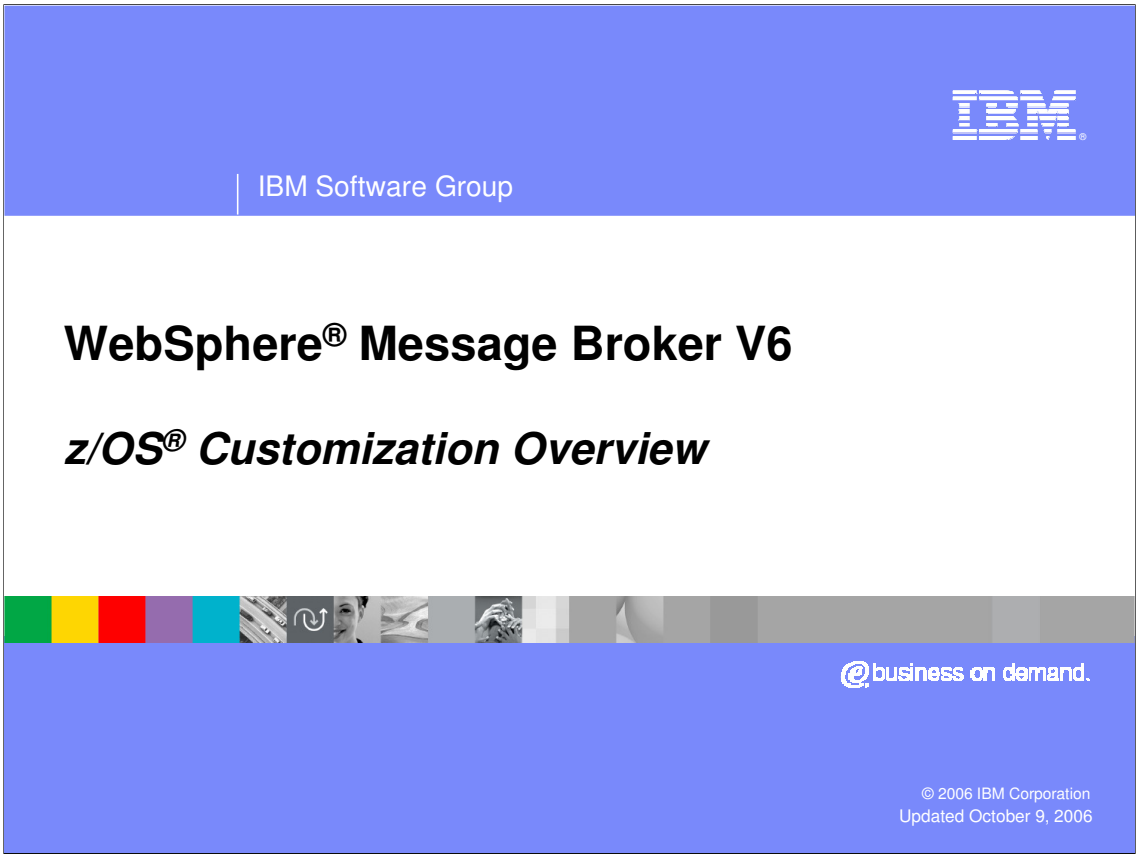


This presentation provides an overview of the WebSphere Message Broker V6 customization process.

Agenda

- V5 customization and V6 customization compared
- V6 customization steps
- Summary and references

This presentation reviews the z/OS V5 customization steps and provides information on the V6 customization.

Section

V5 customization and V6 customization

This section compares the customization process for WBIMB V5 and WMB V6

Overview

- What is customization?
  - Process of creating and tailoring broker components to produce runtime components
    - Brokers
    - User Name Servers
    - Configuration Managers (V6 only)
  - Output from the customization is the same for V5 and V6
    - Registry
    - MQ resource definitions
    - Database resource definitions

After the SMPE installation of a broker product on z/OS, the broker components must be created and customized. Customization is the process of tailoring a broker component to produce a runtime component, whether a broker, Configuration Manager (V6 only) or a User Name Server.

The output of a V6 customization is the same as in V5, including the registry, WebSphere MQ resource definitions, and Database resource definitions.

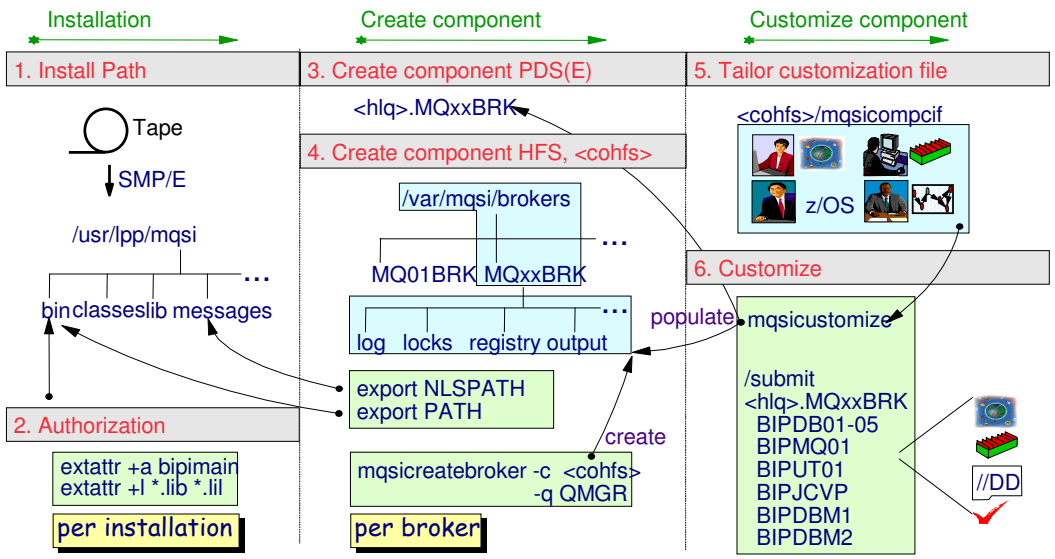
V5 customization environment

- Used UNIX® System Services (USS) commands submitted under TSO environment
- Automatically generated JCL to create the component
- mqsicompcif was the master source of customization
- mqsicustomize command would update the environment file
- Verification program did not automatically run under the broker user ID

In earlier versions of message broker, the z/OS customization involved running USS commands under the TSO environment. The master source for customization, mqsicompcif, resided in the broker HFS. The mqsicustomize command updated the customization JCL and the runtime ENVFILE based on data in mqsicompcif. The ENVFILE could be changed independently of the mqsicompcif; however, those changes would be overwritten if the mqsicustomize command was run again and the changes made to the ENVFILE were not reflected in mqsicompcif.

The verification program was invoked separately from running the broker, so user IDs were not the same.

Overview of V5 customization



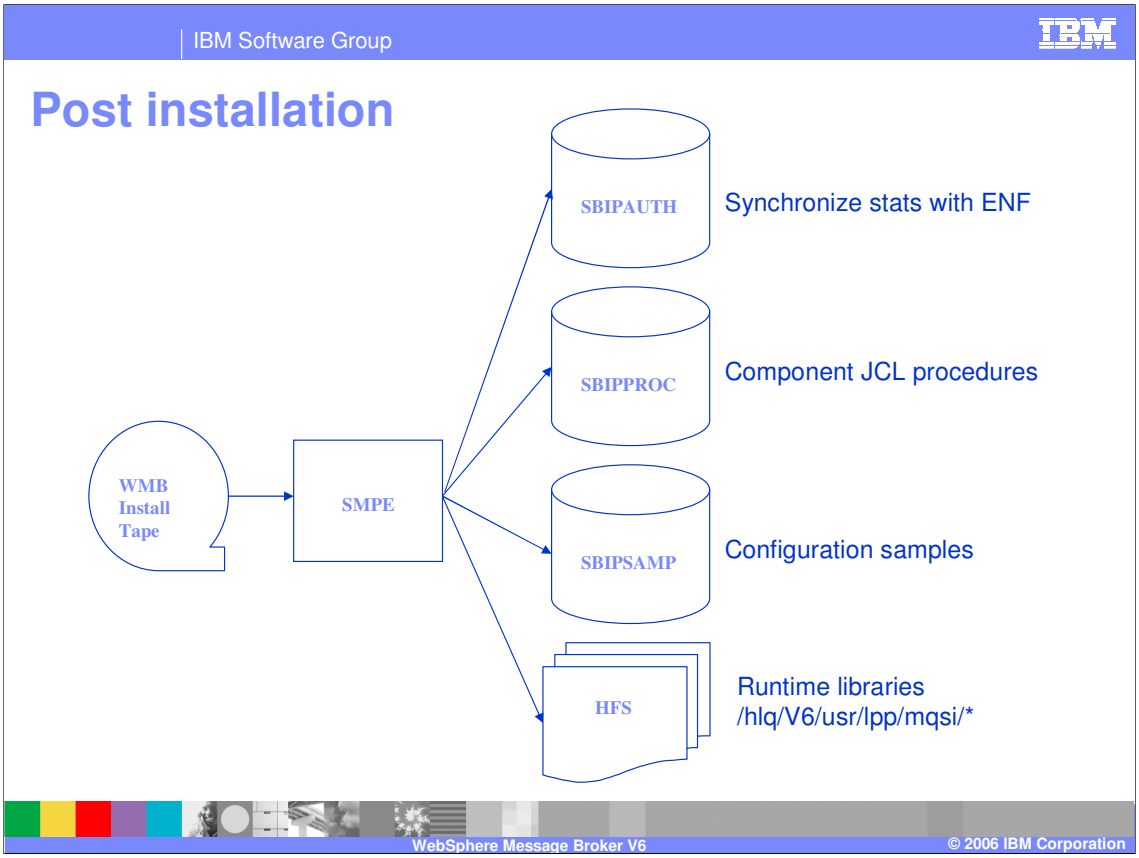
This slide provides an overview of the V5 installation, configuration and customization process.

The broker code is the same base as distributed and is written in C. It resides in the UNIX System Services (USS) on z/OS. In V5 the mqsicreatebroker command is issued from the pre-allocated component HFS and creates the file structure in the HFS directory. After tailoring the mqsicompcif file with path names, queue manager name, database names, and so on, the mqsicustomize command is run to populate the component PDS with JCL to build the component.

Objectives for V6 customization

- Provide a process natural to z/OS users
  - Simple
  - JCL in place of shell scripts
  - Easily tailored
  - Similar to other z/OS products
    - MQ, CICS®, IMS …
  - Provides for possible organizational issues
    - Systems, MQ, DBA…
- Support all z/OS components
  - Broker
  - Configuration Manager
  - User Name Server

Unlike previous broker versions, the V6 customization is by JCL; the USS commands are run from within the batch jobs. This process supports all components on z/OS: brokers, Configuration Managers and User Name Servers.



After the SMP/E installation of WMB V6, there are three PDS files in addition to the HFS which holds the product runtime libraries.

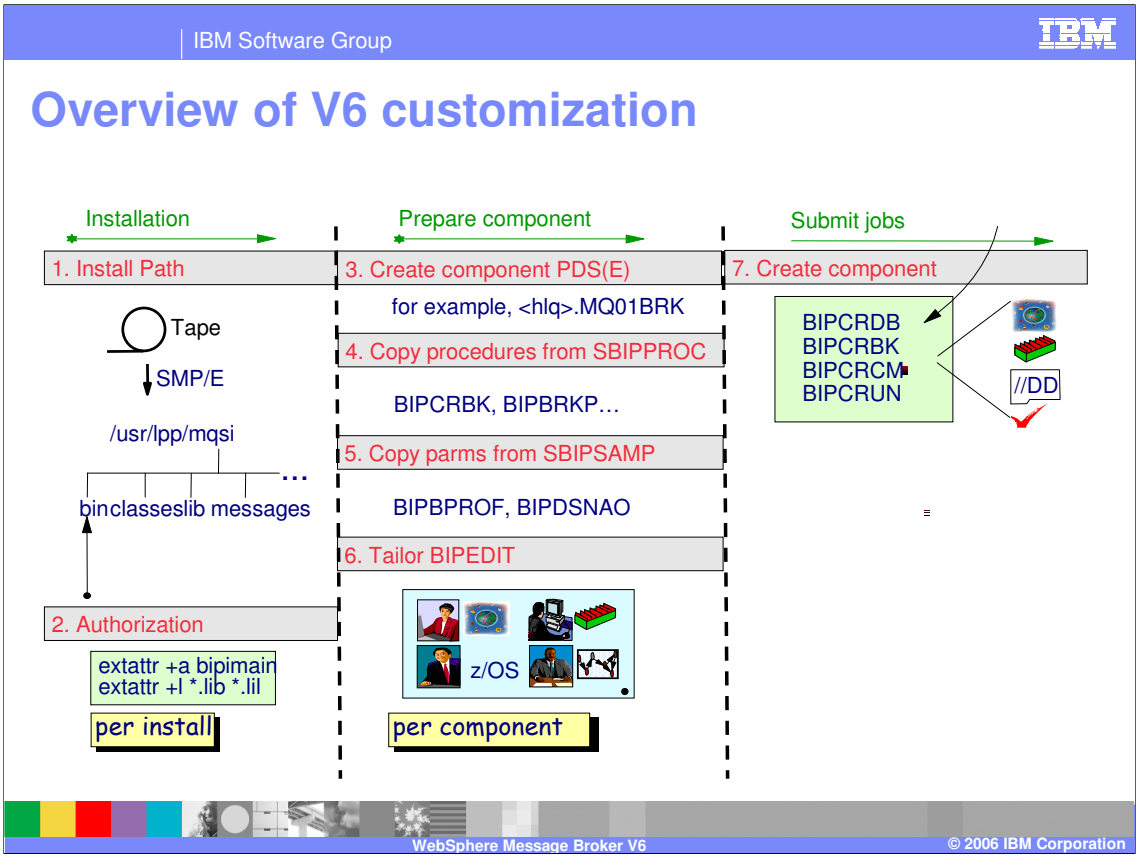
hlq.SBIPROC contains the JCL procedures; hlq.SBIPSAMP has configuration sample jobs. hlq.SBIPAUTH is for ENF, event notification facility. This synchronizes the broker with other products (for example, WebSphere MQ) when writing SMF records.

Post installation

- Component JCL procedures (hlq.SBIPPROC)
  - Broker, Configuration Manager and UNS JCL procedures
    - BIPBRKP, BIPCMGRP, BIPUNSP
  - Broker, Configuration Manager and UNS JCL to create components
    - BIPCRDB, BIPCRBK, BIPCRCM, BIPCRUN
  - Broker, Configuration Manager and UNS JCL commands
    - BIPBRWS, BIPCHBK, BIPCHMS…
      - mqsibrowse, mqsichangebroker, mqsichangeflowstats..
  - ALL require customization
- Configuration Samples (hlq.SBIPSAMP)
  - Sample broker, Configuration Manager and UNS profiles
    - BIPBPROF, BIPCPROF, BIPUPROF
  - Sample ODBC parameters (BIPDSNAO member)
  - Sample RUNSTATS job
  - ALL require customization
- ALL steps are JCL and all use regular MVS datasets

The component JCL procedures dataset (hlq.SBIPPROC) provides jobs that issue broker commands, including the commands to create the components and the started task procedures. All of these jobs require customization for each specific component.

The configuration samples dataset (hlq.SBIPSAMP) provides the parameter information and jobs needed to customize a component. These also require customization for each specific component.



In V6 customization, a separate component PDS must be created for each broker, Configuration Manager, and User Name Server. It is suggested that you use the queue manager name followed by the component identifier for the component name, started task procedure name, and PDS name. For example, use hlq.MQ01BRK for the component PDS when MQ01 is the queue manager associated with the broker component; and use MQ01BRK for the started task procedure name.

Populate the component PDS with JCL and parameters from the hlq.SBIPPROC and hlq.SBIPSAMP datasets from the product install.

After tailoring the component PDS, the jobs to create the component are available to run.

Section

V6 customization steps

This is an overview of the steps required to accomplish the customization.

Customization preparation

- **Allocate Broker, Configuration Manager or UNS PDS(E)**
  - Copy required component procedures and JCL from SBIPPROC
- **Customize component dataset**
  - Copy profiles from SBIPSAMP
    - BIPBPROF - Broker profile
    - BIPCPROF - Configuration Manager profile
    - BIPUPROF - UNS profile
  - Other parameter files from SBIPPROC
    - BIPEDIT – JCL edit file
    - BIPDSNAO – ODBC parameters for DSNAOINI
- **Tailor BIPEDIT JCL edit procedure**
  - Assign settings for registry, DB2®, Java™ and so on

Allocate and populate your component PDS from the sample dataset, SBIPPROC, choosing the members that are relevant to the particular component you are creating (broker, Configuration Manager, User Name Server).

Copy the profile dataset for your component (broker, Configuration Manager, User Name Server) from SBIPSAMP.

ODBC parameters must be provided for the broker databases; sample BIPDSNAO is provided in SBIPSAMP dataset.

You must provide environmental information such as file location for additional product prerequisites, for example DB2, Java, and so on. Other JCL can be updated by using BIPEDIT after you have tailored it with your installation specifics.

Job submission

- Target appropriate jobs with BIPEDIT
  - BIPCRDB, BIPCRBK, BIPCRCM, BIPCRUN – Create Broker Database, Create Broker, Configuration Manager, UNS
  - Optionally modify options according to authority
    - Default is all resources
      - -1 only create the registry in USS. (USS Administrator).
      - -2 only do the MQ objects pass (MQ Administrator).
      - -3 only do the DB2 objects pass (DB2 Administrator).
- Run BIPGEN to create ENVFILE
- Create component
  - For a broker submit BIPCRDB to created DB2 storage groups, database and table space definitions.
  - Submit BIPCRxx to create component
- Start component
  - for example, /S MQ01BRK
- Component verification automatically runs as the first step.

BIPEDIT REXX exec is provided as a means to modify the jobs in your component PDS with your customization parameters.

BIPGEN job is used to create the ENVFILE which is accessed at run time.

After each job has been tailored for your component, you run the jobs to create the component.

Once you have set up the started procedure for your component, you can start the component from the system console (in this case the broker component associated with queue manager MQ01 is MQ01BRK).

Component verification is run as the first step of the component job.

Section

Summary and references

This section contains a summary and references.

Summary

- V5 customization and V6 customization compared
- V6 customization steps

This presentation has reviewed how the V5 broker was customized on z/OS; and how and why V6 customization is different.

References

- WebSphere Message Broker library:

http://www-306.ibm.com/software/integration/wbimessagebroker/library/

- WebSphere Message Broker Information Center:

http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.