WebSphere® MQ Integrator Enabler

Release Notes

Fifth Edition (June 2002)

Printed in USA.

Introduction

The WMQI Enabler provides an enterprise-wide scalable framework that allows multiple front-end applications to inter-operate with back-end applications with its messaging hub architecture.

Features

- Message Header support: WMQI Enabler header within IAA-XML messages for message management and distribution.
- Cross Reference Function (CRF) allowing WMQI Enabler to maintain keys, plus a tool to allow an initial load to the CRF.
- Use of a DB2 database (similar to a LightWeight Directory Access Protocol) to maintain logical symbolic application names and physical location of the message routing, plus a tool is provided to maintain the directory.
- Ability to change or add participating applications and routing messages without disturbing either front-end or back-end applications.
- Logging abilities: Business Event log for logging messages processed by the WMQI Enabler hub and Error conditions.
- Message management for Fire and Forget and Request/Reply message types.
- Support for long running transactions via process management (MQSeries Workflow).
- Provides model office (Use Cases) sample sets as a starting point for Global Services implementation.
- XML message support for messages to be processed by WMQI Enabler only, or with/without interfacing to MQSeries Workflow.
- Session validation that allows interactive sessions to Logon and Logoff as well as to perform other WMQI Enabler only commands.
- Sequence validation that orders messages in the same session as they are processed through WMQI Enabler.
- Tracing ability within WMQI Enabler flows for user error analysis.

- Symbolic Destination Resolution (SDR) ability to include backup lists of systems to route messages to if the primary destination is inactive.
- Publish and Subscribe capability.
- HUBONLY Message Configuration Utility.
- Sample LDAP as SDR.
- Enhanced Authentication for HUBONLY messages.
- Performance improvements.
- Complex Processes support.
- Improved documentation including Installation and Setup Guide and Industry Reference Manuals.
- NLS error message support.

The following table highlights some of the features found in WMQI Enabler along with the benefits of those features:

Feature	**Benefit**
Publish and Subscribe capability.	Messages, by message type and topic, can be configured for WMQI Publish/Subscription.
HUBONLY Message Configuration Utility.	A GUI tool to manage HUBONLY messages is available for system administrators.
Sample LDAP as SDR.	An optional SDR using LDAP feature is included. This LDAP feature makes use of LDAP plug-in nodes. The availability and support of the LDAP plug-in is not a part of WMQI Enabler. LDAP support capability is provided as is and limited to Windows NT systems only.
Enhanced Authentication for HUBONLY messages.	All message types defined to WMQI Enabler now include an additional administration enable/disable feature. System administrators can enable/disable individual message types, as required. This feature allows more complete administrator control of the WMQI Enabler run time environment.

Feature (Continued)	**Benefit**
Performance improvements.	Subflows, including CRF, and the underlying WMQI Enabler database structure have been redesigned and re-implemented to improve runtime performance.
Supports the latest releases of the MQSeries family of products.	WMQI Enabler is supported on the following Operating Systems and MQ stack applications: • Windows NT V4.0 with Service Pack 6a or Windows 2000 Service Pack 2 • MQSeries V5.2 for Windows • WebSphere MQ Integrator V2.1 for Windows • MQSeries WorkFlow 3.3.2 for Windows • DB2 UDB V7.2 • AIX V4.3.3 • MQSeries V5.2 for Unix (AIX or Solaris) • WebSphere MQ Integrator V2.1 for Unix (AIX or Solaris) • MQSeries WorkFlow 3.3.2 for Unix (AIX or Solaris) • DB2 UDB V7.2 for Unix (AIX or Solaris)
NLS error message support.	Error messages reported by WMQI Enabler can now be configured in non-English languages.
WMQI Enabler supports the ability to dynamically define container structures for passing data between WMQI and MQSWF.	This optional feature allows the user to specify customized data to be passed from WMQI to MQSWF. This includes the ability for MQSWF to send data changes which WMQI will use to update the message content

Feature (Continued)	**Benefit**
WMQI Enabler lets you name versions of messages within MQSWF so that they can be stored and reused within transactional message processing. By combining this feature with the dynamic container feature, WMQI Enabler now provides support for multiple message types within a message flow.	This optional feature allows MQSeries Workflow to support using more than one message type/format within the process template, and allows existing MQSWF API compliant applications to have access to message data.
WMQI Enabler supports hub based Logon and Logoff messages that allow WMQI Enabler to generate a session token that can be used for simple session/state handling for validating message affinities.	This option gives WMQI Enabler the ability to validate that a message request is being made by someone that has previously been granted access to the enterprise. WMQI Enabler uses a table approach to provide authorization/authentication, but can be customized by an engagement to support an existing user security system.
WMQI Enabler supports a message store and forward option that allows messages to be re-processed when unavailable applications become available.	This option allows WMQI Enabler to decide, on a message-by-message basis, what to do with undeliverable messages. The options are: “kill the message” and send a system unavailable response; “store the message” so that it can be reprocessed when the required system(s) are available. The determination is made based on the content of the system profile table about the source system from which the message originated.
WMQI Enabler provides support for both a lightweight WMQI based message-processing supervisor and a transactional MQSWF message-processing supervisor.	This optional feature allows for determining, on a message-by-message basis, whether MQSWF is used to supervise the receipt of a response message from an adapter, or if WMQI directly delivers the message and relies on the adapter to respond.

Feature (Continued)	**Benefit**
WMQI Enabler provides a profile that allows applications to signal their availability; as well as, the definition of mirror and alternate applications to use as the target for message requests.	This optional feature allows WMQI Enabler to determine if systems required to process a specific message are available. If not, WMQI Enabler can determine if there are alternative destinations available on a message-by-message basis. Included in this is the ability for a system to signal that it wants to change its availability. For a system requesting a shutdown, WMQI Enabler will signal the requesting application when it is OK for that system to shut down based on when in-process messages have completed. WMQI Enabler will then divert new messages to alternate destinations. For systems that are requesting start up, WMQI Enabler will restart any messages that were stored by the store and forward function.
WMQI Enabler provides message maintenance of internal tables	WMQI Enabler provides message support for maintenance of internal tables such as the message profile, system profile, cross reference file, symbolic destination, and other tables.
WMQI Enabler now tracks and records the status of messages as they are processed through the hub. WMQI Enabler internally traces activities including process template completion to provide comprehensive tracing and problem determination support.*	This feature provides comprehensive tracing across the MQ products integrated within WMQI Enabler.

Operational characteristics

HUB commands

A set of HUB_ONLY_ONLINE and HUB_ONLY_OFFLINE XML message commands are supported. The commands include requests for internal database maintenance, Logon and Logoff, System startup notify or shutdown requests, update SDR profile, get and update system profile, get or update NLS error messages, get or update installation profile data, set topic subscription, kill session requests, kill process requests, and get or update message profile requests.

Message routing interface

WMQI Enabler XML messages can be structured to request interface support with just the Hub (HUB_ONLY_ONLINE and HUB_ONLY_OFFLINE), with WMQI Enabler and user applications, or WMQI Enabler with MQ Workflow and user applications.

Sequence validation

Sequence validation supports the ordering of messages in the same session through the WMQI Enabler product. Each message type can use the MessageTypeDependency flag in the Message Profile to indicate a message type must complete successfully in the same session before the current message type can begin processing. The sequence of each message can be validated one at a time to govern a list of message types.

Interaction check

Interaction check support consists of System Interaction and System Profile databases that contain a list of valid front-end and back-end systems to interact with and identifies whether these systems are available, respectively. With these indicators, the WMQI Enabler product can check whether active requests to a system can be made (whether a system is up, going inactive, or is inactive).

Symbolic resolution destination

SDR provides the System Interaction database that contains a list of symbolic names to identify systems, their Queue Managers, and the queues used to communicate with them. In addition, the list supports a backup list for systems that can be used to receive a message if the “primary” system is unavailable (assuming the backup system can successfully receive/process the message).

An optional SDR using LDAP feature is included. This LDAP feature makes use of LDAP plug-in nodes. The LDAP flows are provided as is and their support capability is limited to Windows NT systems only.

Session validation

WMQI Enabler supports the Logon command to establish a session ID to be used on subsequent messages. Session validation checks whether the session exists and hasn't timed out. If timed out, a Logon can be requested again using the same authorization information to re-establish a session. Session validation has been extended to the HUBONLY command set.

Message Type Administration

All message types defined to WMQI Enabler now include an additional administration enable/disable feature. System administrators can enable/disable individual message types, as required, on the fly. This feature allows more complete administrator control of the WMQI Enabler run time environment.

Database

The WMQI Enabler database structure is designed to improve application performance.

Logically, the new database structure is separate databases. Each database has an alias. Actual implementations may organize their databases' logical-to-physical configuration to fit their specific environment.

The installed implementation configures the logical databases to a single physical database.

The database tables included are:

1. Build_Workflow_Parameters_Table
2. Install_Data_Table
3. NLS_Error_Message_Table
4. Message_Log_Table

5. Message_Profile_Table
6. CRF_Table
7. Process_State_Table
8. SDR_Table
9. Session_Authentication_Table
10. System_Backup_Table
11. Workflow_Correl_Table
12. Trace_Table
13. Workflow_Parameters_Table
14. System_Interaction_Table
15. System_Status_Table
16. System_Store_Flag_Table
17. Session_Table
18. Session_Processes _and_System _Usage_Table
19. Stored_Message_Table
20. Interaction_Dependency_Table
21. Error_Table
22. Exception_Table
23. Message_Table
24. Original_Message_Table

These tables and thier respective aliases may be found in the *I**nstallation and SetUp Guide:** Appendix B.*

WMQI Queues

The following queues have been developed for use by WMQI. As you will note, the queue names have been structured to match the primary WMQI flows as a naming convention. When an error condition is encountered in one of the primary flows, the name, "Failure" is added to the queue name and is the queue upon which the error condition may be retrieved.

1. HUB_IN
2. HUB_IN_FAILURE
3. MQWF_OUT
4. MQWF_OUT_FAILURE
5. HUB_RWF_IN
6. HUB_RWF_IN_FAILURE

7. HUB_R_IN
8. HUB_R_IN_FAILURE
9. MQWF_END
10. MQWF_END_FAILURE
11. HUB_ONLY_ONLINE
12. HUB_ONLY_ONLINE_FAILURE
13. HUB_ONLY_OFFLINE
14. HUB_ONLY_OFFLINE_FAILURE
15. LOGIN
16. TEMPLATE_IN
17. ERROR_BUCKET

Additional queues are required to support sample test cases:

1. FEIN
2. BEIN
3. BEIN2
4. CBPINPUT