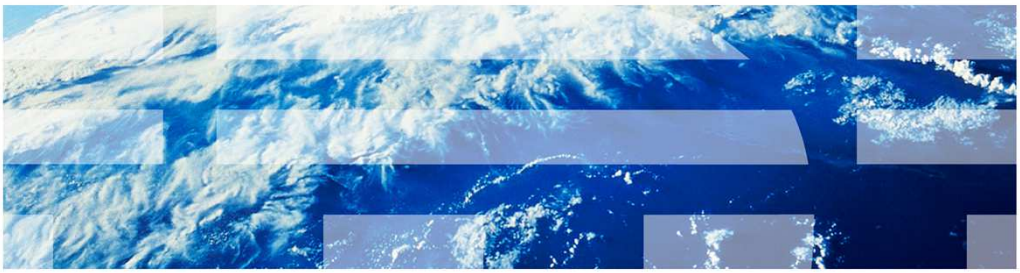
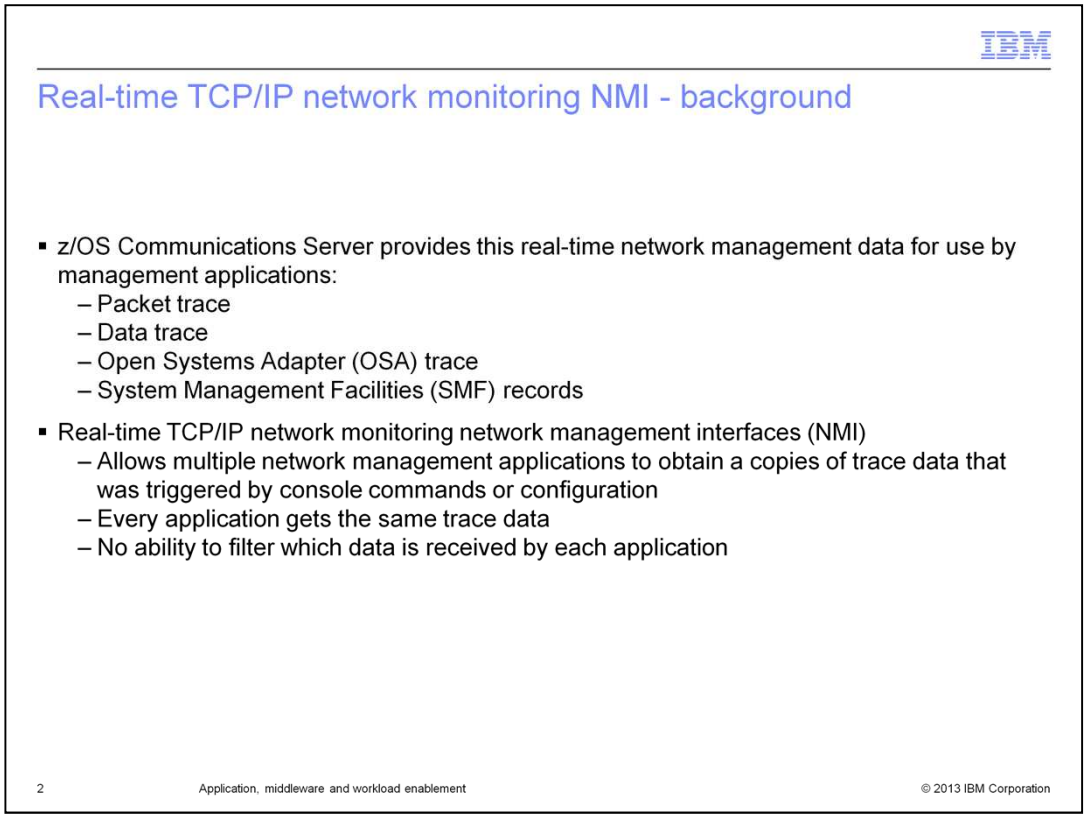
z/OS V2R1 Communications Server

Application, middleware, and workload enablement



This presentation provides an overview of the z/OS® V2R1 Communications Server enhancements for application, middleware, and workload enablement.



z/OS Communications Server provides several network management interfaces, including the real-time TCP/IP network monitoring network management interface. Applications can obtain real-time packet trace, data trace, Open Systems Adapter trace, and System Management Facilities records from this network management interface. Applications can use the real-time network monitoring network management interface to obtain this data. However, the applications cannot specify filters for this data.

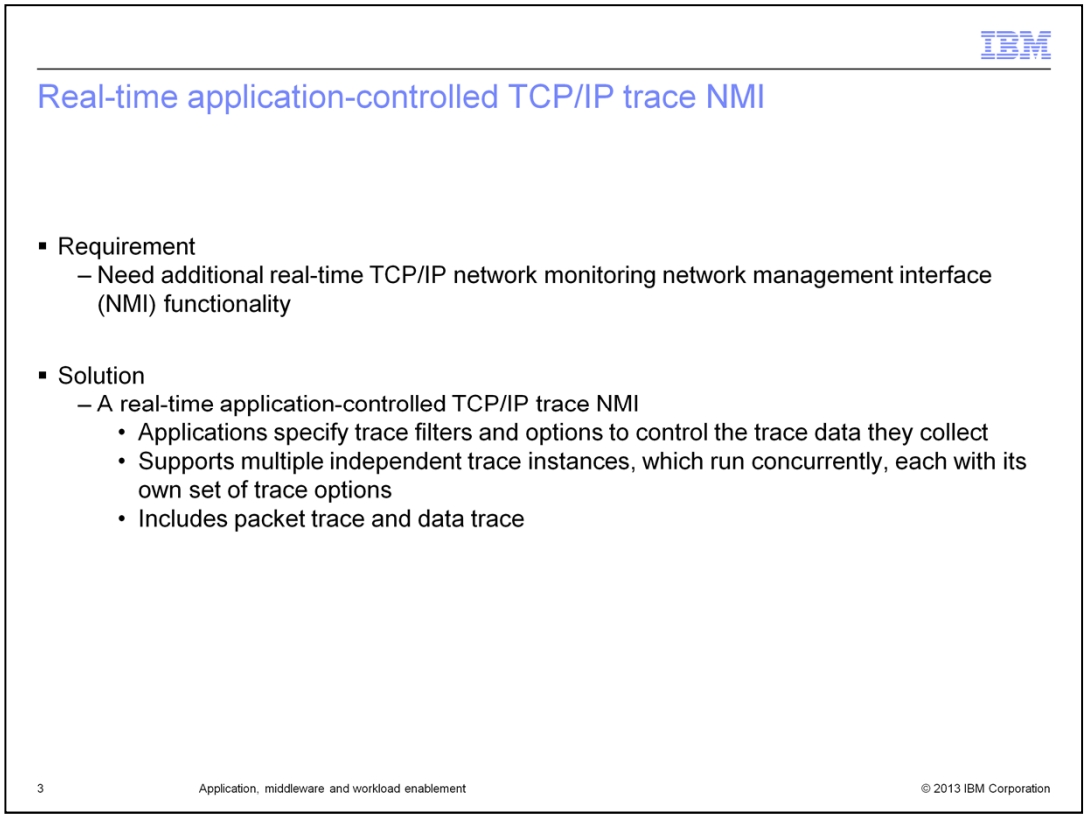
The real-time data that the applications receive depends on several global settings for the stack:

First, packet trace that is set by configuring the PKTTRACE profile statement or by issuing the V TCPIP,,PKTTRACE command.

Second, data trace that is set by issuing the V TCPIP,,DATTRACE command.

Third, Open Systems Adapter trace that is set by configuring the OSAENTA profile statement or by issuing the V TCPIP,,OSAENTA command.

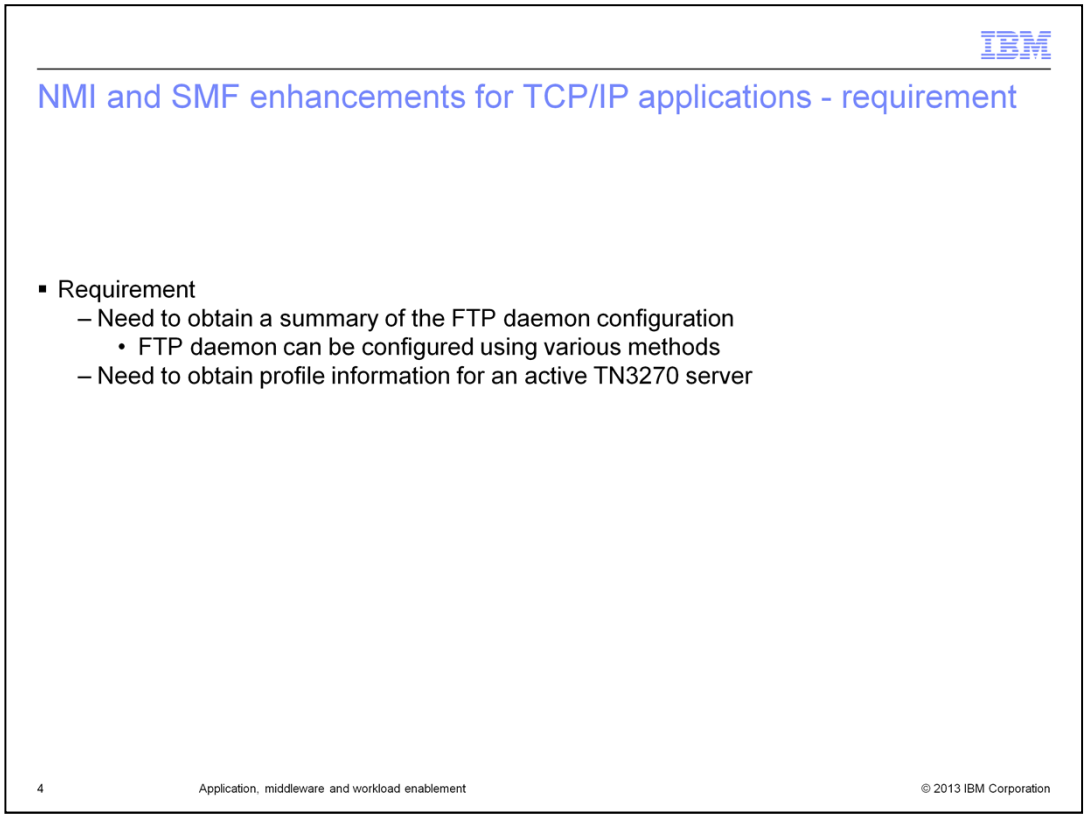
And finally, System Management Facilities records that are set by configuring the SMFCONFIG and NETMONITOR SMFSERVICE profile statements.



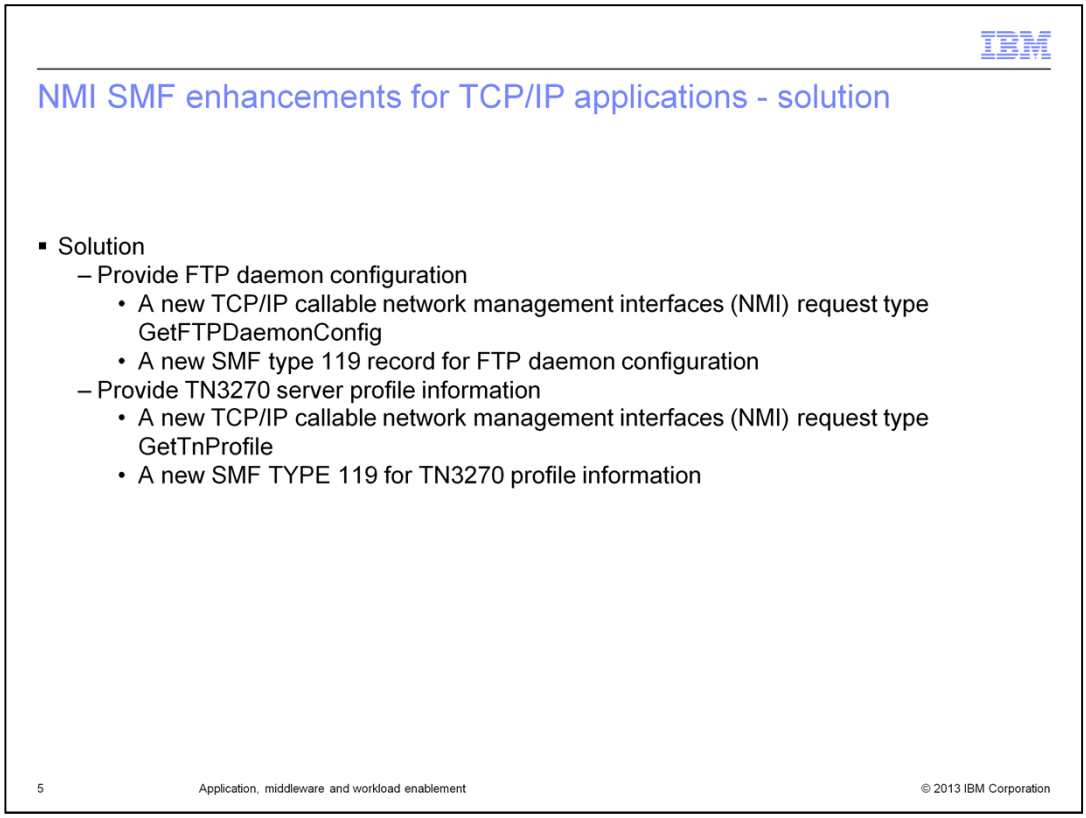
Network management application vendors need additional real-time network management interface functionality. They need to be able to set their own filters for obtaining real-time data and they do not want their filter values to be modified. Vendors need z/OS Communications Server to implement a record-based solution for providing trace data to collector applications, instead of a buffer-based solution. Venders also need to be notified if trace records are lost. And they need to be able to receive all of the different types of real-time data trace records interleaved in one data stream.

Starting in V2R1, Communications Server provides a new callable network management interface called real-time application-controlled TCP/IP trace network management interface. Applications can use this network management interface to obtain real-time TCP/IP stack data. This new network management interface differs from the real-time TCP/IP network monitoring network management interface in the following ways. First, it allows network management applications to specify trace filters and options that are independent of those specified for the TCP/IP stack. Second, the application does not have to wait to receive a buffer token before invoking a request to obtain the trace records. And finally, different types of trace records are included in one, single trace data stream, instead of a separate stream for each type.

Packet trace records can optionally include Internet Protocol Security clear text data. And data trace records can optionally include Application Transparent Transport Layer Security clear text data.

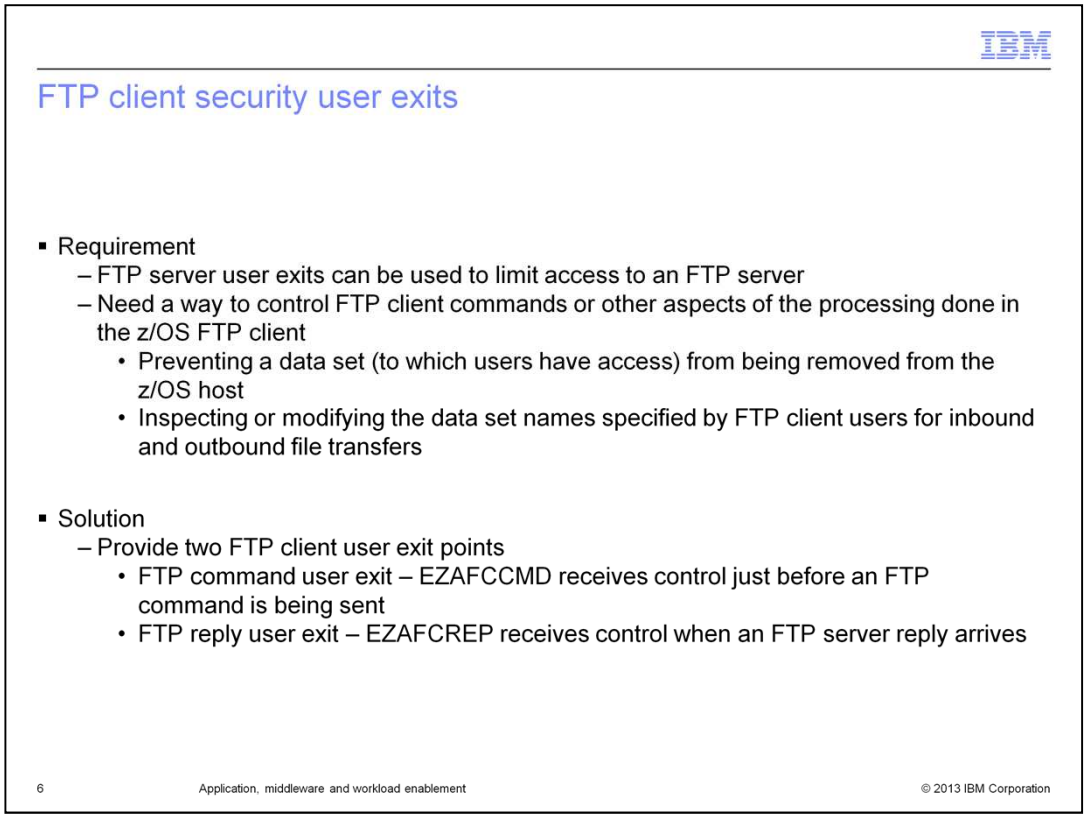


Users can configure the FTP daemon in various places, such as START parameters, FTP.DATA data set, TCPIP.DATA data set, and UNIX environment variables. Before V2R1, there was no way to get a summary of FTP daemon configuration information. And similarly, there was no way to programmatically obtain the TN3270 server profile information.



To allow you to get FTP daemon configuration data, in V2R1 a new TCP/IP callable network management interface request type is introduced. The new network management interface request type, GetFTPDaemonConfig, allows network management applications to programmatically obtain the configuration data for active FTP daemons. The FTP daemon configuration data returned by the network management interface is in a format that is easy for parsing. Also included in V2R1 is a new type 119 SMF subtype 71 record for FTP daemon configuration data. This new System Management Facilities record contains the FTP daemon configuration data and is written after the FTP daemon finishes its initialization and listens on the listening port successfully for the first time. This new System Management Facilities record is also available to the real-time System Management Facilities data network management interface.

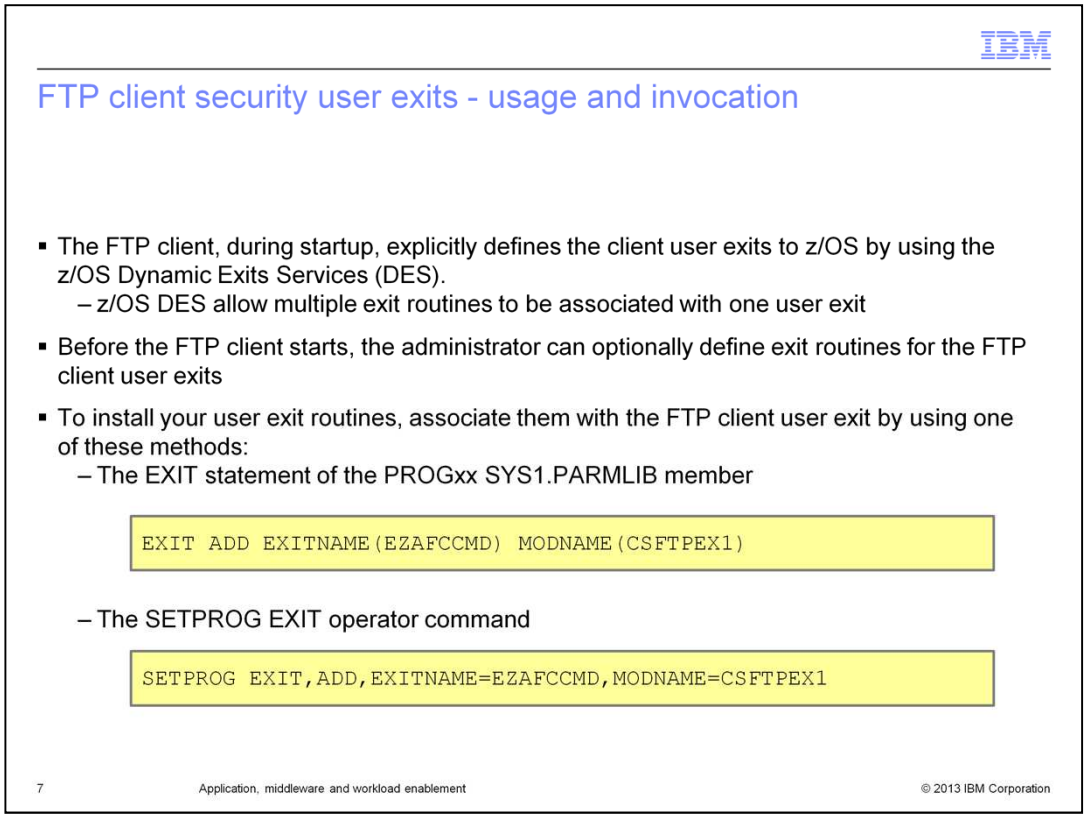
To allow you to get TN3270 server profile information, in V2R1 another new TCP/IP callable network management interface request type is introduced. The new network management interface request type, GetTnProfile, is used to obtain information about the current TN3270 profile statement settings. This request does not support filtering. To detect changes to the profile statement settings, you can use this callable request to obtain an initial set of current profile settings. Then you can repeat the request periodically and compare the data returned in the current response with the data returned in previous responses. Also in V2R1 is a new SMF type 119 subtype 24 TCP/IP profile event record. This record provides information about changes to the profile settings that are made using the VARY TCPIP,tnproc,OBEYFILE command.



The FTP server provides several user exits, which can be used to limit access to the FTP server. For example, the FTP server user exit FTCHKCMD can be called when the server receives an FTP command, for example, RETR or STOR. FTCHKCMD can choose to accept, reject, or modify the command. Similarly, the FTP server user exit FTPOSTPR can be called upon completion of the FTP commands RETR, STOR, STOU, APPE, DELE, and RNTO. It can choose to perform actions based on any of the information passed to it.

However, prior to V2R1, the administrator had no way of controlling the FTP client commands or other aspects of the processing done in the FTP client. For example, the administrator might want to protect some data sets, to which users have access, from being removed from the z/OS host. The administrator might want to inspect or modify the data set names specified by users of the FTP client on PUT, or MPUT, and GET, or MGET, operations. The administrator might want to cancel the address space of an FTP client, if that client is in the process of sending an 'unauthorized' FTP command.

To satisfy this requirement, z/OS V2R1 provides two FTP client user exit points. The FTP command user exit, EZAFCCMD, receives control just before an FTP command is being sent. It enables the user exit to inspect the command, optionally modify the command arguments, reject the command, or request the FTP client session to be terminated. It is important to note that this client user exit cannot change the FTP command itself, only its arguments. The FTP reply user exit, EZAFCREP, receives control when an FTP server reply arrives. It allows the user exit to analyze the results of commands that were sent to the server and to request that the FTP client session be terminated. This client user exit cannot modify FTP server replies.



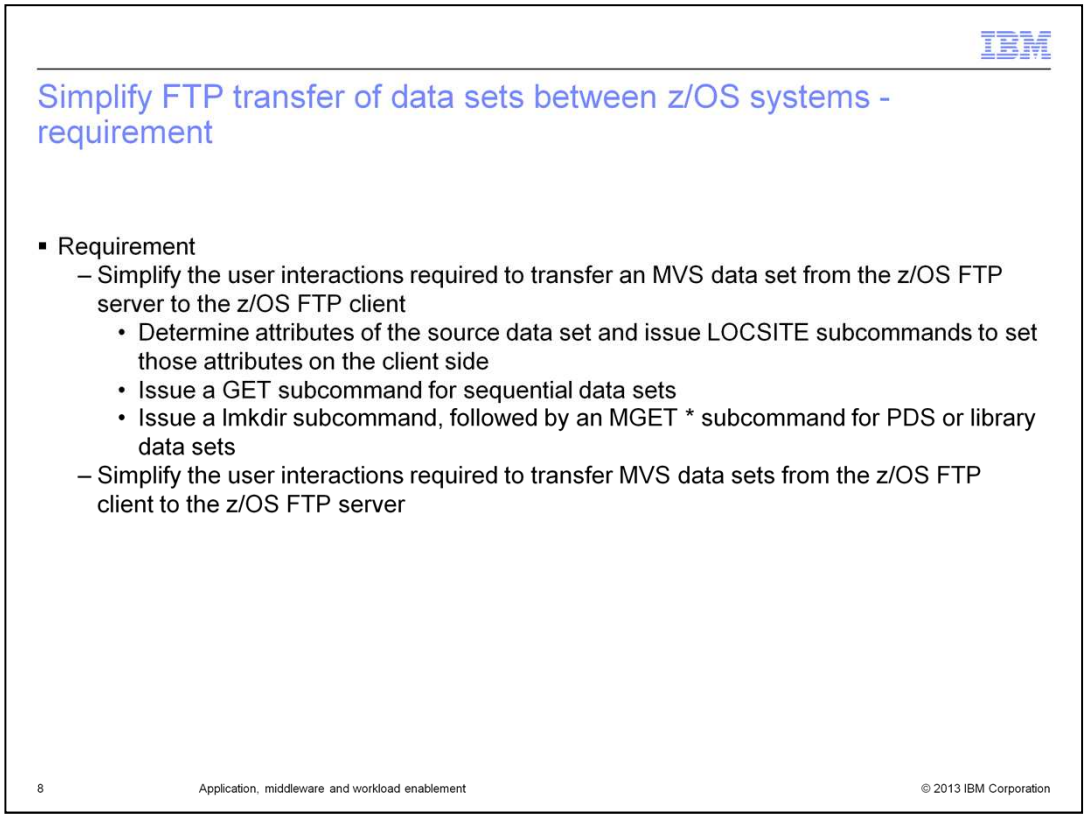
The FTP client user exits are defined and managed using the dynamic exits services of z/OS. This is different from the FTP server user exits. It ensures that users do not replace the FTP client user exit with their own. The z/OS dynamic exits services protect the FTP client from calling an unauthorized exit routine by calling only those exit routines that the system programmer or operator has explicitly installed.

In the examples shown on this slide, CSFTPEX1 is the name of the FTP client user exit routine that you want the FTP client to call at the EZAFCCMD user exit interface.

In the first example, the EXIT statement of the PROGxx parmlib member is used to associate the exit routine with the FTP client's user exit. The EXIT statement of PROGxx allows an installation program to add exit routines to a user exit. At IPL, you can use PROG=xx to specify the particular PROGxx SYS1.PARMLIB member that you want the system to use. During normal processing, you can use the SET PROG=xx command to set a current PROGxx SYS1.PARMLIB member.

In the second example, the SETPROG EXIT command is used to associate exit routines with the user exit.

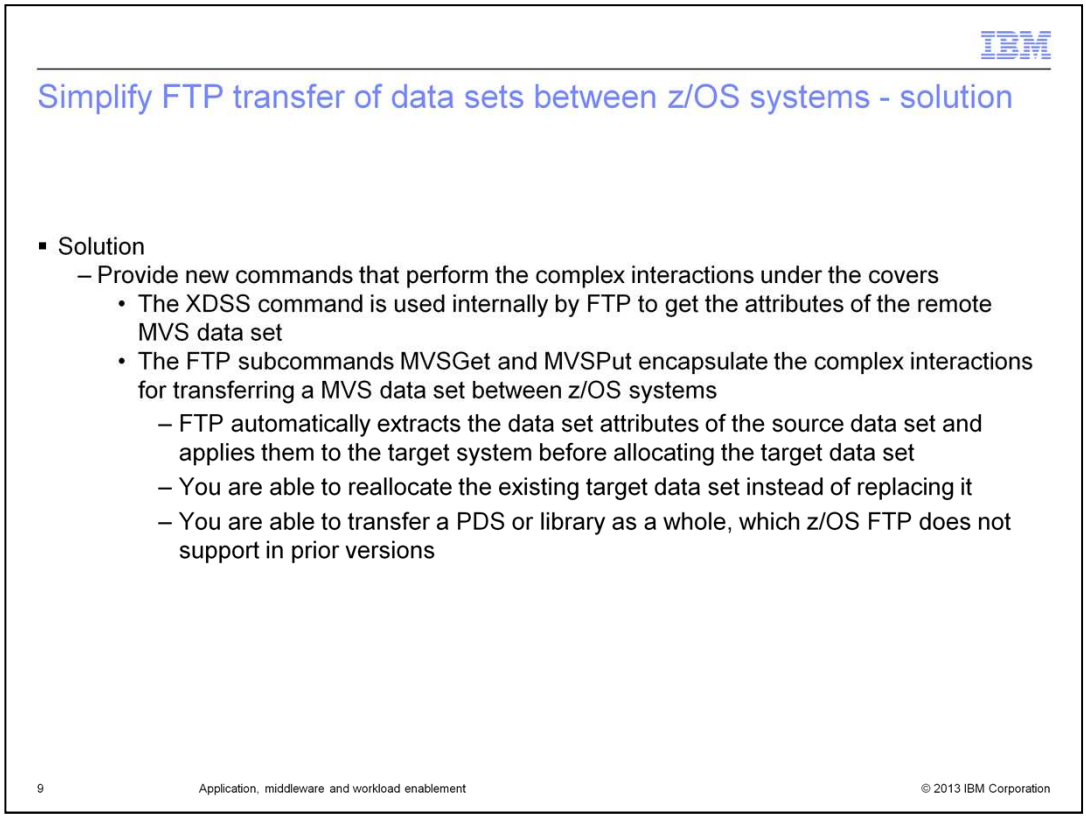
z/OS dynamic exits services allow multiple exit routines to be associated with one user exit. The FTP client cannot control the call sequence of multiple exit routines. The order is determined by the EXIT ADD statements in the PROGxx SYS1.PARMLIB member or the SETPROG EXIT ADD commands. Both techniques do support FIRST and LAST options to control whether a new exit load module is to be placed before or after the exit routines that have already been defined.



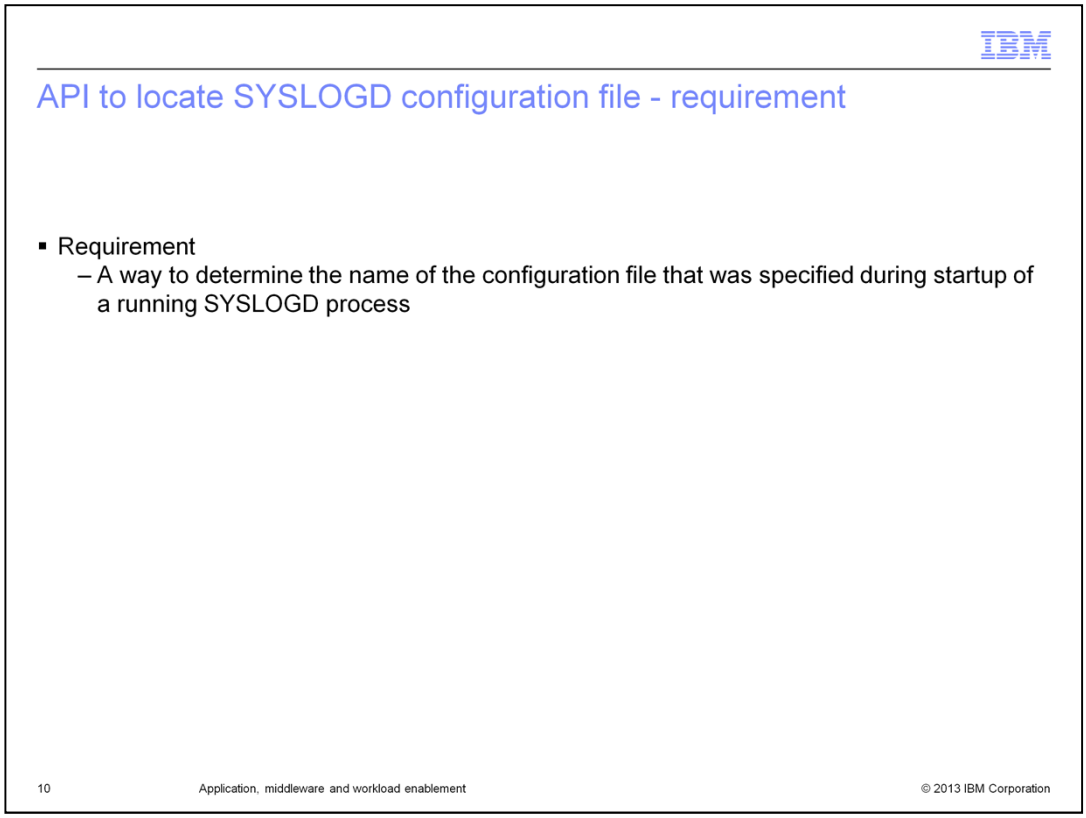
When you log in to the z/OS FTP server with the z/OS FTP client to get an MVS data set from the server, you have to know a lot about the source data set. You also have to follow many steps to complete the transfer. The attributes of the source data need to be determined, for example, record format information, such as LRECL, RECFM, and BLKSIZE, and SPACE information, such as allocation units, and primary and secondary allocation. Other examples are the data set type, such as sequential, PDS, or library, and, if PDS, the number of directory blocks.

One or more LOCSITE subcommands have to be issued to set these attributes on the FTP client side. For sequential data sets, a GET subcommand is then issued. For PDS or library data sets, a lmkdir subcommand, followed by an MGET * subcommand, must be issued. Furthermore, the transfer might result in the source and target data sets having mismatched attributes, if the target data set already exists. This increases the likelihood of data loss because of wrapping, truncation, space constraints, and so on.

The same problem exists when you want to transfer an MVS data set from the z/OS FTP client to the z/OS FTP server.



z/OS V2R1 provides new commands to simplify the FTP data transfer between z/OS systems. A new FTP command, called XDSS, is used internally by FTP to get the attributes of the remote MVS data set and send them back to the z/OS FTP client in a 200 reply. The complex interactions for transferring an MVS data set between z/OS systems are encapsulated in the new FTP subcommands MVSGet and MVSPut. With MVSGet or MVSPut, FTP will automatically extract the data set attributes of the source data set and then apply them to the target system before allocating the target data set. With these subcommands, you are able to reallocate the existing target data set instead of replacing it. This enhances the reliability of data set transfers because the source and target data set attributes are now guaranteed to match. You are also able to transfer a PDS or library as a whole. z/OS FTP does not support this before z/OS V2R1.



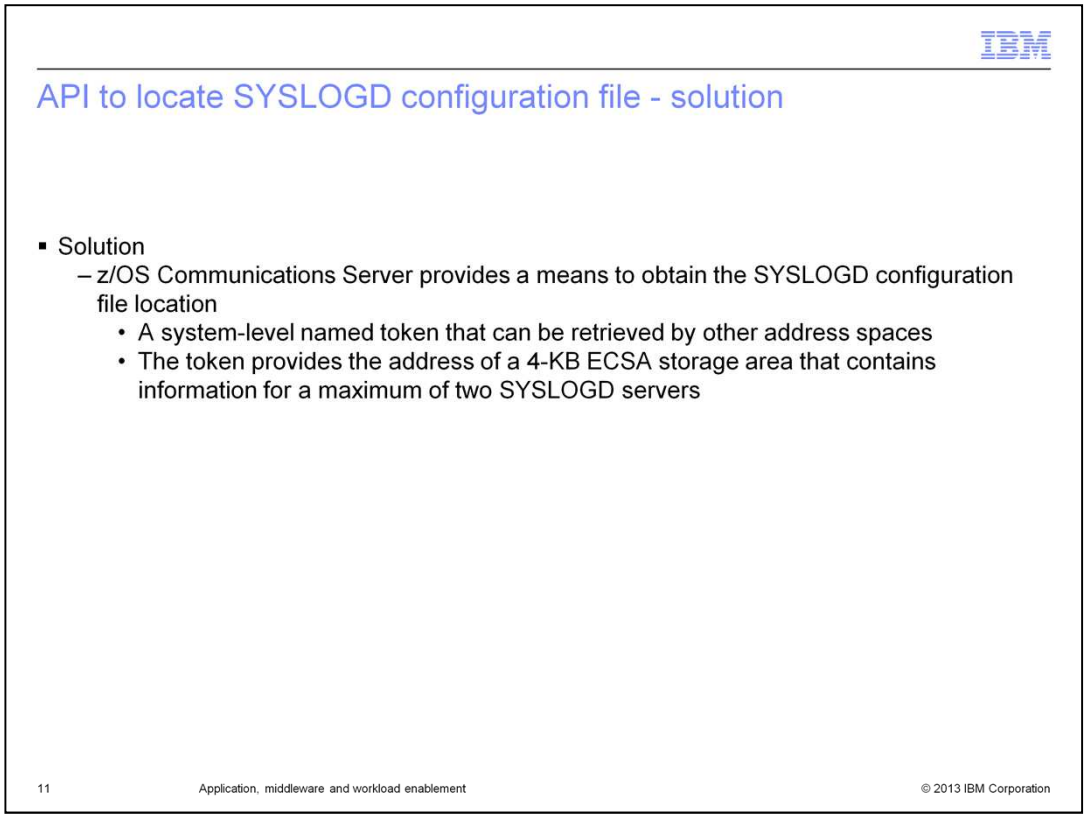
The syslog daemon, SYSLOGD, reads and logs system messages to the MVS console, log files, System Management Facilities, other machines, or users. The syslog daemon processing is controlled by a configuration file. The default configuration file is called /etc/syslog.conf. The syslog daemon can run in one of three modes: normal, local-only, or network-only.

In normal mode, the syslog daemon processes logging requests from the local system and applications by using the syslog() function. And it processes messages sent over the network by remote systems that are running the syslog daemon. Only one instance of the syslog daemon can be run on a z/OS system, if the syslog daemon is started in normal mode.

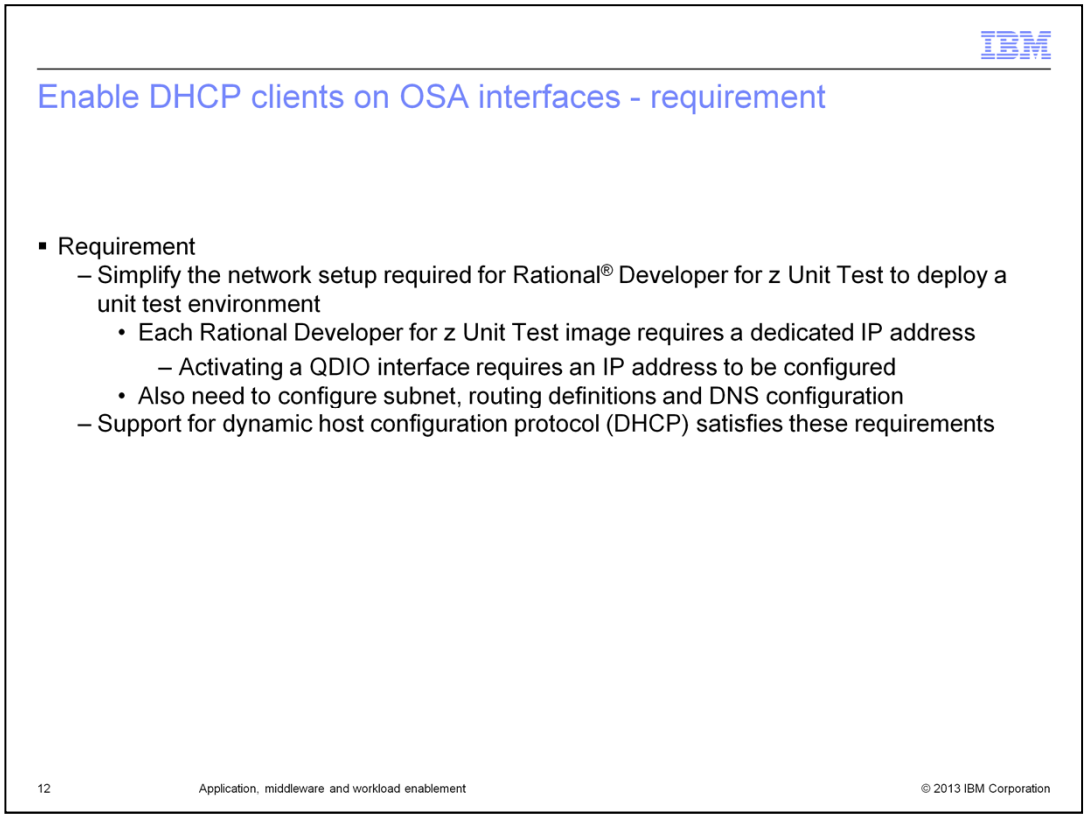
In local-only mode, the syslog daemon processes logging requests from only the local system and applications. This instance of the syslog daemon does not receive or process messages sent over the network from remote syslog daemons.

In network-only mode, the syslog daemon processes only messages sent over the network by remote systems running the syslog daemon. This instance of the syslog daemon does not process logging requests from the local system or applications.

Before V2R1, only the running syslog daemon process knew the name of the configuration file that was specified with the -f option during startup of the syslog daemon.

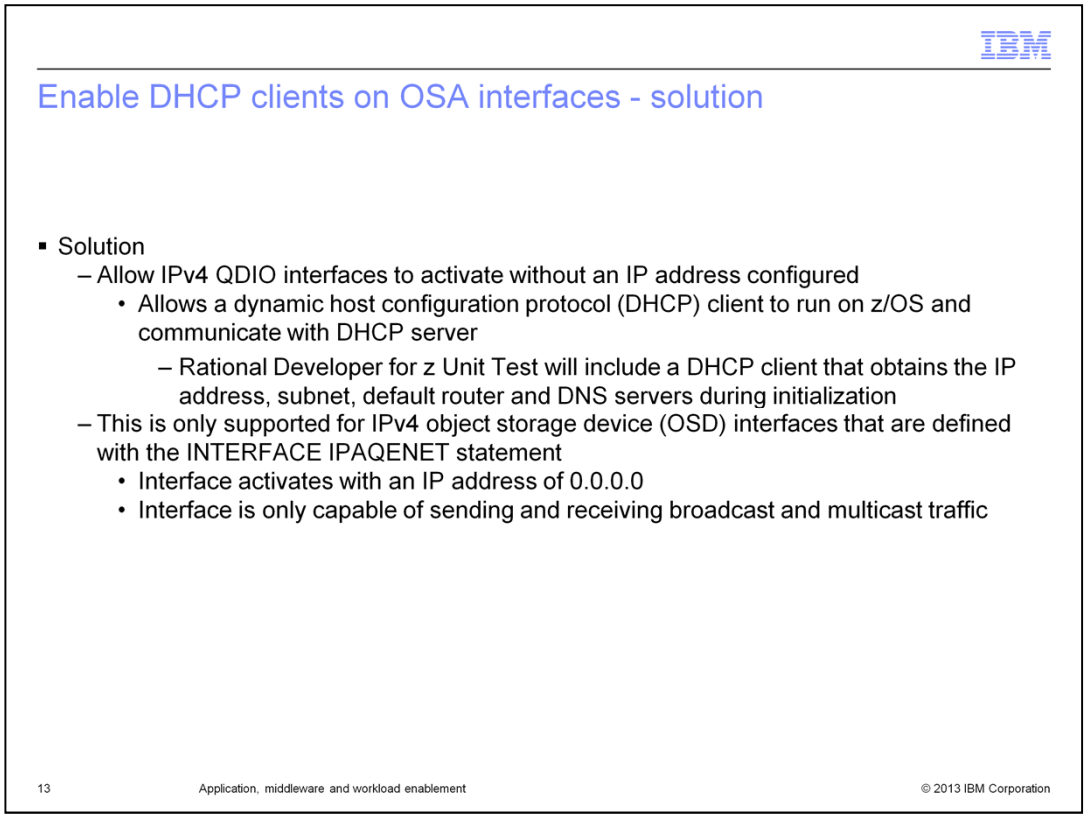


Starting in V2R1, there is an API for determining the syslog daemon configuration file location and name. A system-level name-token pair allows other address spaces to retrieve this configuration file information. The name-token pair persists after the termination of the job that creates the syslog daemon. One name-token pair is created and one ECSA storage area is used for the syslog daemon, regardless of which mode it is running in. The single storage area will never be freed; it is reused. It will contain two instances of the same data structure, one for the local syslog daemon and one for the network syslog daemon, allocated with a single storage allocation. Updating data in the ECSA data area is serialized. But there is no serialization for retrieving data in the ECSA data area. This means that the data can be in the process of being changed while it is being accessed.



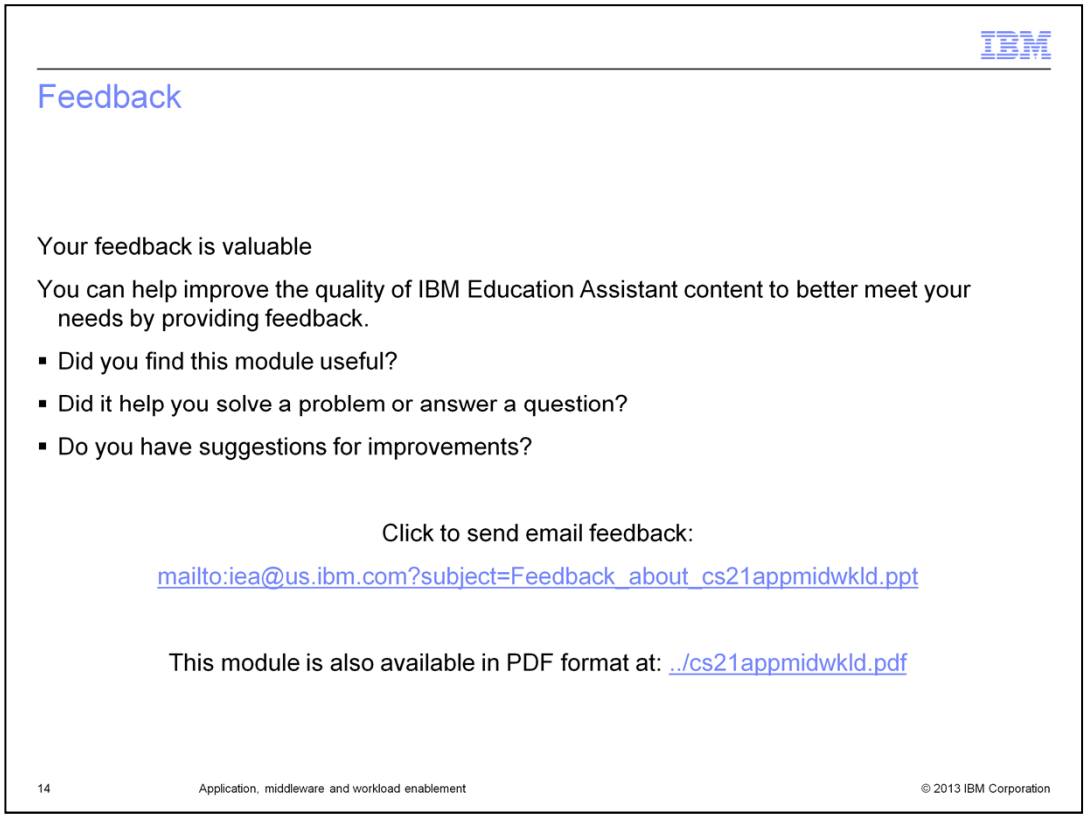
Rational Developer for z Unit Test is an IBM tool that creates a personal z/OS development and unit test environment on each developer's desktop machine or on a shared server. Rational Developer for z Unit Test emulates System z architecture on top of x86 processor architecture. This includes emulation of QDIO and OSA Express at the level it was in the year 2005.

Currently, each Rational Developer for z Unit Test image requires a dedicated or fixed IP address because a QDIO interface will not activate without a specified IP address. Additionally, the subnet mask, routing definition, and DNS servers need to be configured. Support for dynamic host configuration protocol would significantly improve the usability of network configuration tasks related to deploying unit test environments under Rational Developer for z.



In V2R1, IPv4 QDIO interfaces are now allowed to activate without an IP address being configured. Rational Developer for z Unit Test includes a DCHP client that runs on z/OS and communicates with a DHCP server. The DHCP client obtains an IP address, subnet mask, default router, and DNS servers from the DHCP server during initialization.

This functionality is only supported for IPv4 object storage device interfaces that are defined with the INTERFACE IPAQENET statement. The interface completes activation with an IP address of 0.0.0.0. An IP address of 0.0.0.0 is not allowed to be explicitly configured on the interface statement. The interface is only capable of sending and receiving broadcast and multicast traffic.



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, disclaimer, and copyright information