SHARE in Orlando 2015

Nobody Uses Files Any More, Do They? New Technologies for Old Technology, File Processing in MQ MFT and IIB

Session 17891

13th August 2015

Chris Leonard

Team Lead, MQ Distributed Level 3 Service

Geza Geleji

Staff Software Engineer, IBM Integration Bus Development

#SHAREorg

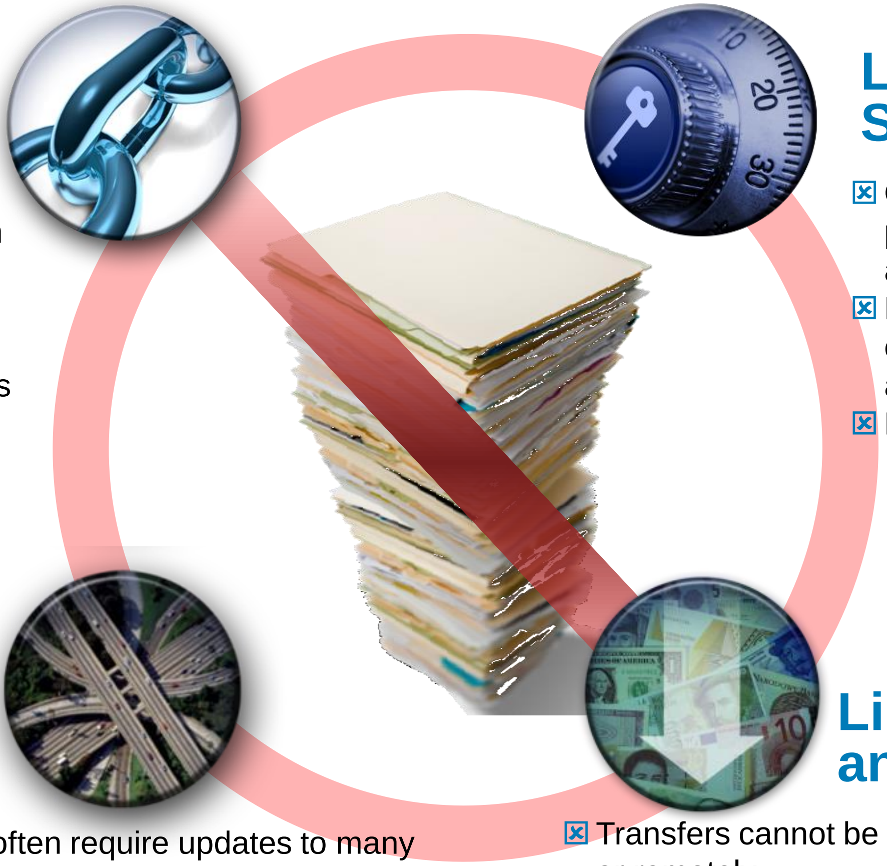
SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence.**

How do most organizations move files today?

Most organizations rely on a mix of homegrown code, several legacy products and different technologies … and even people!

- FTP
  - Typically File Transfer Protocol (FTP) is combined with writing and maintaining homegrown code to address its limitations
- Why is FTP use so widespread?
  - FTP is widely available – Lowest common denominator
  - Promises a quick fix – repent at leisure
  - Simple concepts – low technical skills needed to get started
  - FTP products seem “free”, simple, intuitive and ubiquitous
- Legacy File Transfer products
  - A combination of products often used to provide silo solutions
  - Often based on proprietary versions of FTP protocol
  - Can’t transport other forms of data besides files
  - Usually well integrated with B2B but rarely able to work with the rest of the IT infrastructure – especially with SOA
- People
  - From IT Staff to Business staff and even Security Personnel
  - Using a combination of email, fax, phone, mail, memory keys…

Shortcomings of Basic FTP



Limited Reliability

- ☒ Unreliable delivery – Lacking checkpoint restart – Files can be lost
- ☒ Transfers can terminate without notification or any record – corrupt or partial files can be accidentally used
- ☒ File data can be unusable after transfer – lack of Character Set conversion

Limited Security

- ☒ Often usernames and passwords are sent with file – as plain text!
- ☒ Privacy, authentication and encryption often not be available
- ☒ Non-repudiation often lacking

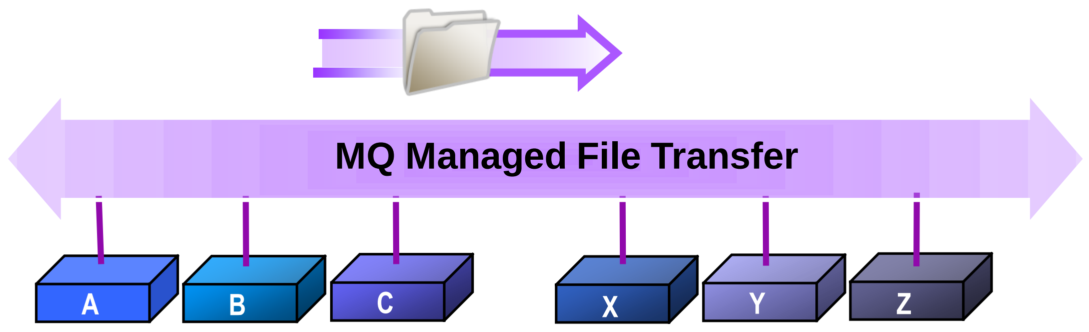
Limited Flexibility

- ☒ Changes to file transfers often require updates to many ftp scripts that are typically scattered across machines and require platform-specific skills to alter
- ☒ All resources usually have to be available concurrently
- ☒ Often only one ftp transfer can run at a time
- ☒ Typically transfers cannot be prioritized

Limited visibility and traceability

- ☒ Transfers cannot be monitored and managed centrally or remotely
- ☒ Logging capabilities may be limited and may only record transfers between directly connected systems
- ☒ Cannot track the entire journey of files – not just from one machine to the next but from the start of its journey to its final destination

What is MQ Managed File Transfer?



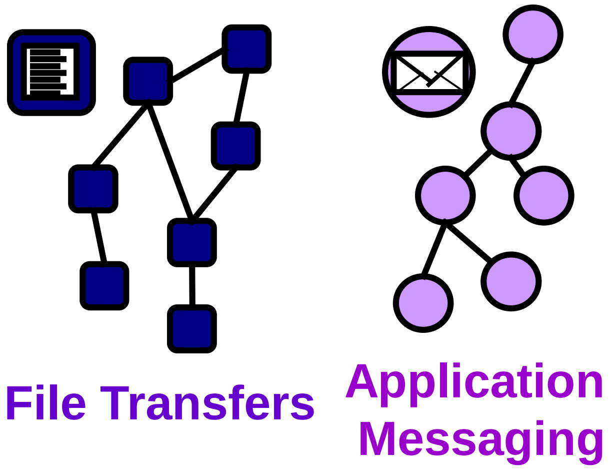
☑ Auditable	Full logging and auditing of file transfers + archive audit data to a database
☑ Reliable	Checkpoint restart. Exploits solid reliability of MQ
☑ Secure	Protects file data in transit using SSL. Provides end-to-end encryption using AMS
☑ Automated	Providing scheduling and file watching capabilities for event-driven transfers
☑ Centralized	Provides centralized monitoring and deployment of file transfer activities
☑ Any file size	Efficiently handles anything from bytes to terabytes
☑ Integrated	Integrates with MB, WSRR, ITCAMs for Apps, DataPower + Connect:Direct
☑ Cost Effective	Reuses investment in MQ. Wide range of support (inc. z/OS and IBM i)

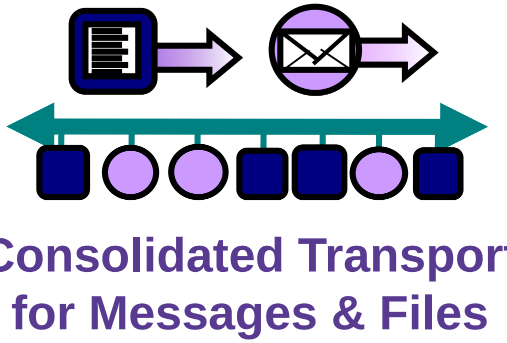
A consolidated transport for both files and messages

- Traditional approaches to file transfer result in parallel infrastructures
  - One for files – typically built on FTP
  - One for application messaging – based on MQ, or similar

- High degree of duplication in creating and maintaining the two infrastructures

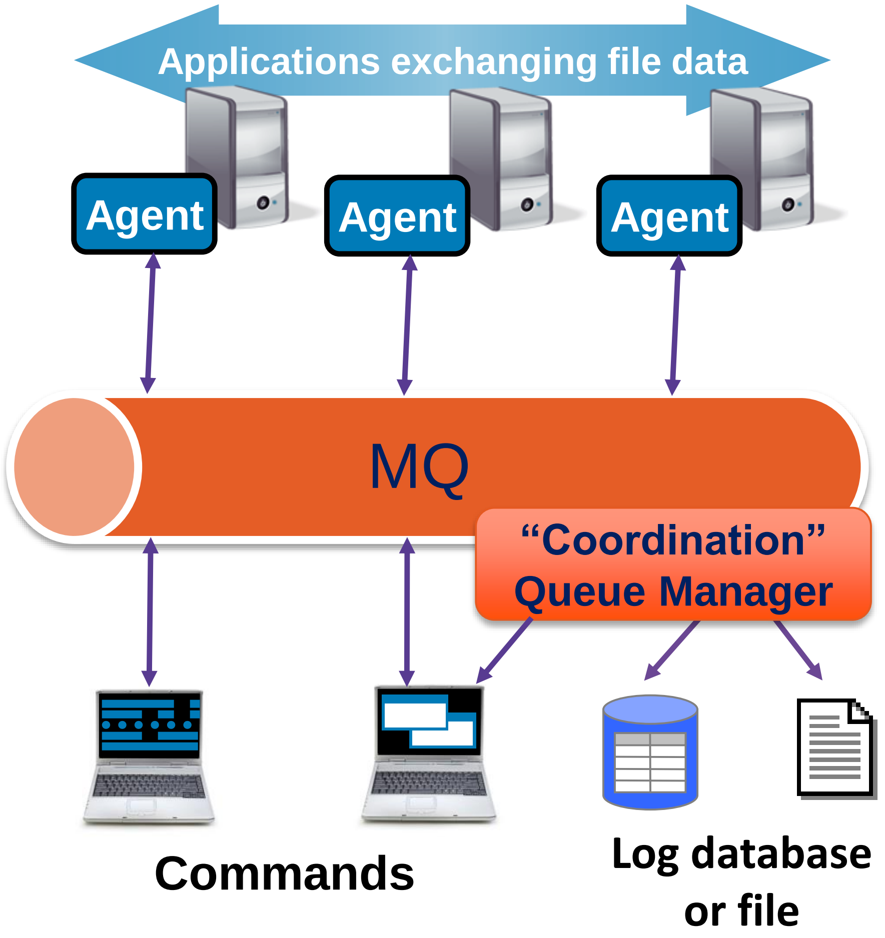
- Managed File Transfer reuses the MQ network for managed file transfer and yields:
  - Operational savings and simplification
  - Reduced administration effort
  - Reduced skills requirements and maintenance





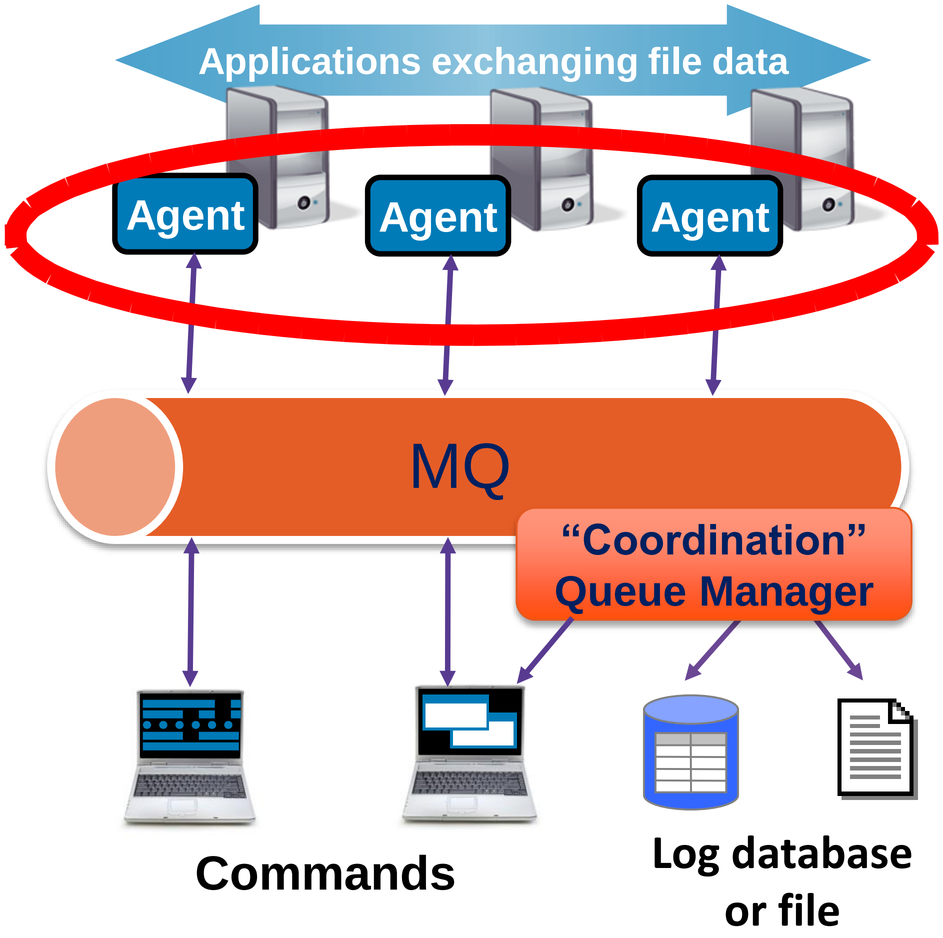
Components of a typical MQ MFT network

- Agents
  - The endpoints for managed file transfer operations
- Commands
  - Send instructions to agents
- Log database or file
  - A historical record of file transfers
- Coordination queue manager
  - Gathers together file transfer events


Applications exchanging file data
Agent
Agent
Agent
MQ
"Coordination" Queue Manager
Commands
Log database or file

Agents

- Act as the end points for file transfers
- Long running MQ applications that transfer files by splitting them into MQ messages
  - Efficient transfer protocol avoids excessive use of MQ log space or messages building up on queues
- Multi-threaded file transfers
  - Can both send and receive multiple files at the same time
- Generate a log of file transfer activities which is sent to the “coordination queue manager”
  - This can be used for audit purposes
- Associated with one particular queue manager (any supported version)
  - Agent state on queues


Applications exchanging file data
Agent
Agent
Agent
MQ
“Coordination” Queue Manager
Commands
Log database or file

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

NOTES

Notes on: Agents

- MFT agent processes define the end-points for file transfer. That is to say that if you want to move files off a machine, or onto a machine – that machine would typically need to be running an agent process
- Agent processes are long running MQ applications that oversee the process of moving file data in a managed way. Each agent monitors a ‘command’ queue waiting for messages which instruct it to carry out work, for example file transfers

NOTES

Notes on: Agents

- The MFT agent process needs connectivity to an MQ queue manager to do useful work. It can connect either directly to a queue manager running on the same system, or as an MQ client using an embedded version of the MQ client library (which is kept completely separate to any other MQ client libraries that may or may not already have been installed onto the system)*
  - Each agent requires its own set of MQ queues – which means that an agent is tied to the queue manager where these queues are defined
  - However – one queue manager can support multiple agents

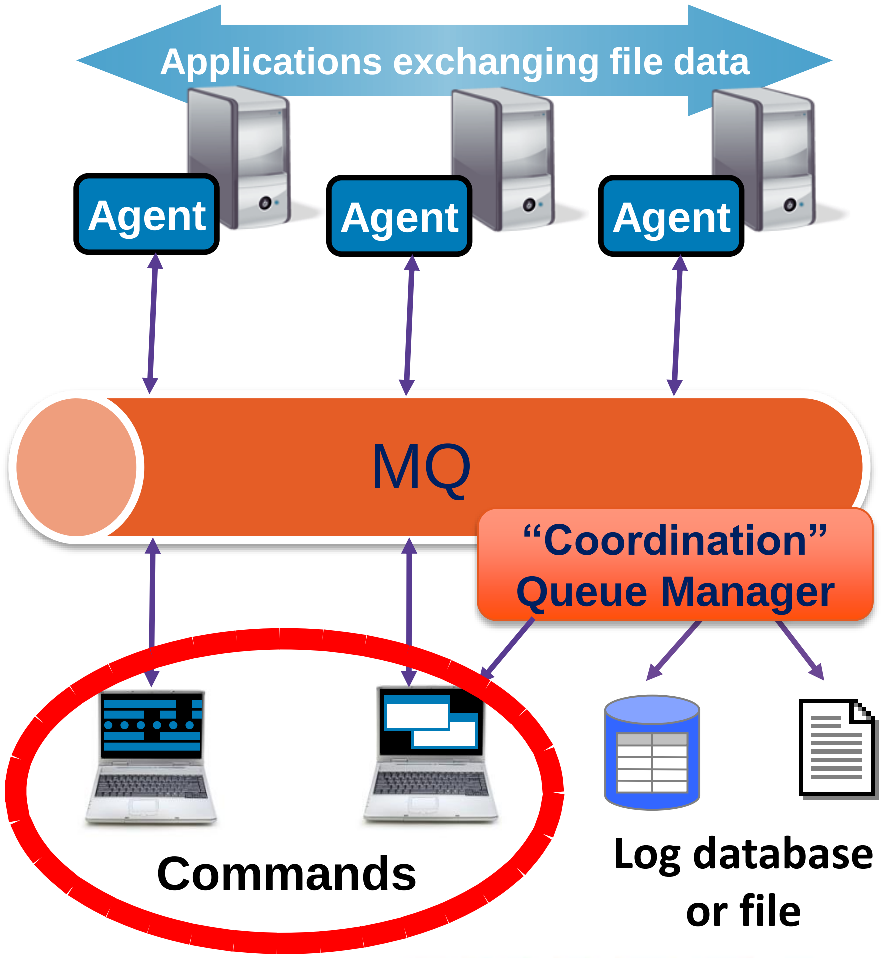
* Note: availability of direct (bindings) connectivity or MQ client based connectivity is dependent on the version of MQ MFT in use

- MQ Managed File Transfer on z/OS does not support the MQ client style of connectivity
- Managed File Transfer on distributed platforms has a ‘server’ and ‘client’ offering. The agent component of the ‘client’ offering is restricted to only supporting MQ client style connectivity. The agent component of the ‘server’ offering may be used either connectivity options

Commands

- Send instructions to agents and display information about agent configuration
  - Via MQ messages

- Many implementations of commands:
  - MQ Explorer plug-in
  - Command line programs
  - Open scripting language
  - JCL
  - Documented interface to program to


Applications exchanging file data
Agent
Agent
Agent
MQ
"Coordination" Queue Manager
Commands
Log database or file

NOTES

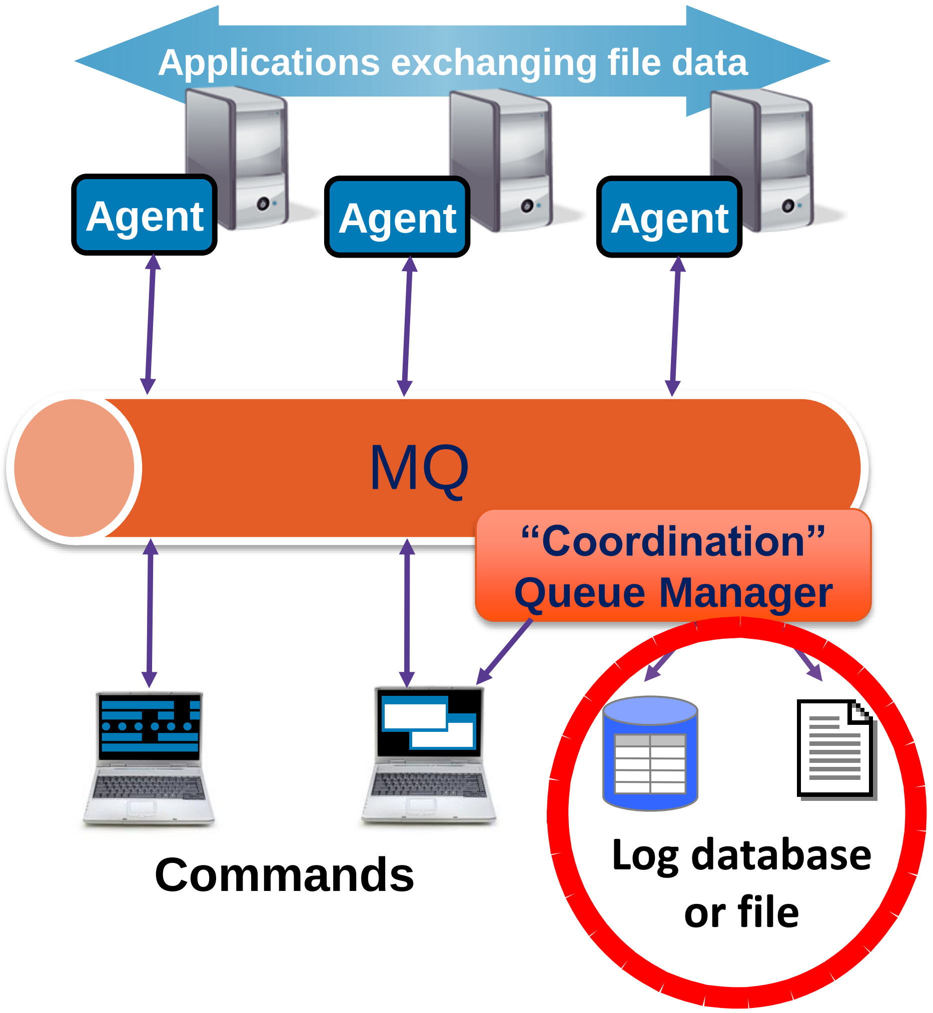
Notes on: Commands

- "Commands" is the name we have given to anything which instructs the MFT agent. As described on the previous slide, there are a wide range of command implementations including graphical and non-graphical command-line based commands
- Commands instruct the MFT agent by sending it messages. The messages themselves use a documented format which can easily be incorporated into your own applications
- The commands that are supplied with MFT can connect either as an MQ client (again based on embedded client libraries) or directly to a queue manager located on the same system

Log Database & File

- Keeps a historical account of transfers that have taken place
  - Who, where, when… etc.

- Implemented by the 'logger' component which connects to the coordination queue manager
  - Stand alone application
    - Can log to database or file
  - Or JEE application
    - Can log to database only


Applications exchanging file data
Agent
Agent
Agent
MQ
"Coordination" Queue Manager
Commands
Log database or file

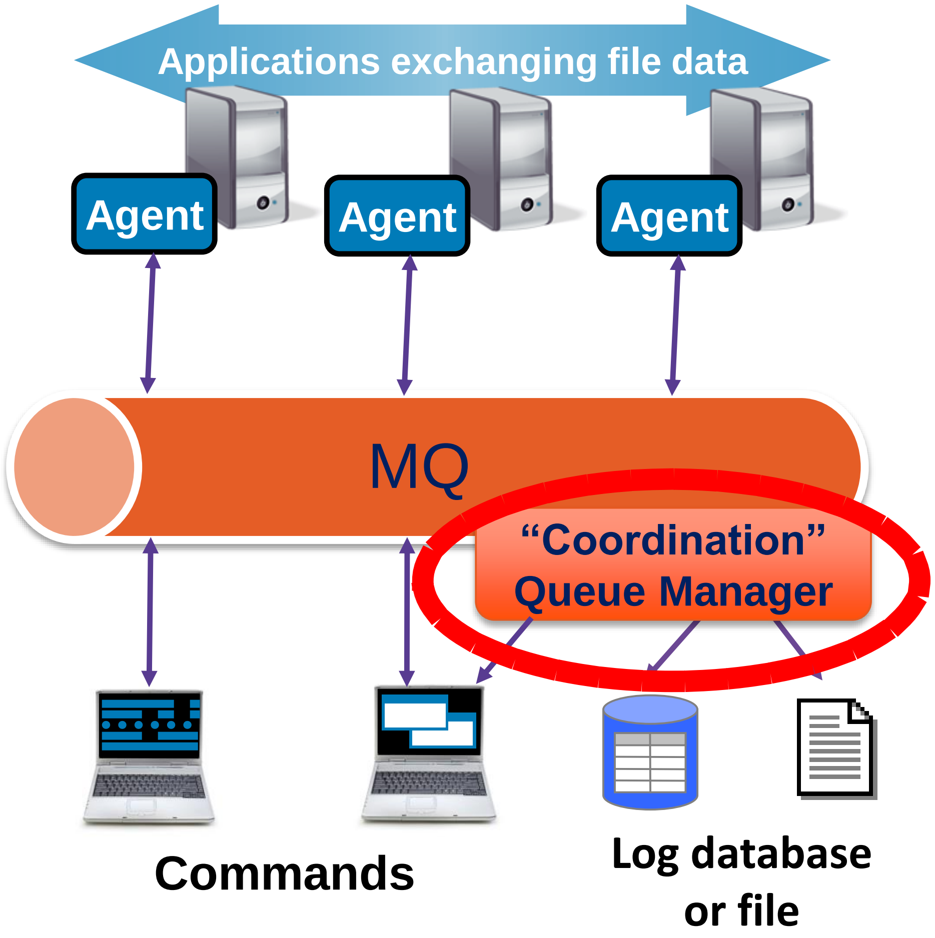
Notes on: Log Database

NOTES

- Managed File Transfer can record a historical account of file transfers to a database using the ‘database logger’ component.
  - This component is available to run as a stand alone process or as a JEE application
- The information used to populate the log database is generated, as MQ messages, by the MFT agents participating in file transfers. This is routed to a collection point in the MQ network, referred to as the ‘coordination queue manager’ (see next slides). The database logger component subscribes to the messages produced by agents and reliably enters them into a database.

Coordination Queue Manager

- Gathers together information about events in the file transfer network
- Not a single point of failure
  - Can be made highly available
  - Messages stored + forwarded
- MQ v7 publish / subscribe
  - Allows multiple log databases, command installs
  - Documented interface

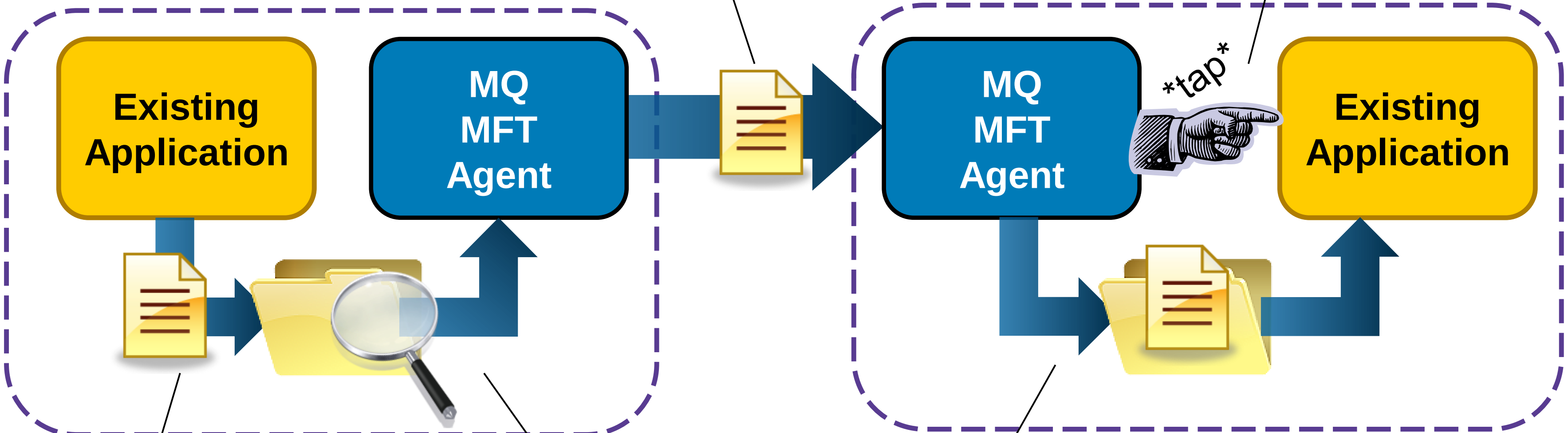

Applications exchanging file data
Agent
Agent
Agent
MQ
"Coordination" Queue Manager
Commands
Log database or file

NOTES

Notes on: Coordination queue manager

- The coordination queue manager is used as the gathering point for all the information about file transfers taking place between a collection of MFT agents
- The queue manager uses publish/subscribe (so it must be MQ version 7) to distribute this information to “interested parties” which typically include:
  - The MQ Explorer plug-in, which provides a graphical overview of MFT activity
  - The database logger component, which archives the information to a database
  - Some of the command line utilities which are part of the MFT product
- The format used to publish information is documented and can be used to develop 3rd party applications which process this data
- Although there is only a single ‘coordination’ queue manager for a given collection of agents it does not represent a point of failure:
  - MQ stores and forwards messages to the coordination queue manager when it is available – so if the coordination queue manager is temporarily unavailable no log data is lost
  - The ‘coordination’ queue manager can be made highly available using standard HA techniques such as MQ multi-instance queue manager or via a HA product such as PowerHA

Example usage of monitoring & program execution



3. MFT transports file to destination

5. MFT can also start another application to process the file

1. Application writes file to file system

2. Agent monitors file system, spots arrival of file and based on rules, transfers the file

4. At destination MFT writes file to file system

Notes on: Monitoring & Program Execution

NOTES

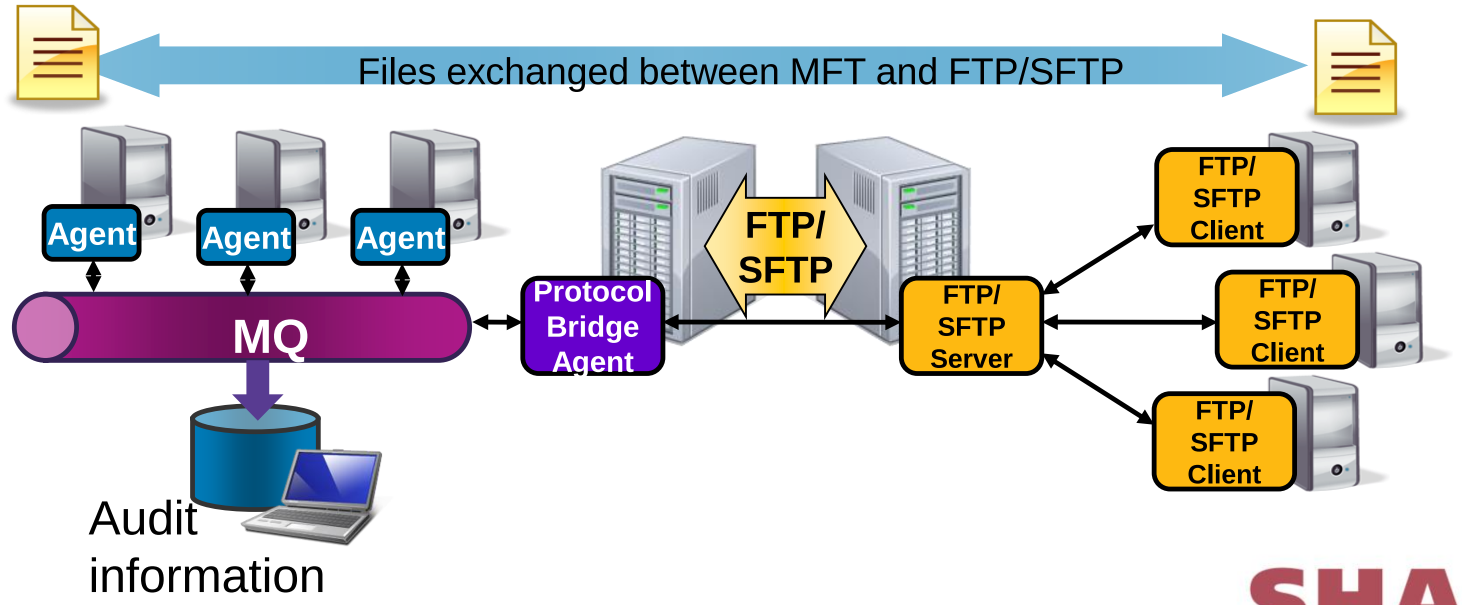
- Resource monitors work in two stages:
  1. Poll a resource (in this case the file system – but as we'll see later the 'resource' can also be an MQ queue) and identify that a condition has been met (perhaps the appearance of a file matching a particular pattern)
  2. Perform an action (which can include starting a managed file transfer, or running a script), optionally propagating information about the resource (for example the name of the file triggered on) into the action

- As shown on the previous slide, resource monitors are typically used to provide integration with an existing system without needing to make changes to the system

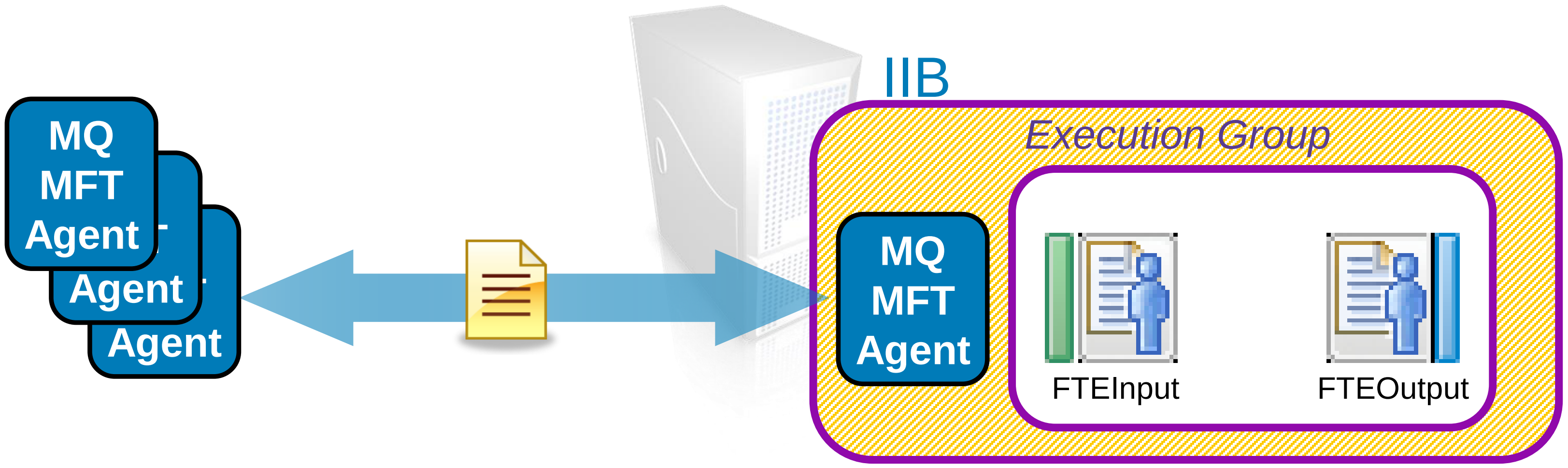
- Another function of MFT, used for integration with existing systems is the ability to execute programs or scripts both on the source or destination systems for a file transfer. This can be used to:
  - Start a program, on the source system, which generates the file data to be transferred prior to performing the managed file transfer
  - Start, or notify, a program on the destination system when the file data has been transferred – allowing it to process the data without having to poll

Protocol bridge agents

- Support for transferring files located on FTP and SFTP servers
  - The source or destination for a transfer can be an FTP or an SFTP server
- Enables incremental modernization of FTP-based home-grown solutions
  - Provides auditability of transfers across FTP/SFTP to central audit log
- Ensures reliability of transfers across FTP/SFTP with checkpoint restart
  - Fully integrated into graphical, command line and XML scripting interfaces
  - Just looks like another MFT agent…



IBM Integration Bus Nodes

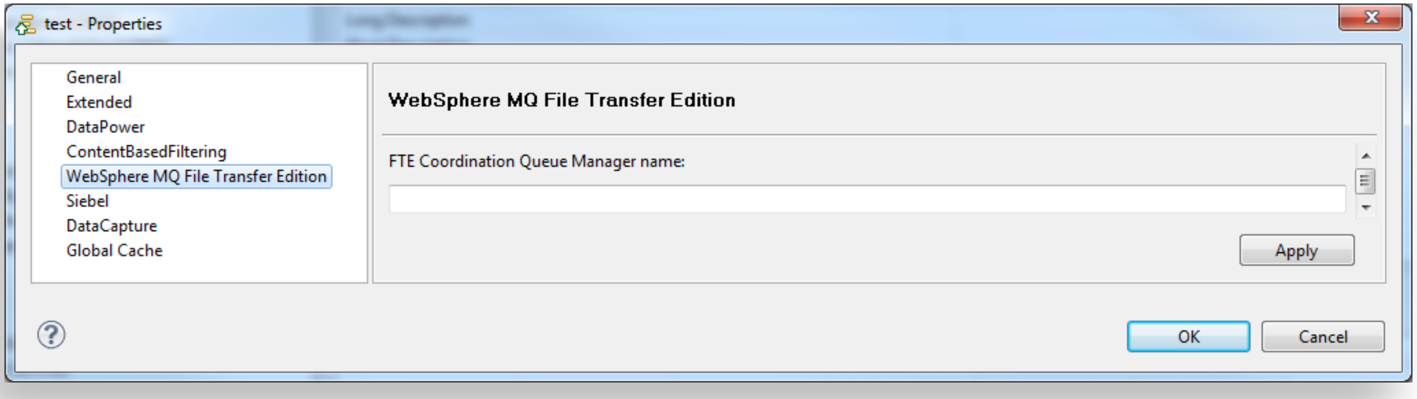


- FTEInput node
  - Build flows that accept file transfers from the MQ MFT network
- FTEOutput node
  - Build flows that are designed to send a file across an MQ MFT network

- When MQ MFT nodes are used in a flow an MFT agent is automatically started in the IIB integration server

Creating and running an agent in Integration Bus

- Install
  - FTE code is installed as part of Integration Bus install. No need for separate install
- Deployment
  - Agents run in the integration server JVMs. One agent per integration server
  - Coordinating queue manager defined as an integration server property


test - Properties

General
Extended
DataPower
ContentBasedFiltering
WebSphere MQ File Transfer Edition
Siebel
DataCapture
Global Cache

WebSphere MQ File Transfer Edition

FTE Coordination Queue Manager name:

Apply

OK Cancel

  - Agent name is derived from integration node and integration server name:
    - <integration node name>.<integration server name>
  - FTE Agents are created automatically by integration node including required queues
- Starting/stopping
  - Agent started when first FTE flow node using it is deployed
  - Agent stopped when last FTE flow node using it is un-deployed or stopped

Notes on: Creating and running an agent in IIB

NOTES

The core FTE product is installed as part of the normal IIB install. No additional components need to be installed apart from the MQExplorer add on which can be used to monitor transfers. It is not required for the transfers to work but is useful for monitoring what is happening with transfers.

Each integration server can be configured to run an FTE agent. This is done by setting the coordination queue manager property either using MBExplorer or the mqsichangeproperties command.

The agent name is derived by concatenating the integration node and integration server name together. If the name is too long to be a valid agent name then it is truncated. If it contains non-valid characters (any characters not supported in MQ queue names) an error is written to the local system log.

All the configuration for the agent is created when the coordination queue manager is set and also deleted when it is unset.

As well as creating the required config files, all the required queues are also created.

The actual agent is started when the first FTE flow node using it is started and stopped when the last FTE flow node using it is stopped.

If a node is deployed without setting the coordination queue manager then an agent will be created at deployment time using the integration node queue manager as the coordination queue manager. This is a temporary agent which is deleted when the last FTE flow node using it has been stopped. It is recommended that the integration server property is set even when the integration node queue manager is the coordination queue manager.

NOTES

Notes on: Administering an agent in IIB

The administration of the agent is very simple with almost all tasks being done automatically by the integration server.

All the config files created are identical to those used in a standalone agent.

Logging by the agent is written to the standard integration node user and service trace but the log file in the config directory is still written to as well.

A location under the workpath is used as the default transfer directory and also to stage files before they are transferred. To separate files being transferred by different nodes a directory structure is created which includes the flow name and node name.

As with the file nodes a mqsitransmit directory is used to build files up before transferring them. The names of files in these directories are mangled to stop clashes of files with the same name which are going to be transferred to different agents or different directories. If is recommended not to delete files from the transmit directory unless the message flow using them has been stopped and all files are deleted.

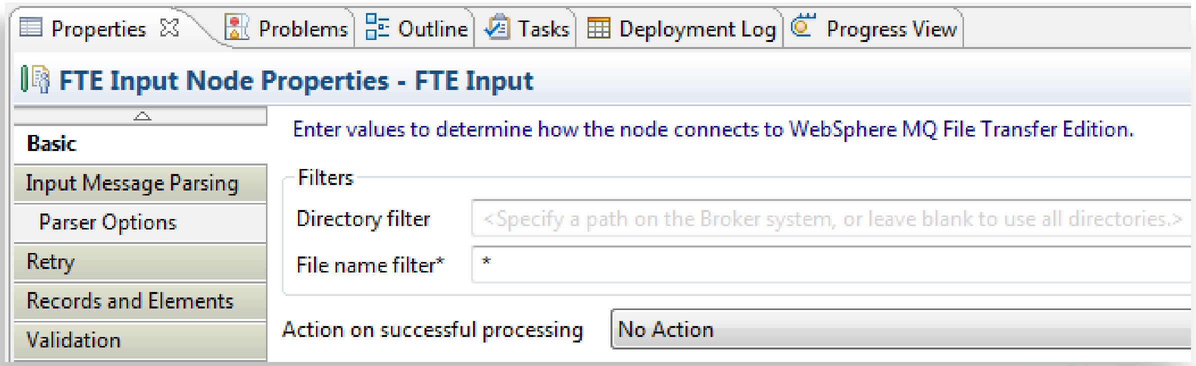
The FTE MQExplorer plugin is very useful to monitor the progress of any transfers which have been done.

Accessing agent from a message flow

- Messages received and sent from agent using the following flow nodes:
  - FTEInput node: receives any file transferred to integration server agent
  - FTEOutput node: constructs a file and sends a request to the integration server agent to transfer a file
- FTE nodes based on the existing file nodes allowing:
  - Record based processing
  - Stream parsing for large file support
  - File filtering
  - Can be used with all other flow nodes
- Agent is started or stopped based on whether any FTE nodes are running

FTE Input Node

- Consistent with file input node but makes full use of the powerof MQ MFT
- Timely
  - The FTE node is notified by the FTE agent when an inbound transfer is complete
  - The node processes the files in the transfer immediately.
  - Each file is processed independently
  - Can leave file unchanged after processing and just delete notification message
- Metadata
  - Metadata associated with the transfer is sent with the notification
  - Includes user defined metadata
- Filter
  - The node by default receives all transfers
  - Can specify which files to receive using a filter



FTE Input Node Properties - FTE Input

Enter values to determine how the node connects to WebSphere MQ File Transfer Edition.

Filters

Directory filter: <Specify a path on the Broker system, or leave blank to use all directories.>

File name filter*: *

Action on successful processing: No Action

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Notes on: FTEInput Node

NOTES

The FTEInput node has all the core function of the standard File input node but has been enhanced to make use of the powerful function provided by MQ MFT.

It does not require any polling mechanism to scan directories because it is directly trigger by the embedded agent when a file has arrived and the transfer is complete.

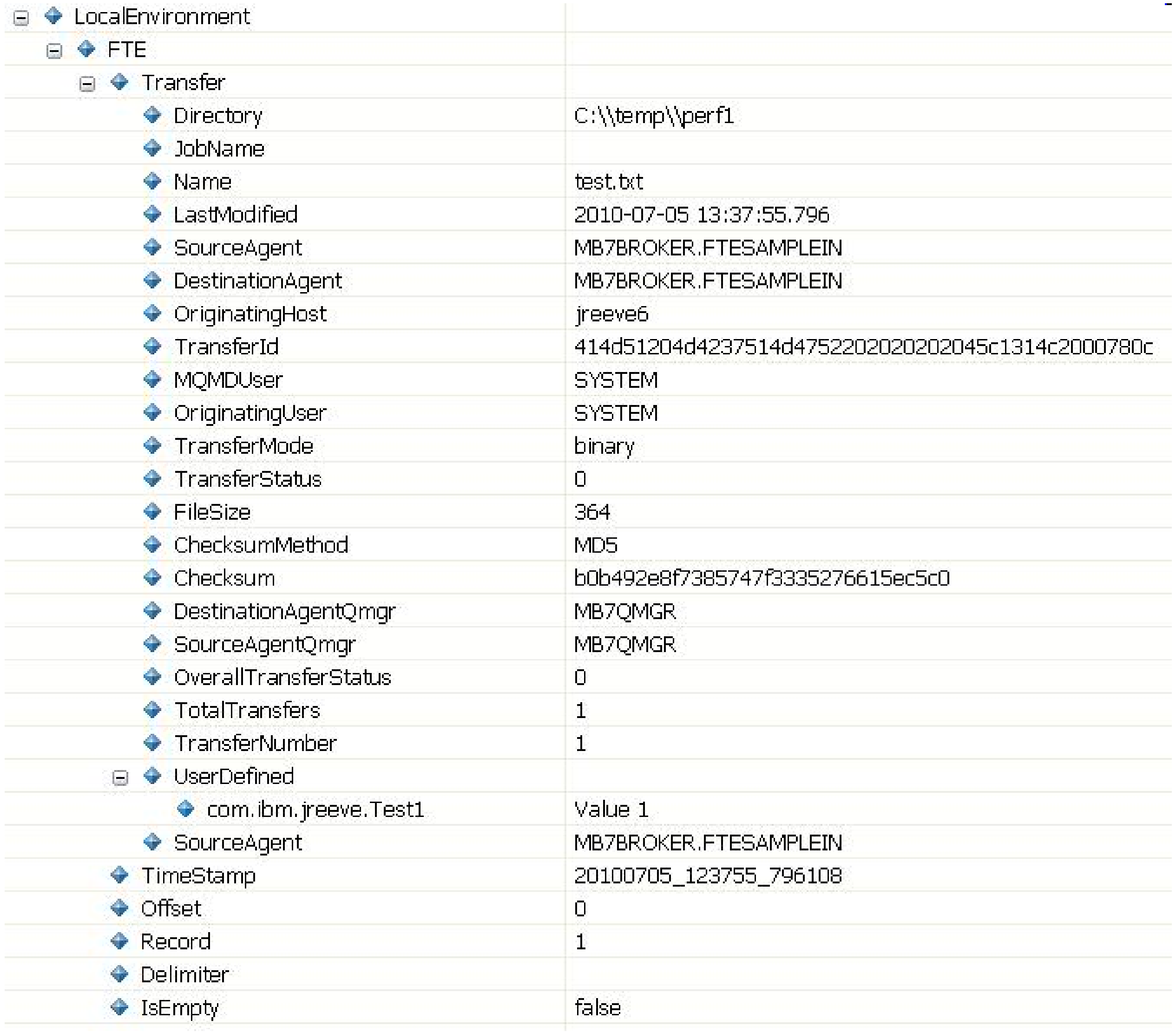
It processes each file in a transfer separately and can process each file in parallel.

As well as receiving the data from the transfer it also receives all the meta data associated with the transfer. This includes lots of information from MQ MFT but also can include user defined data.

Each FTE Input node can specify a filter of which files it wants to process. By default it processes all files. If two nodes both have a filter which matches a file then only one will get given it to process. By default, if no filter expression is given, then the node we accept any transferred file.

The file name filter accepts wild cards but the directory can either be blank (accept any files) or a string that must match exactly. Relative paths are allowed which are taken relative to the default transfer directory.

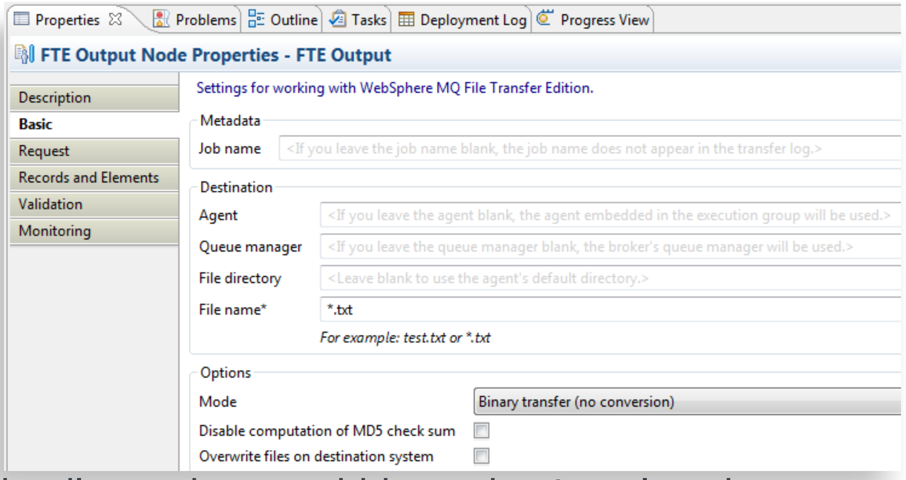
FTE Input Node – metadata



LocalEnvironment	
FTE	
Transfer	
Directory	C:\\temp\\perf1
JobName	
Name	test.txt
LastModified	2010-07-05 13:37:55.796
SourceAgent	MB7BROKER.FTESAMPLEIN
DestinationAgent	MB7BROKER.FTESAMPLEIN
OriginatingHost	jreeve6
TransferId	414d51204d4237514d4752202020202045c1314c2000780c
MQMDUser	SYSTEM
OriginatingUser	SYSTEM
TransferMode	binary
TransferStatus	0
FileSize	364
ChecksumMethod	MD5
Checksum	b0b492e8f7385747f3335276615ec5c0
DestinationAgentQmgr	MB7QMGR
SourceAgentQmgr	MB7QMGR
OverallTransferStatus	0
TotalTransfers	1
TransferNumber	1
UserDefined	
com.ibm.jreeve.Test1	Value 1
SourceAgent	MB7BROKER.FTESAMPLEIN
TimeStamp	20100705_123755_796108
Offset	0
Record	1
Delimiter	
IsEmpty	false

FTE Output Node

- Consistent with file output node but makes full use of the power of MQ MFT
- Transfer details
  - The destination agent, directory, etc. are defined on the node



FTE Output Node Properties - FTE Output

Settings for working with WebSphere MQ File Transfer Edition.

Metadata

Job name: <If you leave the job name blank, the job name does not appear in the transfer log.>

Destination

Agent: <If you leave the agent blank, the agent embedded in the execution group will be used.>

Queue manager: <If you leave the queue manager blank, the broker's queue manager will be used.>

File directory: <Leave blank to use the agent's default directory.>

File name*: *.txt

*For example: test.txt or *.txt*

Options

Mode: Binary transfer (no conversion)

Disable computation of MD5 check sum: ☐

Overwrite files on destination system: ☐

  - All details can be overridden using Local environment
  - Support also for wild card file names
- Staging
  - Uses a local staging directory to build up a file record by record for transfer
  - Once a file is finished a request is sent to the FTE Agent to transfer the file
- Metadata
  - User data – if provided in the Local Environment - is sent with the transfer
  - Other metadata is generated by the FTE Agent or by the integration node.
  - Ant scripts – give details of ant scripts to run on remote agent

NOTES

Notes on: FTEOutput Node

The FTEOutput node makes use of the core function provided by File nodes.

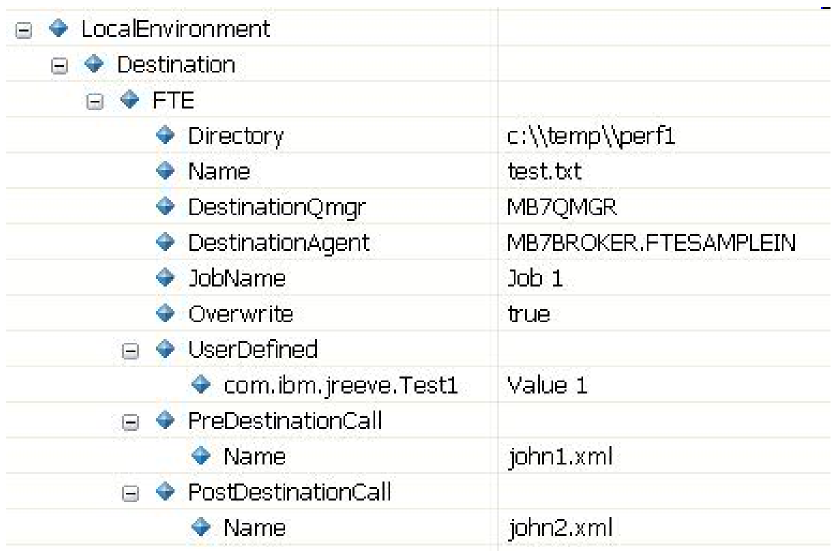
The main properties on the node are details of where to transfer the files to. These can all be overridden using the local environment.

The main difference with an FTEOutput node is that the destination the file is to be written to is not local to the integration node file system but instead is a file system on a remote agent. The node first writes it to a staging directory on the local file system and then sends a request to the embedded agent to transfer it.

Files are built up in the mqsitransmit directory just like with a file node before being moved to the final file name once the file is complete and the transfer request is sent to the FTE agent. The name in the staging area is the same as the name the file will be transferred to unless a file with that name already exists on the local file system. If it exists than a number is appended to the file. The file name it is transferred to on the remote system is not effected by this and will not have the number appended.

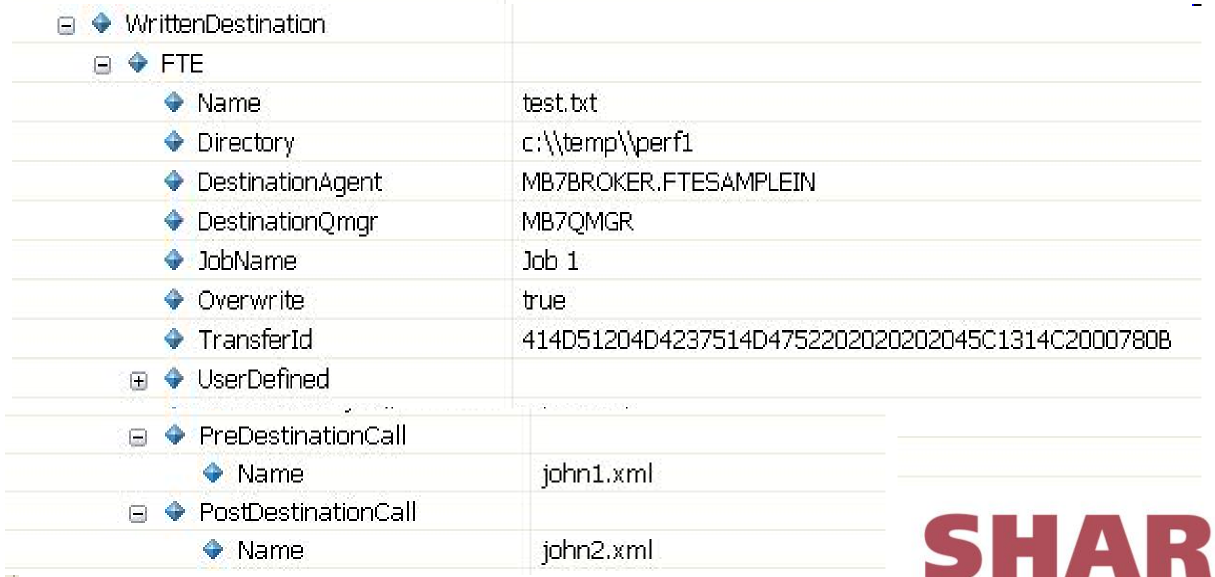
FTE Output Node – Overrides and metadata

Overrides:



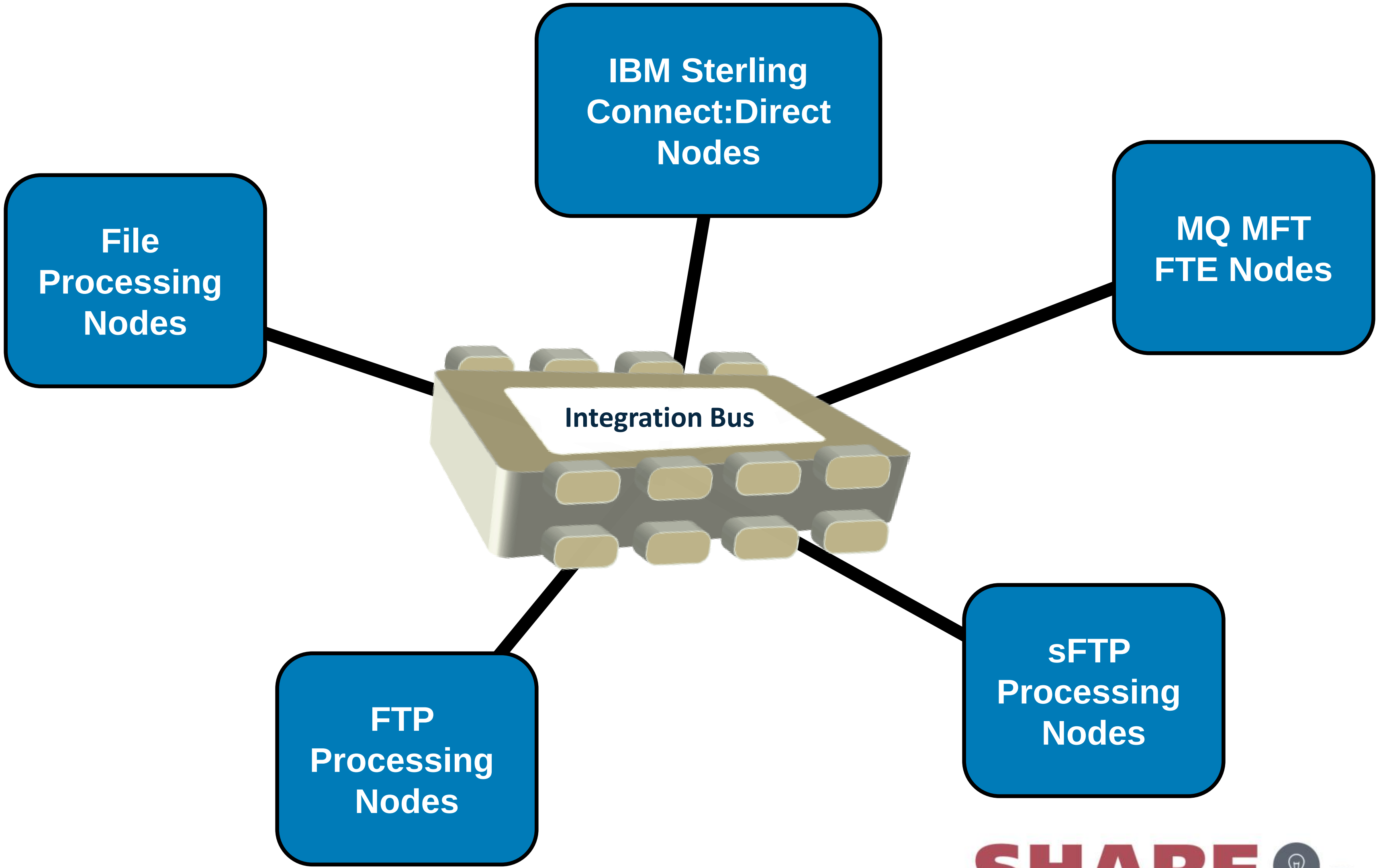
LocalEnvironment	
Destination	
FTE	
Directory	c:\\temp\\perf1
Name	test.txt
DestinationQmgr	MB7QMGR
DestinationAgent	MB7BROKER.FTESAMPLEIN
JobName	Job 1
Overwrite	true
UserDefined	
com.ibm.jreeve.Test1	Value 1
PreDestinationCall	
Name	john1.xml
PostDestinationCall	
Name	john2.xml

Metadata:



WrittenDestination	
FTE	
Name	test.txt
Directory	c:\\temp\\perf1
DestinationAgent	MB7BROKER.FTESAMPLEIN
DestinationQmgr	MB7QMGR
JobName	Job 1
Overwrite	true
TransferId	414D51204D4237514D4752202020202045C1314C2000780B
UserDefined	
PreDestinationCall	
Name	john1.xml
PostDestinationCall	
Name	john2.xml

File integration in IBM Integration Bus



IBM Integration Bus : FileInput Node

Notes on: FileInput Node

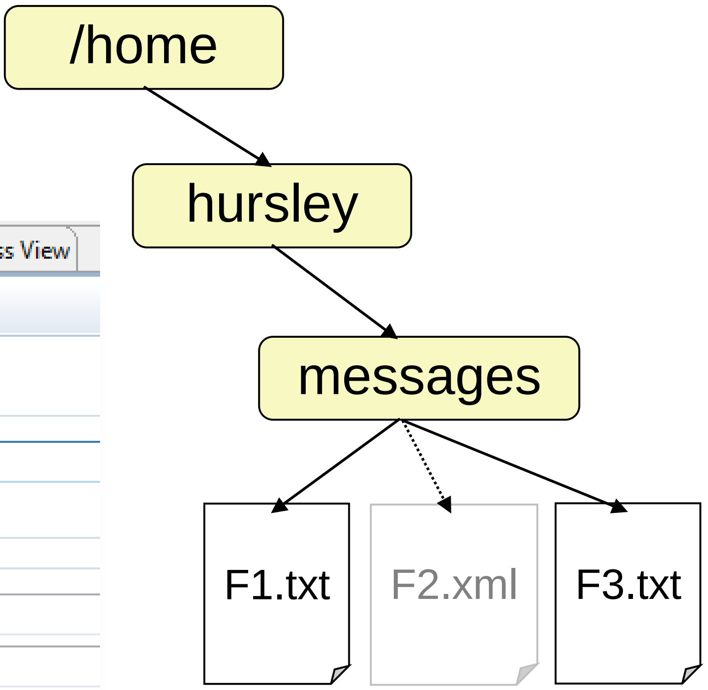
NOTES

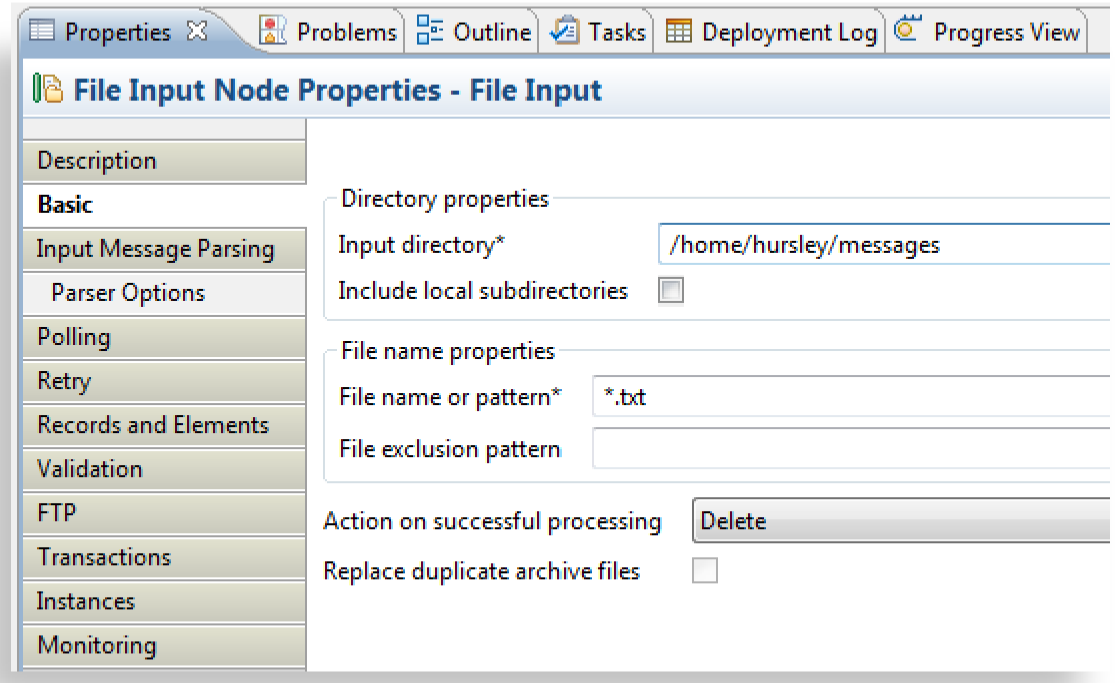
- The FileInput nodes reads data from a file and triggers the start of processing in the flow. Three main areas will be covered in detail:
  - The mechanism used to detect a file is ready to be processed
  - The splitting of the file up into records
  - The archiving or deleting of the file once processing has been finished

FileInput node – Operation

- Scans a pre-configured directory (relative or absolute) for files that match a given specification
- Locked files are ignored until they become unlocked

File Input





File Input Node Properties - File Input

Directory properties

Input directory*	/home/hursley/messages
Include local subdirectories	☐

File name properties

File name or pattern*	*.txt
File exclusion pattern	

Action on successful processing	Delete
Replace duplicate archive files	☐

FileInput node – Record Detection

- Handling options (on the Records and Elements tab):
  - Whole file
  - Fixed Length *
  - Delimited *
  - Parsed Record Sequence *

    *Note: * - results in separate records – message flow is invoked multiple times*

- Only requires one record to be in memory at any one time
  - Allows very large files (Gigabyte) to be streamed efficiently
  - Streaming possible with DFDL, MRM (CWF and TDS) and XMLNSC parsers only

- If connected, ‘End Of Data’ terminal is triggered at end of file
  - Empty BLOB message and a LocalEnvironment.File structure

Record Detection Examples

- With input file `Boston|10|San Francisco|9|Seattle|8|`
- Whole file
  - Propagates one message:

Boston|10|San Francisco|9|Seattle|8|

- Fixed length (size = 10 bytes)
  - Propagates four messages:

Boston|10|	San Franci	sco|9|Seat	tle|8|

- Delimited (character = ‘|’)
  - Infix: propagates seven messages

Boston	10	San Francisco	9	Seattle	8	“”

  - Postfix: propagates six messages

Boston	10	San Francisco	9	Seattle	8

Record Detection Examples 2

- With input file:

```
<cities><city name="Boston" rating="10"></cities>
<cities><city name="San Francisco" rating="9"></cities>
<cities><city name="Seattle" rating="8"></cities>
```

- Parsed record sequence
  - When the parser is set to XMLNSC, this propagates three XML messages

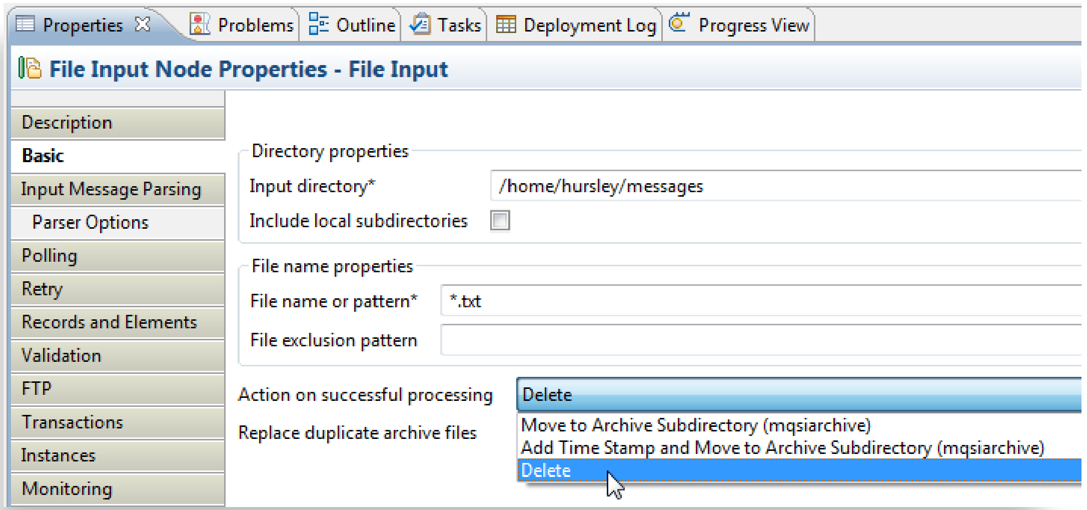
```
<cities><city name="Boston" rating="10"></cities>
```

```
<cities><city name="San Francisco" rating="9"></cities>
```

```
<cities><city name="Seattle" rating="8"></cities>
```

FileInput node – Archiving

- Upon successful processing, file is either deleted or moved to an mqsiarchive subdirectory
- Dealing with files with duplicate names:
  - Option to include timestamp in archived filename
  - Option to replace any existing file



File Input Node Properties - File Input

Directory properties

Input directory* /home/hursley/messages

Include local subdirectories ☐

File name properties

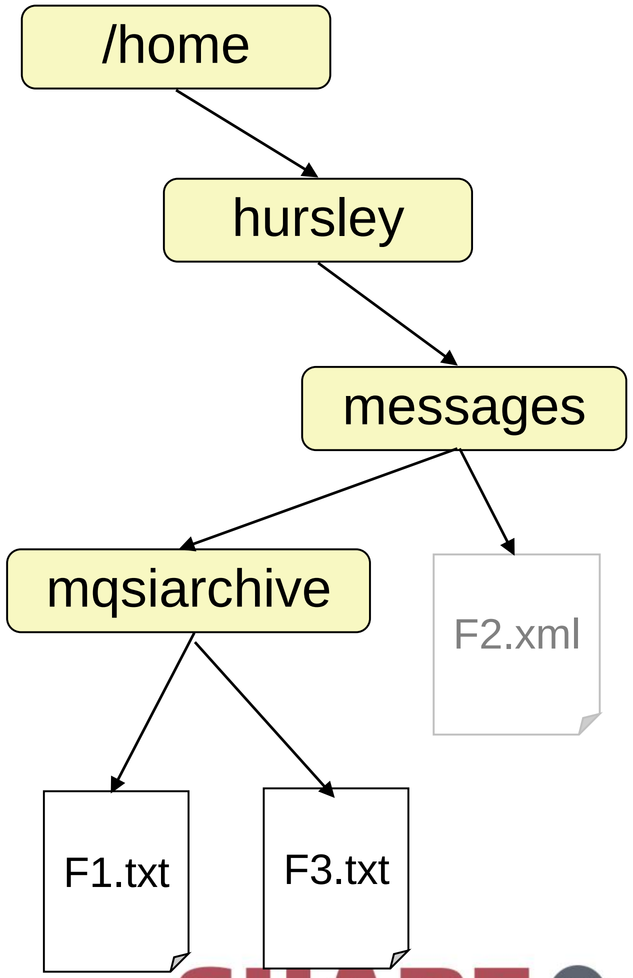
File name or pattern* *.txt

File exclusion pattern

Action on successful processing: Delete

Replace duplicate archive files

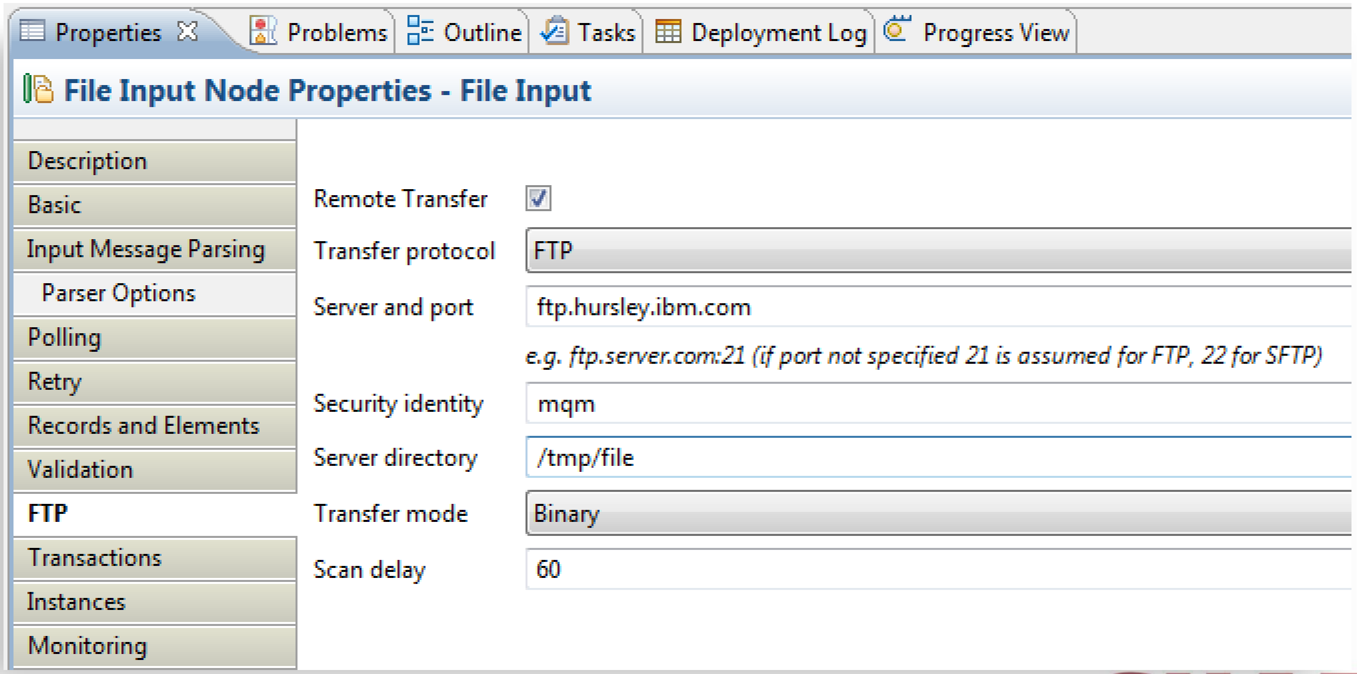
Move to Archive Subdirectory (mqsiarchive)
Add Time Stamp and Move to Archive Subdirectory (mqsiarchive)
Delete



Complete your session evaluations online at www.SHARE.org/Orlando-Eval

FileInput node – using FTP and SFTP

- When active, FTP settings cause the node to periodically transfer files on a remote server to the local directory for input.
- Security Identity 'UserMapping' set using runtime command:
  - mqsisetdbparms BROKER –n ftp::UserMapping –u USER –p PASS



File Input Node Properties - File Input

Remote Transfer	✓
Transfer protocol	FTP
Server and port	ftp.hursley.ibm.com
	e.g. ftp.server.com:21 (if port not specified 21 is assumed for FTP, 22 for SFTP)
Security identity	mqm
Server directory	/tmp/file
Transfer mode	Binary
Scan delay	60

IBM Integration Bus: FileOutput Node

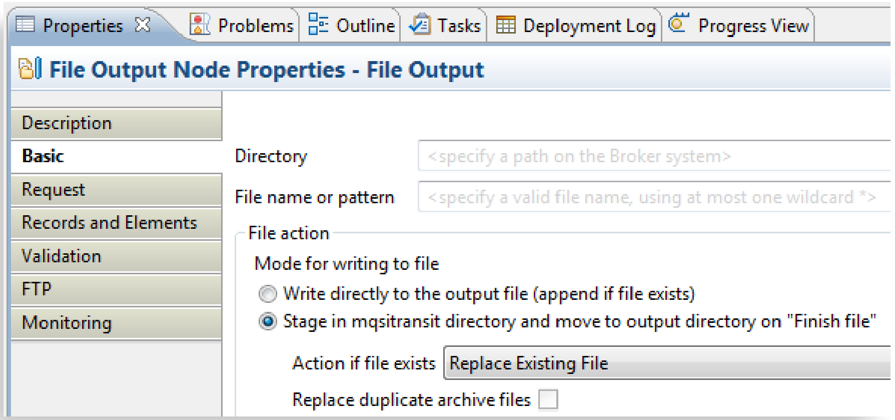
NOTES

Notes on: FileOutput Node

- The file output node is used to create and write to a file anywhere in the middle of a flow:
  - How it decides which file to create and where
  - How a file is created using a series of records appended
  - What happens if the file it attempts to create already exists

FileOutput – Writing files

- In the simplest scenario, the received message body is written to the pre-configured file:



Properties | Problems | Outline | Tasks | Deployment Log | Progress View

File Output Node Properties - File Output

Description | **Basic** | Request | Records and Elements | Validation | FTP | Monitoring

Directory: <specify a path on the Broker system>

File name or pattern: <specify a valid file name, using at most one wildcard *>

File action

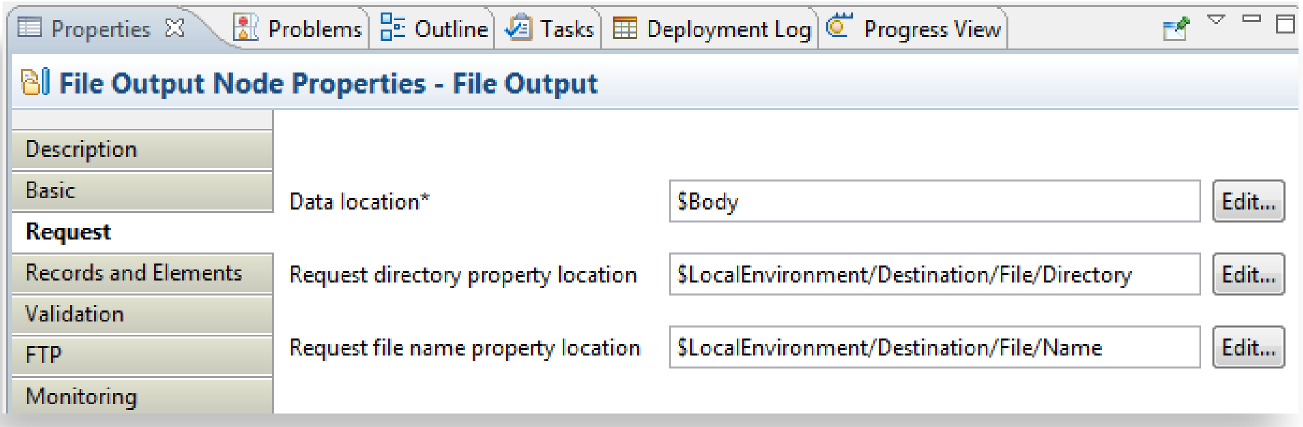
Mode for writing to file

- ○ Write directly to the output file (append if file exists)
- ● Stage in mqsitransit directory and move to output directory on "Finish file"

Action if file exists: Replace Existing File

Replace duplicate archive files ☐

- When writing to the output file, the wildcard (if present) is replaced with the value of LocalEnvironment.Wildcard.WildcardMatch
  - Allows you to preserve elements of a filename during processing



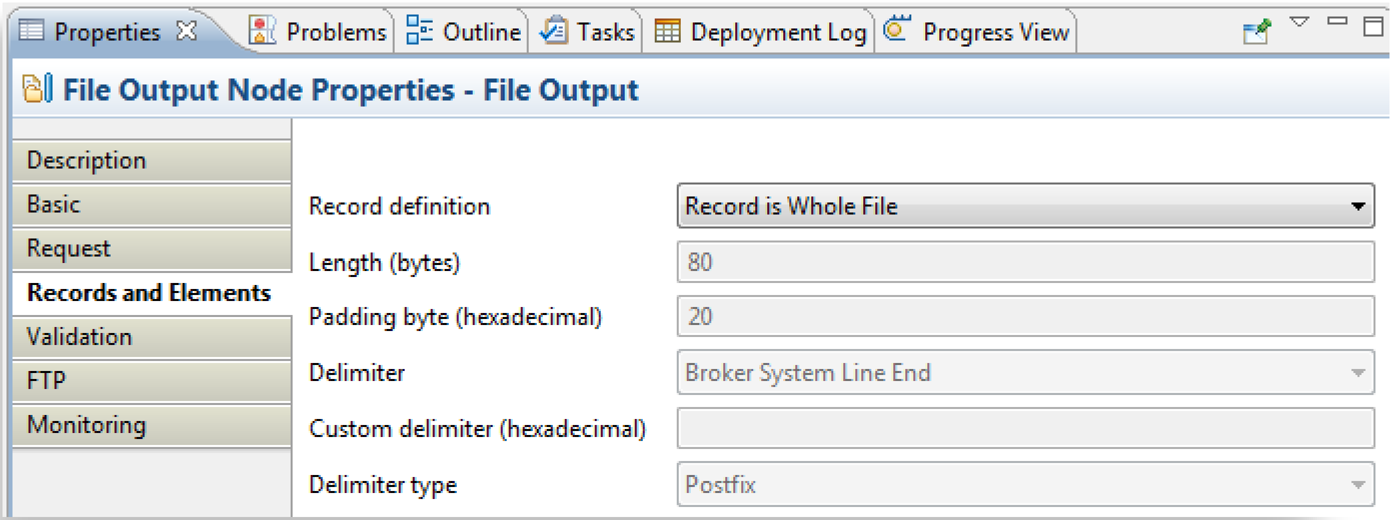
Properties | Problems | Outline | Tasks | Deployment Log | Progress View

File Output Node Properties - File Output

Description | Basic | **Request** | Records and Elements | Validation | FTP | Monitoring

Data location*	$Body	Edit...
Request directory property location	$LocalEnvironment/Destination/File/Directory	Edit...
Request file name property location	$LocalEnvironment/Destination/File/Name	Edit...

Appending Records


Properties | Problems | Outline | Tasks | Deployment Log | Progress View

File Output Node Properties - File Output

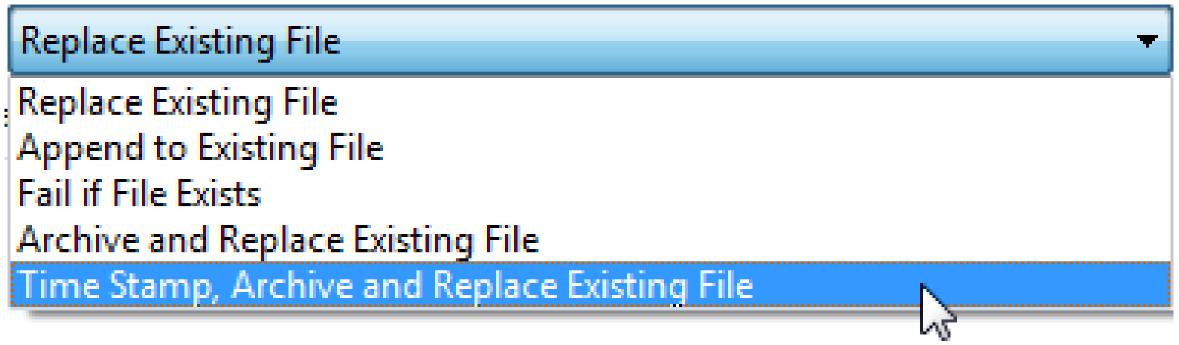
Description
Basic
Request
Records and Elements
Validation
FTP
Monitoring

Record definition	Record is Whole File
Length (bytes)	80
Padding byte (hexadecimal)	20
Delimiter	Broker System Line End
Custom delimiter (hexadecimal)	
Delimiter type	Postfix

- “Records and Elements” tab defines how multiple writes to the same file are handled
- Record definition options:
  - Record is Whole File – close file automatically after first write
  - Record is Unmodified Data – the message bit-stream appended to file
  - Record is Fixed Length Data – specify length in bytes and padding character
  - Record is Delimited Data – specify delimiter and infix/postfix option
- Unless “Record is Whole File” is selected, the file will be closed when the “Finish File” terminal is triggered
  - The Finish File message is propagated on to the FileOutput’s “End Of Data” terminal
- The mqsitransit subdirectory holds all files that have not yet been closed

Options if the file already exists

- Output file action (basic tab)

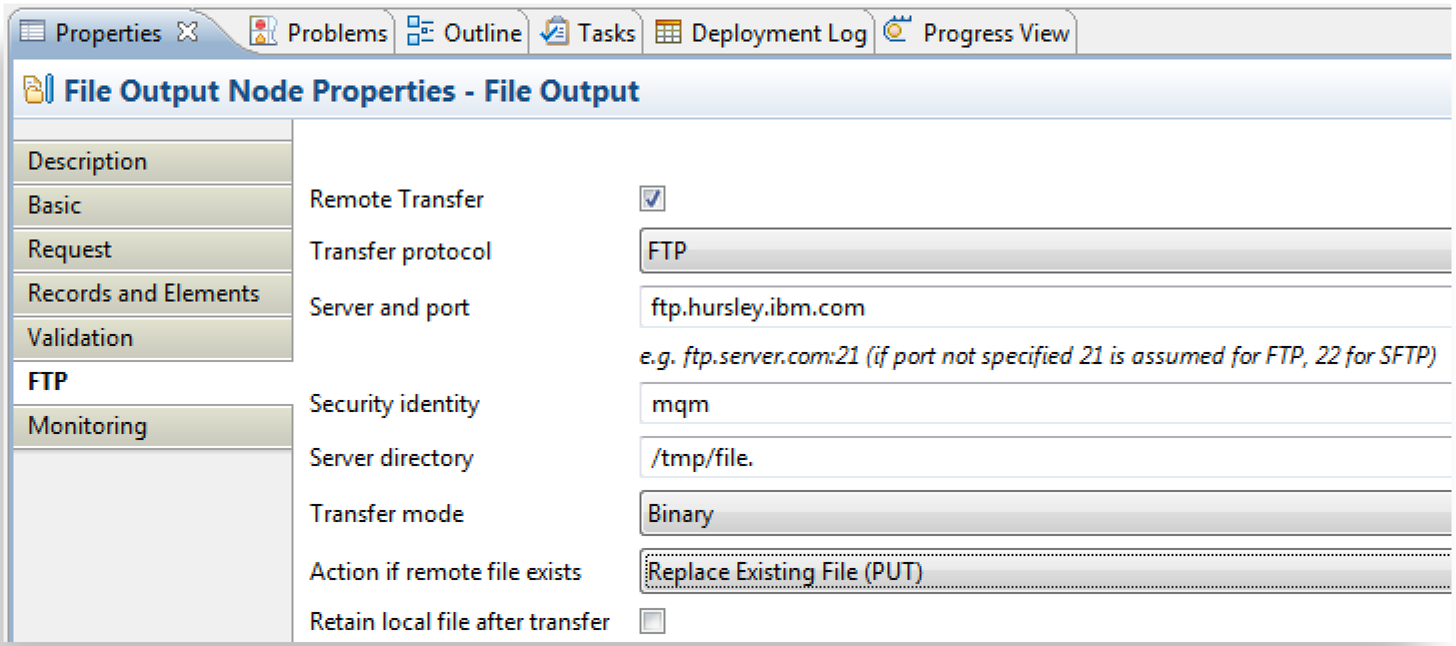


Replace Existing File

Replace Existing File
Append to Existing File
Fail if File Exists
Archive and Replace Existing File
Time Stamp, Archive and Replace Existing File

- If the file already exists
  – Replace it
  – Append to it
  – Go down failure terminal
  – Move to mqsiarchive subdirectory and replace
  – Add timestamp, move to mqsiarchive subdirectory and replace

FileOutput - FTP Support



File Output Node Properties - File Output

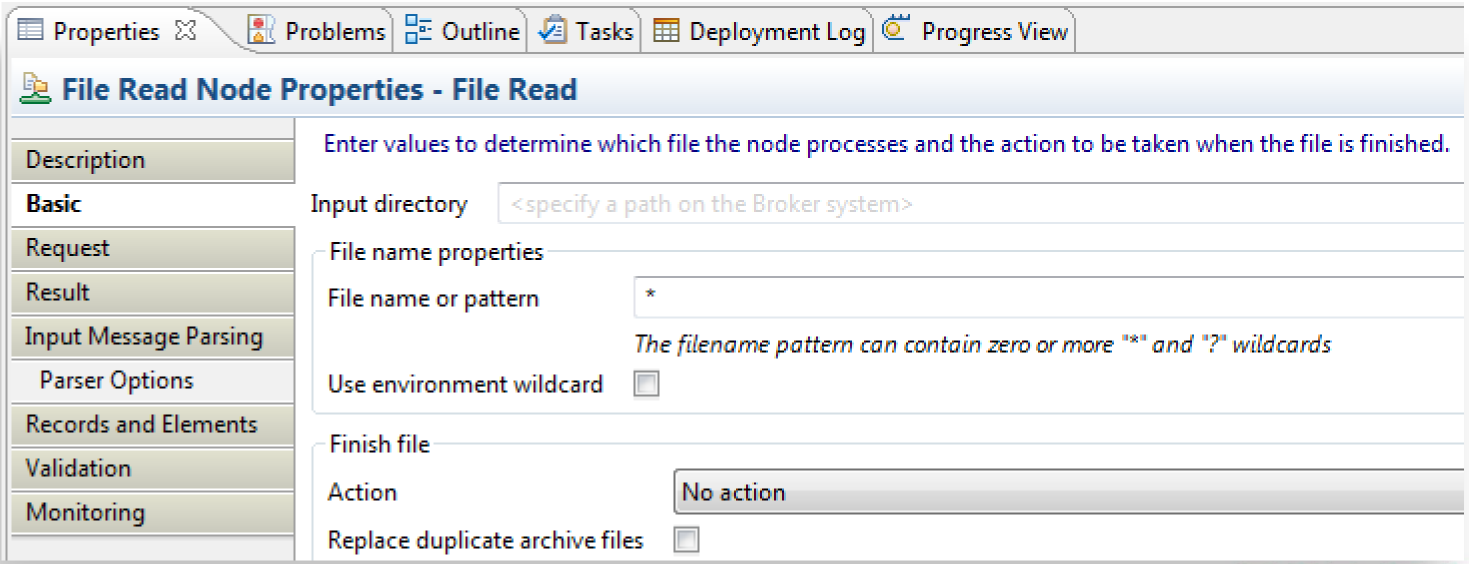
Remote Transfer	✓
Transfer protocol	FTP
Server and port	ftp.hursley.ibm.com
	e.g. ftp.server.com:21 (if port not specified 21 is assumed for FTP, 22 for SFTP)
Security identity	mqm
Server directory	/tmp/file.
Transfer mode	Binary
Action if remote file exists	Replace Existing File (PUT)
Retain local file after transfer	☐

- If enabled, whenever a complete file is closed an FTP transfer of the file is attempted to the supplied FTP server
- File is optionally deleted from the local file system when the transfer completes
- Transfer is synchronous
  - Use additional instances if throughput rate is an issue
- If remote file exists, choose either replace or append

IBM Integration Bus : FileRead Node

FileRead node

- File Read Node
  - Reads data in middle of flow like an MQGet or a TCPIPReceive node
  - Reads either whole file contents or one record from the file
  - Allows user to override the file to be read and the offset within the file to start reading from
- Has a No match terminal which the message is sent to if it can not find a file or a record in the file



File Read Node Properties - File Read

Enter values to determine which file the node processes and the action to be taken when the file is finished.

Input directory: <specify a path on the Broker system>

File name properties

File name or pattern: *

The filename pattern can contain zero or more "" and "?" wildcards*

Use environment wildcard: ☐

Finish file

Action: No action

Replace duplicate archive files: ☐

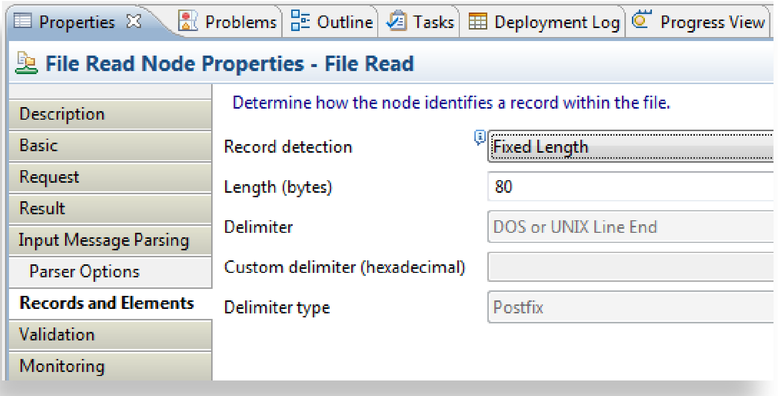
NOTES

Notes on: FileRead Node

- FileRead node behaves like a MQGet or TCPIPReceive node in the sense that it reads in data within a flow without first sending data out. For example: MQGet reads a message from a queue, TCPIPReceive reads data from a TCPIP input stream and the Fileread node reads data from a file.
- Can either read the whole contents of a file or a single record from the file and then parses and constructs a message to propagate down the flow.
- The node is very configurable both at design time and during runtime where most properties can be overridden based on the message content.
- The basic properties are similar to a File input node where the details of the file to process are given.

Defining which record to read and propagate

- Where the record starts
  - Defaults to the beginning of the file
  - Give the offset into the file to start record from based on contents of the message
- Where the record ends
  - Define the record detection mechanism:
    - Whole file
    - Fixed size
    - Delimited
    - Parser

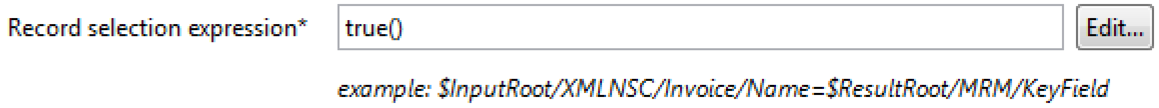


File Read Node Properties - File Read

Determine how the node identifies a record within the file.

Record detection	Fixed Length
Length (bytes)	80
Delimiter	DOS or UNIX Line End
Custom delimiter (hexadecimal)	
Delimiter type	Postfix

- Which record to propagate
  - Define an expression to specify which record to propagate
  - Node iterates through all records from the start offset until one matches
  - Only propagates the first match



Record selection expression* true() Edit...

example: $InputRoot/XMLNSC/Invoice/Name=$ResultRoot/MRM/KeyField

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

SHARE in Orlando 2015

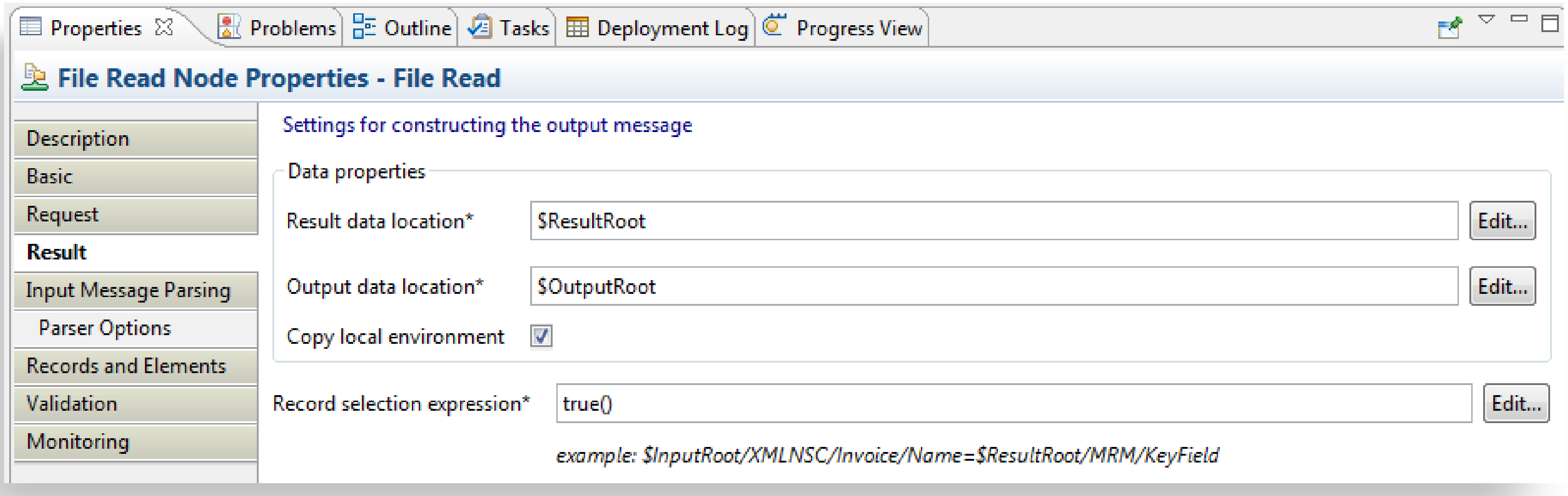
Notes on: Defining which record to read and propagate

NOTES

- Three key pieces of information need to be defined to create and propagate a record from the file.
- Where the record starts. By default the file read node always starts the reading of a record from the beginning of the file. It does not remember or store the result of the last record read so the next time through the node will start from the same place. Unless the user specifies in the local environment where to start from. It is possible to configure the node to use the end record offset from a previous fileread node as the start of the current record.
- Where the record ends. The normal record detection mechanisms are used to find the end of the record. Fixed size just reads that many bytes, delimited scans for a delimiter and parser uses an integration node parser to determine the end (like XMLNSC if the record is an XML document). When using fixed size it is possible to override the length being used using the message content or local environment.
- Which record to propagate. After finding the record the *Record selection expression* is evaluated. If it is true then the record is propagated otherwise the next record is found using the end of the last record as the start of the new one. The process is repeated until either the expression is true or the end of file is found. Only the first matching record is propagated. If no record matches then the file gets sent to the no match terminal.

Constructing the message using the record read location

- Which part of the record read to propagate
  - Result data location
- Where to put the record in the outgoing message
  - Output data location



File Read Node Properties - File Read

Settings for constructing the output message

Data properties

Result data location*	$ResultRoot
Output data location*	$OutputRoot
Copy local environment	✓
Record selection expression*	true()

example: $InputRoot/XMLNSC/Invoice/Name=$ResultRoot/MRM/KeyField

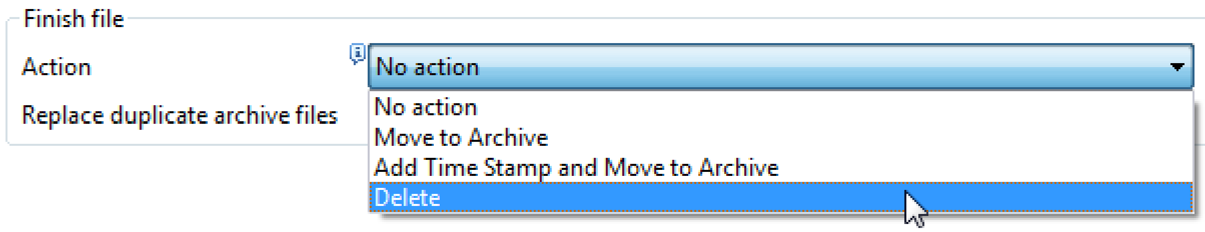
Notes on: Constructing the message using the record read location

NOTES

- The *Result* panel has properties which specify how the outgoing message is constructed based on the contents of the file and the incoming message.
- By default, the whole incoming message is replaced with the contents of the record retrieve from the file.
- The Result data location is used to extract a piece of information from the file to insert into the outgoing message.
- The Output data location is used to find the location to write the extracted information to.

Changing the file disposition after the read

- By default file is left unchanged after read
- Disposition change is always done when the FileRead node executes Finish File:
  - The end of the file is reached
  - A message is sent to the finish file terminal
- Following actions are available:



Finish file

Action: No action

Replace duplicate archive files

- Archived files are moved to the mqsiarchive directory
- Can override the archive directory and archive name using local environment or data in the message

Notes on: Changing the file description after the read

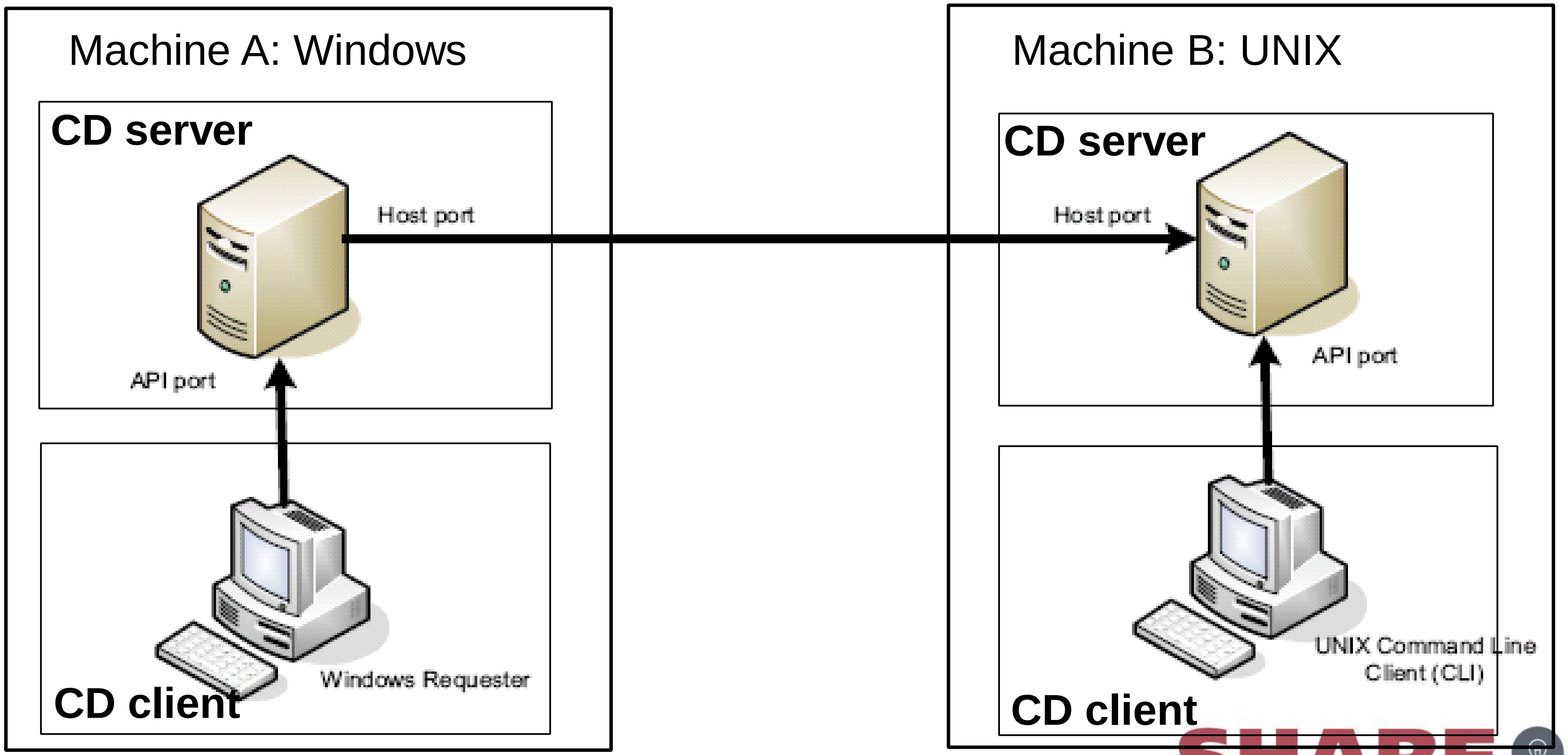
NOTES

- By default, after any read, the file read node will leave the file unchanged and will close any connections to it.
- It is possible to configure the node to modify the file disposition when the nodes *finish file* action is triggered. Finish file is defined as either when the file read node reads to the end of a file (whole file mode or when the last record is read) or when a message arrives at the *finish file* terminal.
- It is possible to delete or archive the file. Archiving moves the file to the mqsiarchive directory but it is possible to override this based on the contents of the message to move the file to any new directory and any new file name.

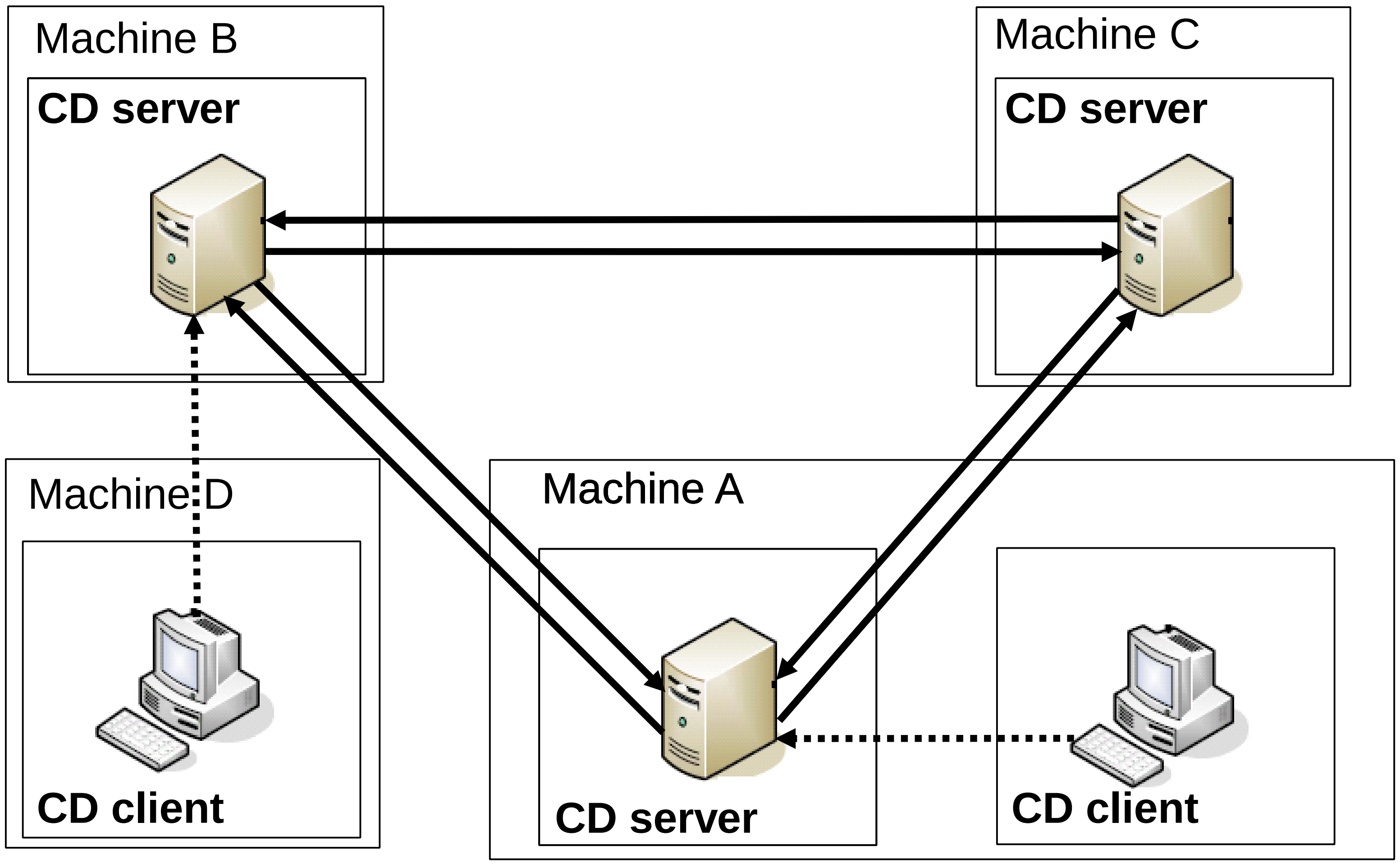
IBM Integration Bus: Connect:Direct nodes

IBM Sterling Connect:Direct (no IIB)

- IBM Sterling Connect:Direct is a managed file transfer solution

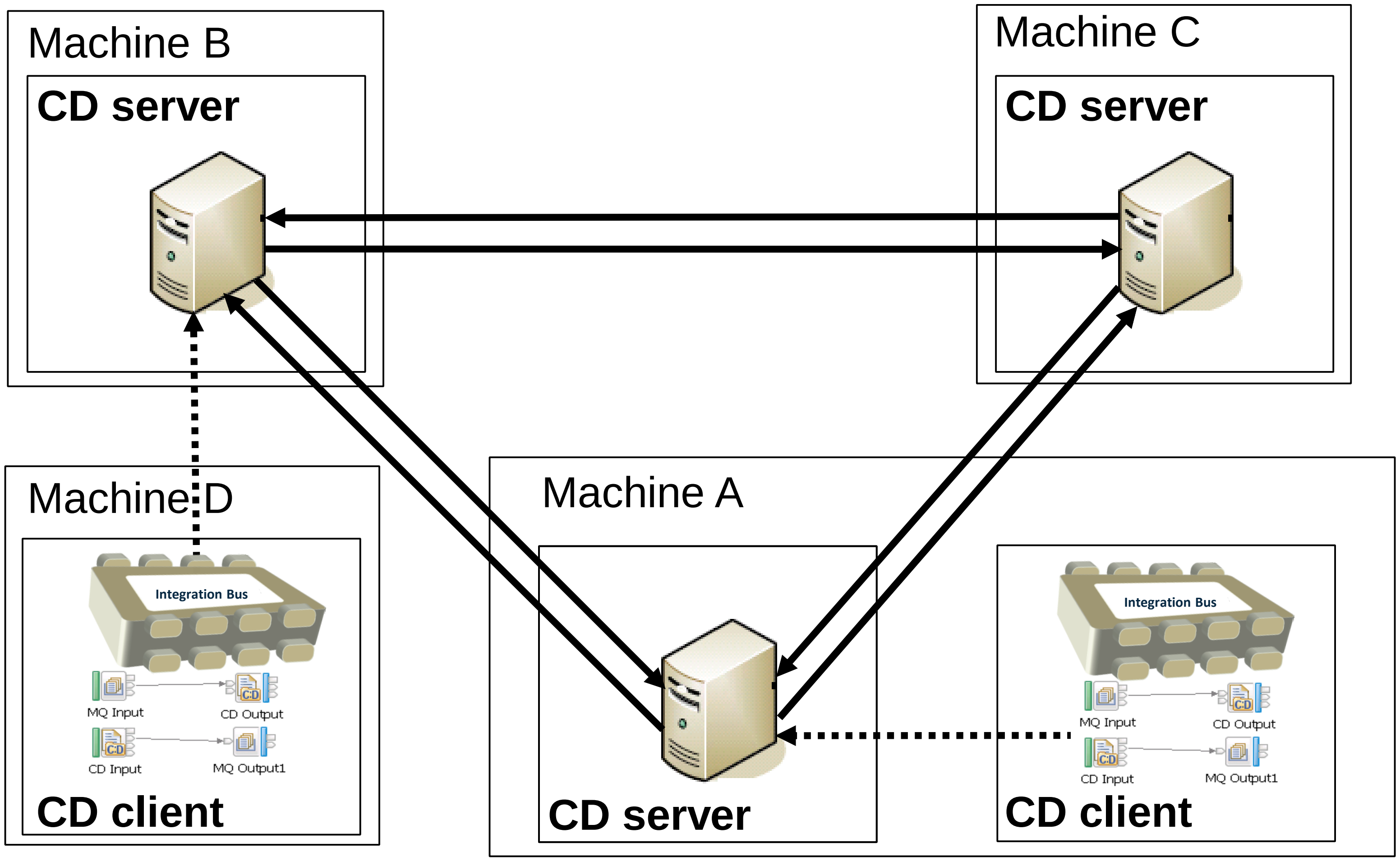


IBM Sterling Connect:Direct (no IIB)



Complete your session evaluations online at www.SHARE.org/Orlando-Eval

IBM Sterling Connect:Direct (with IIB)

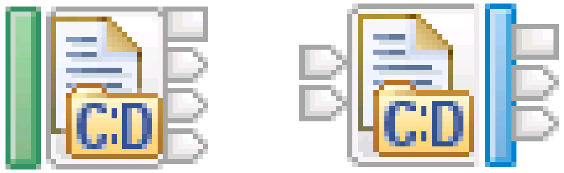


Adding IBM Integration Bus to a CD network

- Install Integration Bus
- Create and start an integration node on the machine
- Set up a security identity to be used to connect to CD server:

```
mqsisetdbparms BROKER –n cd::default –u gorman–p ********
```

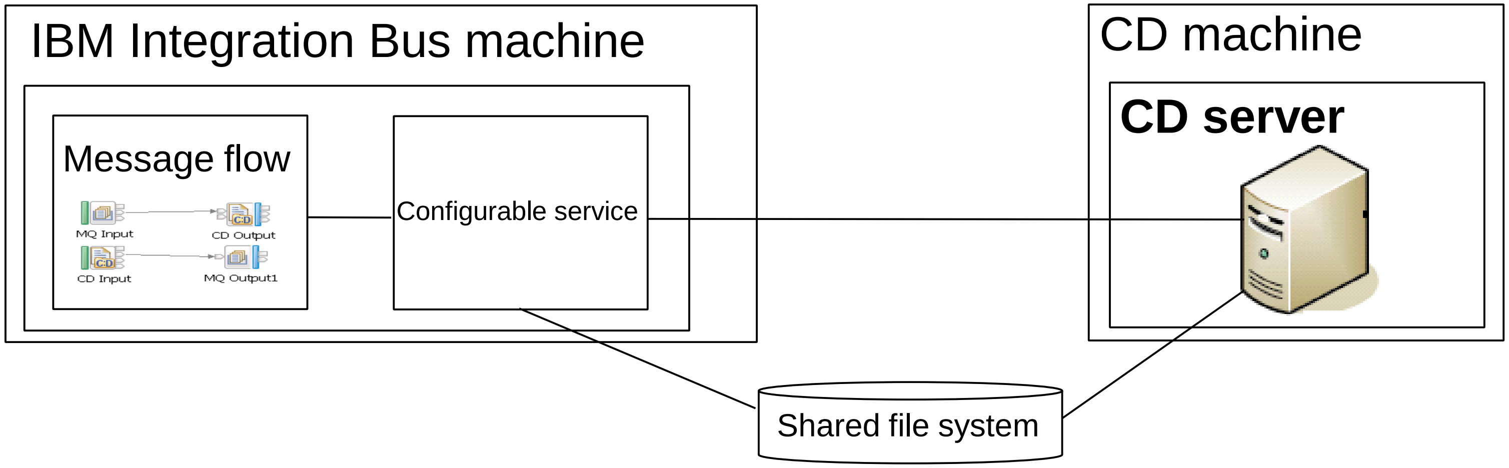
- Create and deploy flows which interact with CD via built in CD nodes



- That is all that is needed if the broker is on the same machine

Adding IBM Integration Bus to a CD network

- IBM Integration Bus does not have to be on the same machine as CD server.
- They must have access to a shared file system where files are transfer to.
- A configurable service can be created with details of CD Server:
  - Hostname
  - API port to connect to
  - Security identity
  - Filepath of the shared file systems

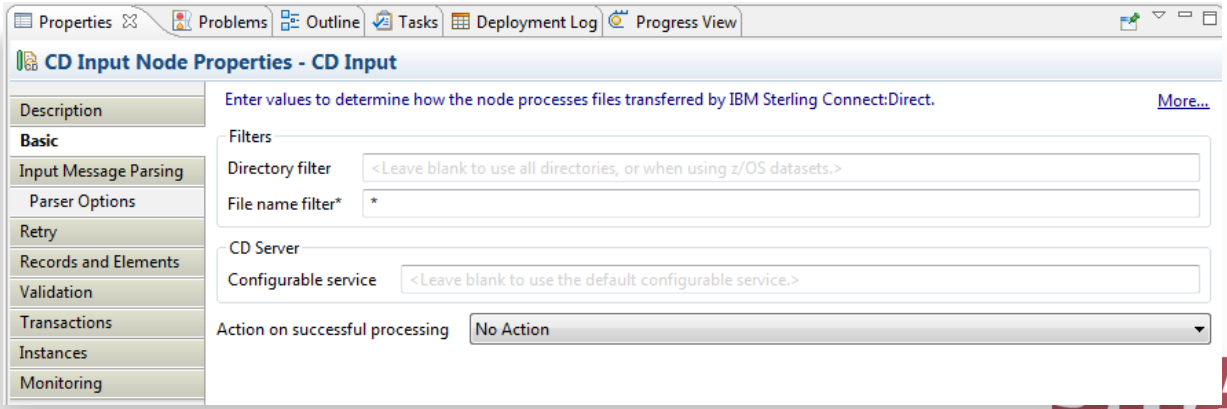


Accessing a CD server from a message flow

- Messages sent and received from the CD server using Integration Bus nodes:
  - CDInput node: receives any file transferred to the CD server
  - CDOutput node: constructs a file and sends a request to the CD server to transfer the file to a remote system.
- CD nodes based on the current file nodes allowing:
  - Record based processing
  - Stream parsing for large file support
  - Can be used with all other message flow nodes
- CD server is not stopped, started or administered by Integration Bus. Both products are decoupled:
  - Integration Bus outages have no effect on CD server
  - CD server outages only effect Integration Bus when a transfer is requested by a message flow

CD Input Node

- Consistent with file input node but makes full use of CD function
- Timely
  - The CD node monitors the CD servers stats for transfers that have completed
  - The node processes the files in the transfer immediately.
  - Each copied file is processed independently
  - Can leave the file unchanged after processing and just delete notification message
- Metadata
  - Metadata associated with the transfer is sent with the notification
  - Includes user defined application data field
- Filter
  - The node by default receives all transfers
  - Can specify which files to receive using a filter



CD Input Node Properties - CD Input

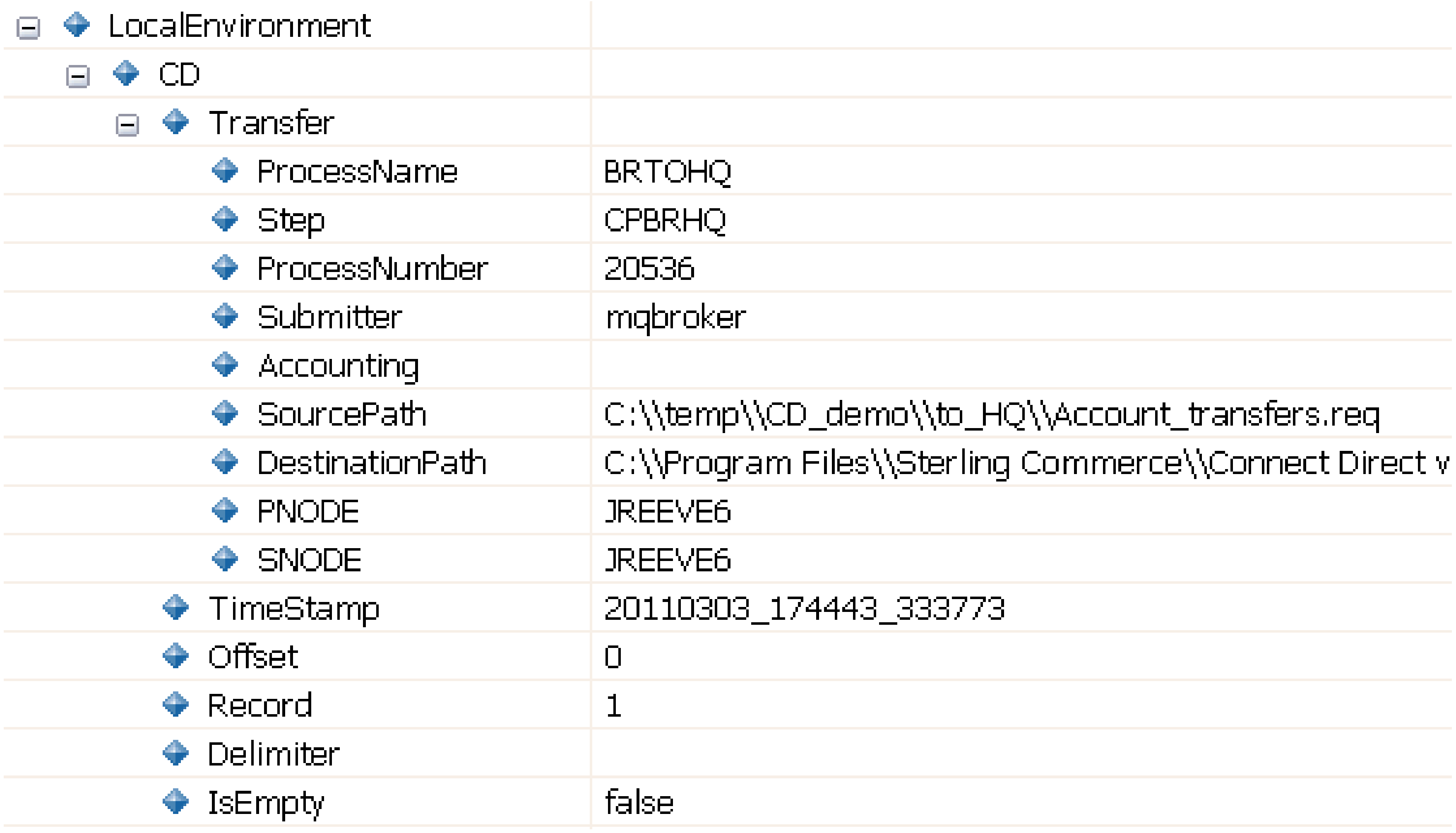
Enter values to determine how the node processes files transferred by IBM Sterling Connect:Direct.

Filters
Directory filter: <Leave blank to use all directories, or when using z/OS datasets.>
File name filter*: *

CD Server
Configurable service: <Leave blank to use the default configurable service.>

Action on successful processing: No Action

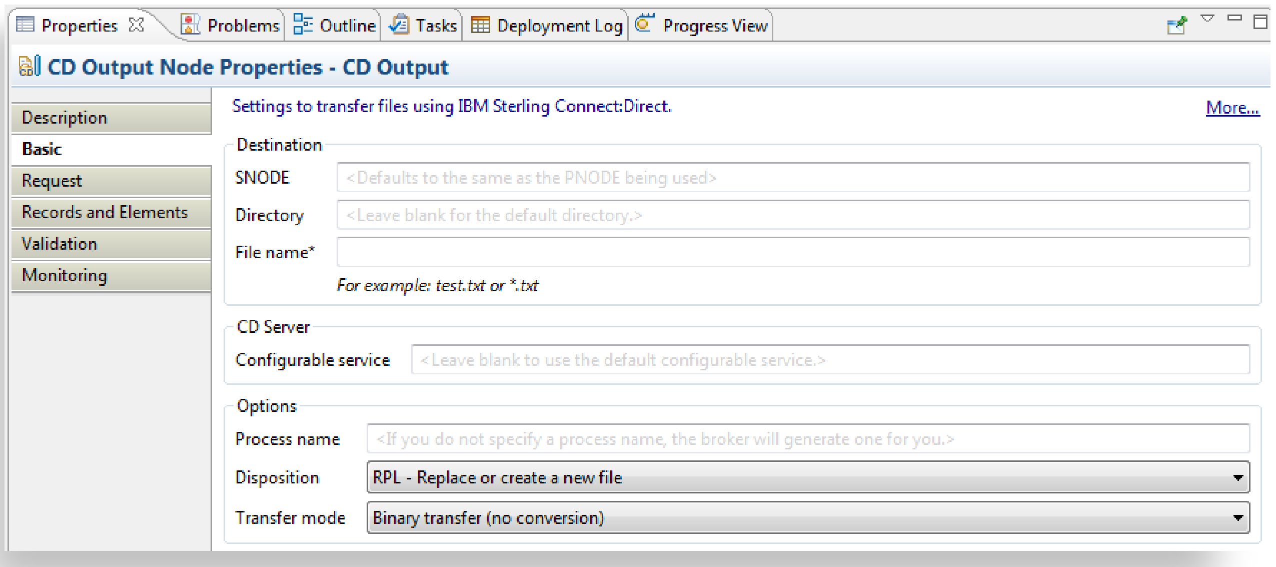
CD Input Node – metadata



LocalEnvironment	
CD	
Transfer	
ProcessName	BRTOHQ
Step	CPBRHQ
ProcessNumber	20536
Submitter	mqbroker
Accounting	
SourcePath	C:\\temp\\CD_demo\\to_HQ\\Account_transfers.req
DestinationPath	C:\\Program Files\\Sterling Commerce\\Connect Direct v
PNODE	JREEVE6
SNODE	JREEVE6
TimeStamp	20110303_174443_333773
Offset	0
Record	1
Delimiter	
IsEmpty	false

CD Output Node

- Consistent with file output node but makes full use of CD function
- Transfer details
  - The destination CD server, directory etc are defined on the node



CD Output Node Properties - CD Output

Settings to transfer files using IBM Sterling Connect:Direct.

Destination

SNODE	<Defaults to the same as the PNODE being used>
Directory	<Leave blank for the default directory.>
File name*	
	*For example: test.txt or *.txt*

CD Server

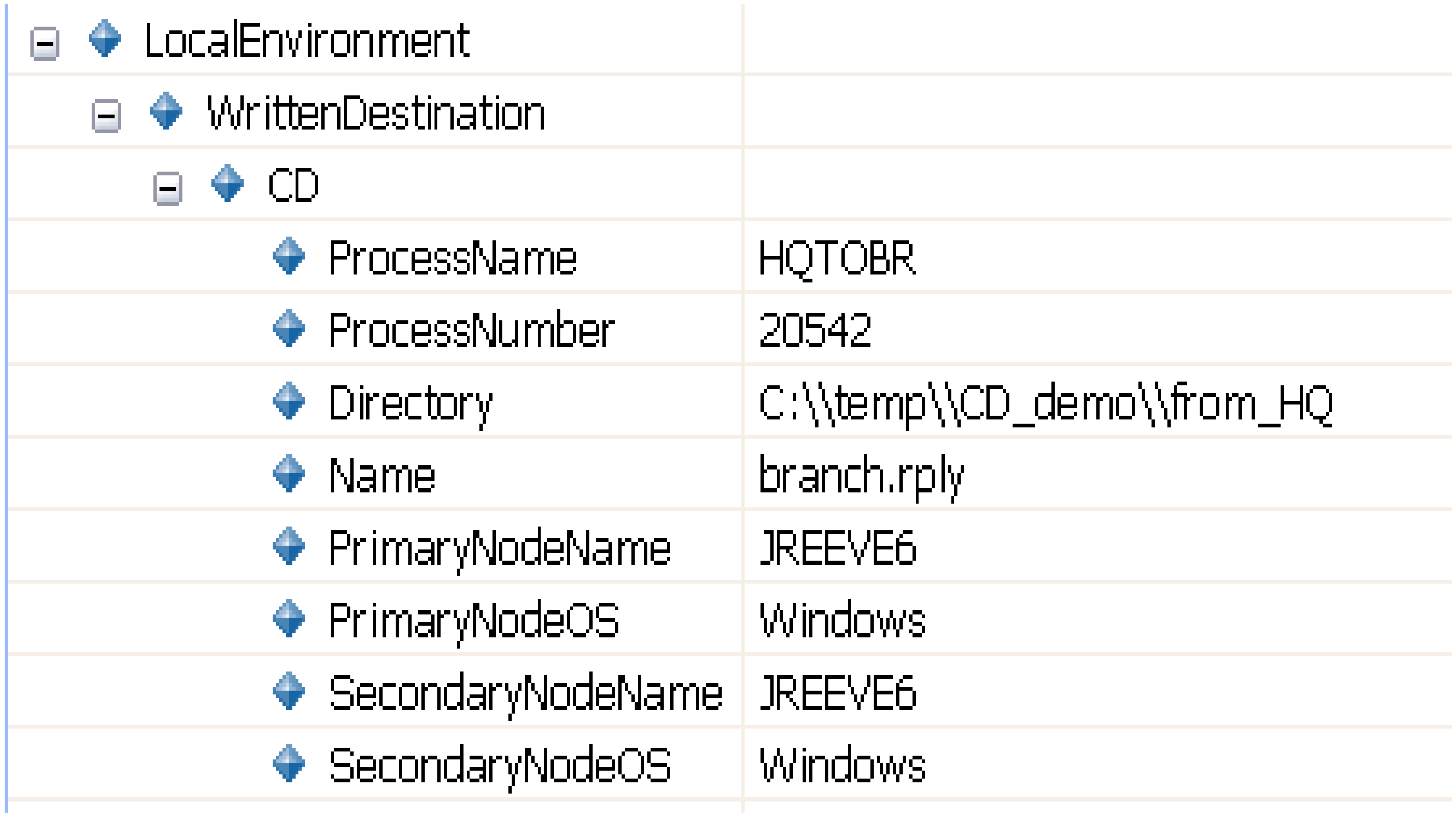
Configurable service	<Leave blank to use the default configurable service.>

Options

Process name	<If you do not specify a process name, the broker will generate one for you.>
Disposition	RPL - Replace or create a new file
Transfer mode	Binary transfer (no conversion)

  - Support also for wild card file names
- Staging
  - Uses a local staging directory to build up a file record by record for transfer
  - Once a file is finished a request is sent to the CD Server to transfer the file
- Metadata
  - User data if provided in the Local Environment is sent with the transfer
  - Other metadata is generated by CD server or by Broker

CD Output Node – Metadata



⊟ LocalEnvironment	
⊟ WrittenDestination	
⊟ CD	
ProcessName	HQTOBR
ProcessNumber	20542
Directory	C:\\temp\\CD_demo\\from_HQ
Name	branch.rply
PrimaryNodeName	JREEVE6
PrimaryNodeOS	Windows
SecondaryNodeName	JREEVE6
SecondaryNodeOS	Windows

File types supported

- Flat files on windows, UNIX and z/OS
  - Treated and processed the same as in the normal file nodes
- z/OS Sequential datasets
  - File name is the dataset name: JREEVE.TEST1.TEST2
  - Wildcard values allowed anywhere: JREEVE.*.*
  - Staged to HFS and then treated the same as in the normal file nodes
- z/OS Partitioned datasets
  - For the input node file name is a pattern like:
    - JREEVE.TEST1.TEST3(MEM1)
    - JREEVE.TEST1.TEST3(MEM*)
    - JREEVE.TEST1.TEST3(*)
    - JREEVE.*.TEST3(*)
  - Each member is staged to HFS and then treated the same as in the normal file nodes
  - For the output node the file name must be a single member:
    - JREEVE.TEST1.TEST3(MEM1)

Summary

- MQ MFT
- MQ MFT (FTE) nodes in IIB
- File nodes in IIB
- IBM Sterling Connect:Direct nodes in IIB

This was session #17891. The rest of the week…

	Monday	Tuesday	Wednesday	Thursday	Friday
08:30			17895: MQ for z/OS, Using and Abusing New Hardware and the New v8 Features	17891: Nobody Uses Files Any More Do They? New Technologies for Old Technology, File Processing in MQ MFT and IIB [z/OS & Distributed]	17898: Monitoring and Auditing MQ [z/OS & Distributed]
10:00	17885: Introduction to MQ - Can MQ Really Make My Life Easier? [z/OS & Distributed]	17901: MQ for z/OS: The Insider Story	17896: IBM Integration Bus MQ Flexibility [z/OS & Distributed]	17902: Common Problems and Problem Determination for MQ z/OS	17897: IBM MQ and IBM Integration Bus - from Migration and Maintenance to Continuous Enhancements, How and Why to Stay Current
11:15	17887: Introduction to IBM Integration Bus on z/OS [z/OS & Distributed]	17889: Introduction to the New MQ Appliance	16732: MQ V8 Hands- on Labs! MQ V8 with CICS and COBOL! MQ SMF Labs!		
01:45	17886: What's New in the Messaging Family - MQ v8 and More [z/OS & Distributed]		17899: Getting Started with Performance of MQ on z/OS	17890: IBM MQ: Are z/OS & Distributed Platforms Like Oil & Water?	
03:15	17888: What's New in IBM Integration Bus [z/OS & Distributed]	17903: Application Programming with MQ Verbs [z/OS & Distributed] 17821: SHARE Live!: End to End Security of My Queue Manager on z/OS	17900: Digging into the MQ SMF Data [z/OS]	17822: MQ Parallel Sysplex Exploitation, Getting the Best Availability from MQ on z/OS by Using Shared Queues [z/OS]	
04:30	17894: MQ Security: New v8 Features Deep Dive [z/OS & Distributed]	17893: The Do's and Don'ts of IBM Integration Bus Performance [z/OS & Distributed]	17892: Giving It the Beans: Using IBM MQ as the Messaging Provider for JEE Applications in IBM WebSphere Application Server [z/OS & Distributed]	17884: Challenge the MQ & IIB Experts? [z/OS & Distributed]	

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

SHARE in Orlando 2015