTS containers

The DPL Stub program creates Process containers that can be read by activities at all levels

The Navigator uses ACTIVITY containers to pass input data to its child activities and receive responses.

Error and status information is also held in containers



Request message: Structure

← maximum message length with all headers: 25,472 bytes →

← message data →

MQSeries Header MQMD	MQSeries-CICS Bridge Header (MQCIH)	DPL Stub program ("DFHMADPL")	MQSI Agent for CICS Message Header (DFHMAHV)	application data
--------------------------------	---	---	--	------------------

324 bytes

180 bytes

8 bytes

384 bytes

24,576 bytes


MQMD-FORMAT = MQFMT-CICS

MQCIH-FORMAT = MQFMT-MSG-ADAPTER

DFHMAHV-FORMAT = Request/reply format


Request/reply format describes the structure of the application data and is used by the controlling application

Legend:

 = The only portion required if the controlling application is using a CICS-supplied interface to pass the request message to the DPL Stub program



Standard MQCIH header structure with DFHMADPL and DFHMAHV ancillary header data to drive the Adapter and "micro flow" .



Processing modes

- Asynchronous mode
 - ▶ The BTS process is run asynchronously from the initiating task
 - used for update requests
 - requests must use MQ/CICS bridge
- Synchronous mode
 - ▶ The BTS process is run synchronously with the requesting task
 - used for inquiry requests
- Synchronous rollback requests
 - ▶ As synchronous, but detection of an error results in task being abended and resources rolled back
 - use for update (DPL requests only)

Specified in runtime properties file

in syncpoint rollback mode detection of an error will cause all preceding steps to be rolled back. This could be useful if the request processing required a series of DPL requests to be run as a single unit of work. in asynchronous mode, syncpoints are taken as the Navigation Manager, Navigator and server adapter programs complete their processing. These sync points provide the necessary state, status and journalling information in the event of subsequent failure

To the outside world, the adapter is actually a "synchronous" micro flow though - re-affirm (request / response model).



e-business

Compensation

- Compensation flows modelled as separate adapters
- Failure of a request is notified to controlling application, which must request compensation flow
- When a request fails, state and status information is retained in containers (so process is not completed), and can be used by compensation flow
- When a compensation flow is requested, DPL stub copies compensation data from old process to containers belonging to the new process, cancels the old process, and runs new process for compensation flow

IBM

Request must have been run in asynchronous mode for compensation to be possible

Fields in the MAH specify that a request is a compensation request

State and status are held in failing process's process containers, journaling data in failing DFHROOT's activity container



Further information

- MQIAC web site
 - ▶ <http://www-3.ibm.com/software/ts/cics/mqiac/>
- MQIAC Runtime User's Guide (SC34-5899)
- MQIAC Using the Control Center (SC34-5901)

