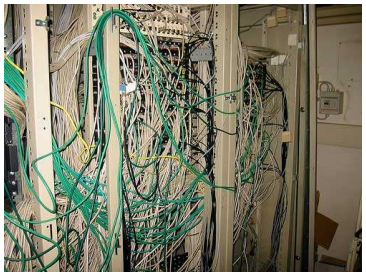
This presentation describes the enhancements to the Communications Server in z/OS V1R11 for simplification and usability for the policy agent.

Simplification and usability for policy agent

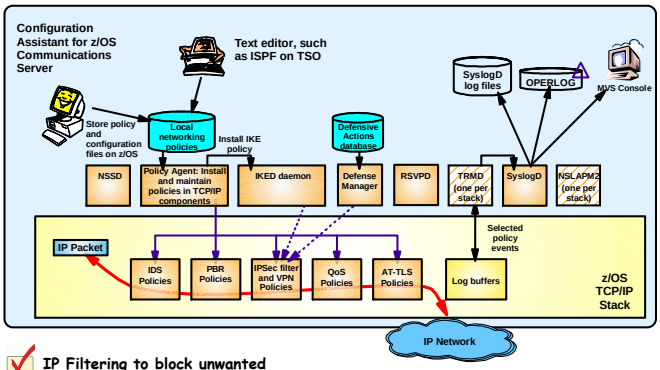
- Policy infrastructure management
- MVS console support for selected TCP/IP commands
- Configuration Assistant



The main area of enhancements within this theme is the networking policy infrastructure and functions around it. That includes the IBM Configuration Assistant for z/OS Communications Server, the policy agent, and the syslog daemon.

z/OS Communications Server policy-based networking infrastructure overview

- **Perceived by many customers as a complex infrastructure**
  - Some initial cost to set up and enable the infrastructure
  - Difficult to manage and operate the infrastructure
  - But many valuable functions
- **z/OS V1R11 Communications Server simplifies the overall setup and operation of the networking policy infrastructure**
  - Making it simpler to gain the benefits of the networking policy-based functions on z/OS



IP Filtering to block unwanted traffic from entering or leaving your z/OS system

Connection-level security for TCP applications without application changes

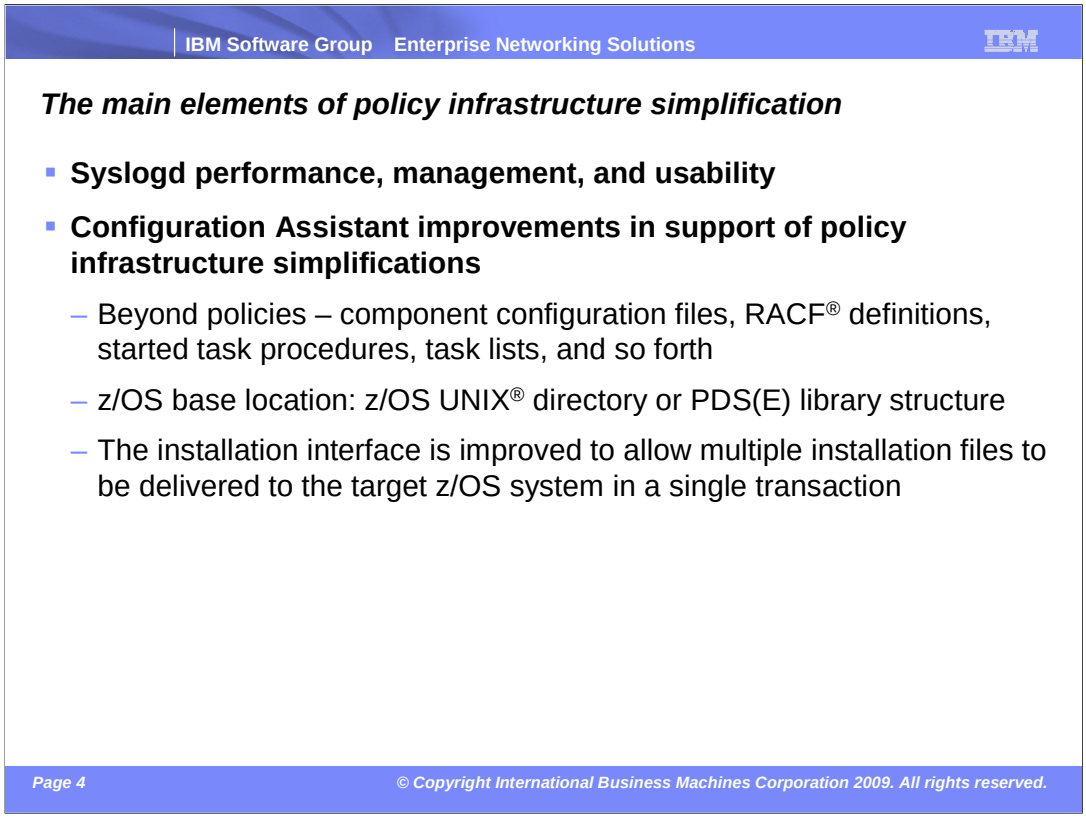
Making sure high-priority applications also get high-priority processing by the network

Application-specific selection of outbound interface and route (Policy-based routing PBR)

Providing secure end-to-end IPSec VPN tunnels on z/OS

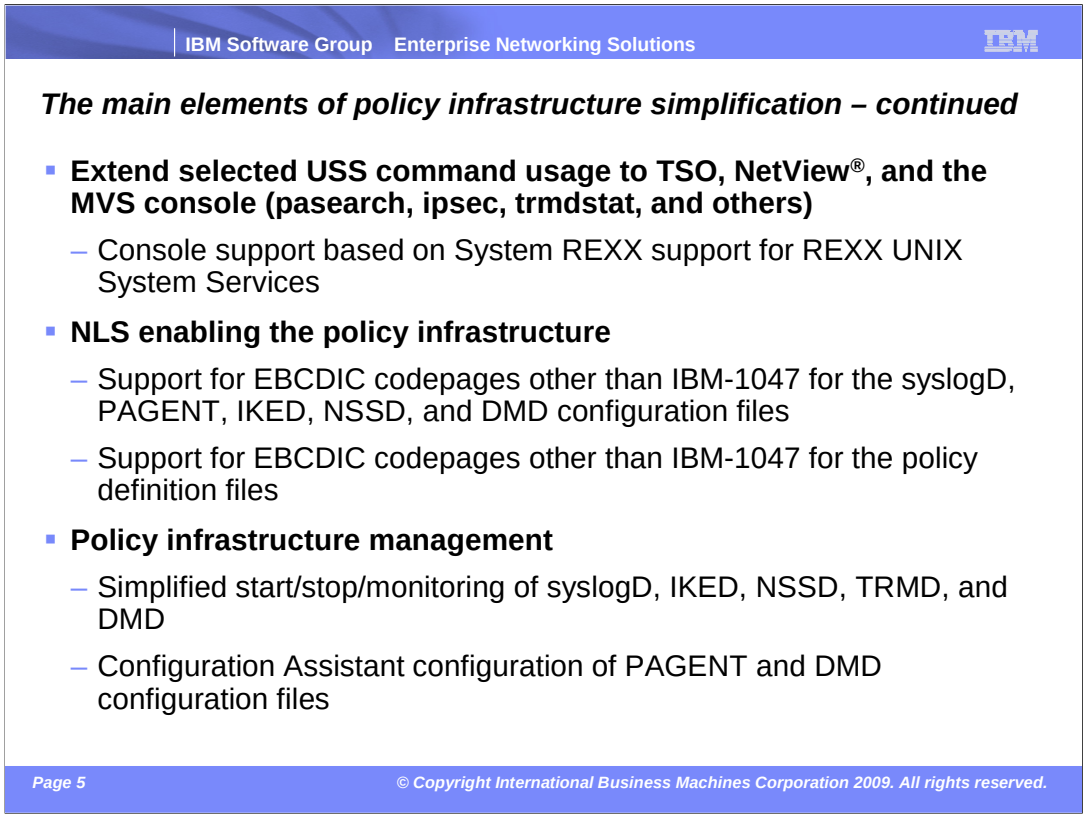
Protection against "bad guys" trying to attack your z/OS system

This slide is a reminder of what the z/OS Communications Server networking policy infrastructure is and supports. The infrastructure consists of many components that together deliver support for a range of policy-based networking functions on z/OS. IP filtering, IP Security, Application Transparent SSL/TLS, network quality of service, Intrusion detection, and policy-based routing. None of these functions are available unless the networking policy infrastructure has been customized and set up.



Syslog daemon is used not only for policy-based logging, but also for logging from other z/OS UNIX applications. However, with respect to networking policies, syslogd becomes a very central element of the full setup. Without syslogd, an installation does not have an audit trail of events that have been detected by the IP security or IDS components of z/OS Communications Server. Ensuring log messages are captured and retained for a certain period of time becomes crucial to the overall reliability of the networking policy environment. For the details of the syslogd changes, see the syslogd presentation.

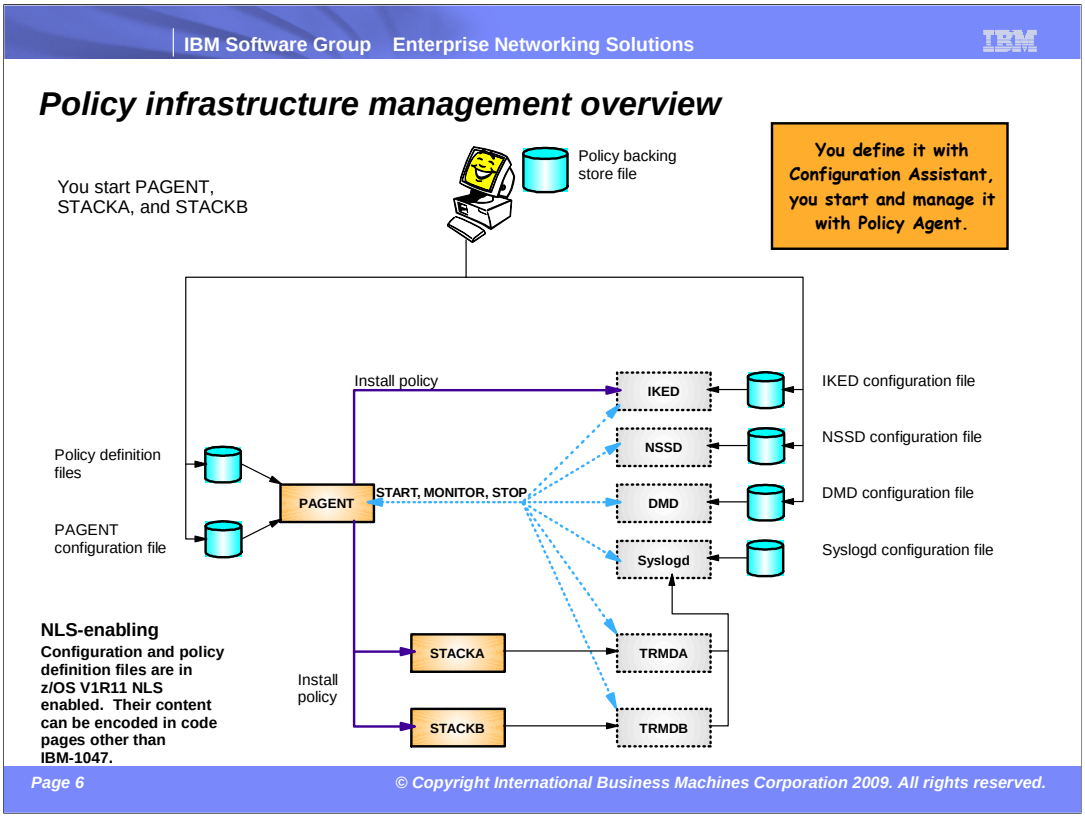
The Configuration Assistant is enhanced in many ways in support of the overall objective to simplify the z/OS networking policy infrastructure. Configuration Assistant will now create more configuration files (policy agent and Defense Manager daemon). It will create sample RACF command input when setting up new elements of the policy infrastructure, and it will create sample started task JCL procedures. Policy definitions, RACF commands, sample procedures, and so on can now be transferred to z/OS in single ‘transactions’. A base location per image can be set up in such a way that Configuration Assistant will transfer all objects into members of one or more base PDS(E) libraries.



A selected set of policy-based z/OS UNIX shell commands are available in environments other than the z/OS UNIX shell. This set of commands is now available in TSO, the MVS console, and NetView.

Since many of the policy-related files include POSIX-variant characters, the use of EBCDIC code page 1047 has so far been enforced. However, in z/OS V1R11, you can specify which single byte EBCDIC codepage the files are maintained in. The various policy components will honor that code page (and translate from that code page to IBM-1047 when the file is read). This allows customers with NLS-enabled ISPF browsers to better view their policy definitions in TSO. The Configuration Assistant is enhanced to prompt for the required EBCDIC code page. The files are transferred to z/OS and stored in the requested code page.

Finally, the various components of the policy infrastructure did not lend themselves very well to use of the TCP/IP AUTOLOG function. Most of them do not have a listening socket and most of them are not stack-specific, but can serve more stacks in a common INET environment. Policy agent is enhanced with a start/monitor/stop capability that allows installations to start policy agent. The policy agent will then start and monitor the remaining necessary policy components.



You use the policy infrastructure to define and manage several different policy types for your network.

The Quality of Service policy type lets you assign different network priorities to different types of work. You can also implement connection limits to certain servers, apply differentiated service principles to aggregates of traffic types, and perform other functions.

The Intrusion Detection Services policy type lets you indicate the types of actions to take when various types of intrusions are detected, and to regulate TCP and UDP traffic.

The IP Security policy type applies to two related functions. The first function is IP filtering which allows you to block or accept different kinds of traffic from or to different users or sub-networks. The second function allows you to establish dynamic or manual virtual private networks (VPNs) or tunnels to protect traffic.

The Application Transparent Transport Layer Security policy type lets you deploy TLS or SSL protection on behalf of applications. The applications themselves might not even be aware that SSL or TLS protection is taking place.

The Policy-Based Routing policy type allows you to establish different routing tables for different types of traffic. For example, you can use this policy type to route different traffic over different outbound interfaces.

Several different components or applications make up the policy infrastructure. Depending on the types of policies you use, some or all of these components might be needed.

The TCP/IP stack implements most policy types. This is where the policies are applied as needed to inbound and outbound traffic.

The syslog daemon (syslogd) provides a system-wide logging mechanism which is used by several policy components.

The Configuration Assistant is an easy-to-use interface for defining policy definitions and configuration files for most policy infrastructure components. You can also define these files manually.

The policy agent (pagent) reads and parses policy definitions, installs these policies in the TCP/IP stacks, and provides IPSec VPN-related policies to the IKE daemon.

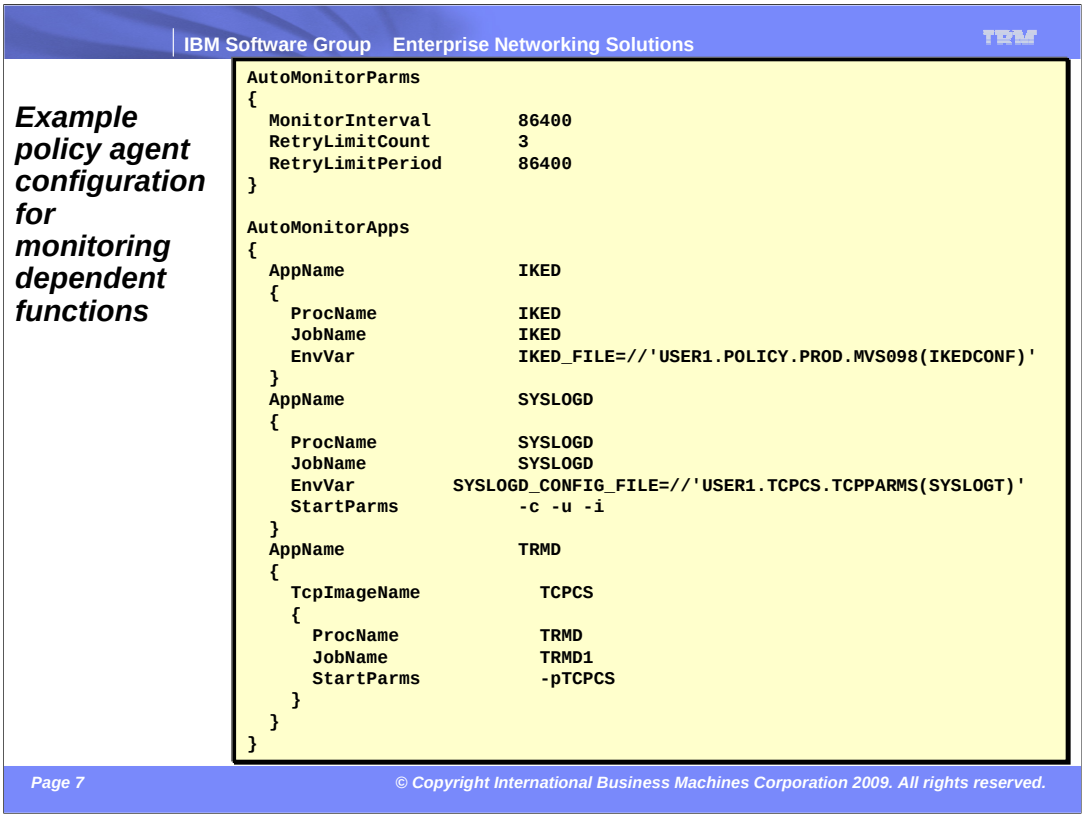
The Traffic Regulation Management daemon (TRMD) writes a variety of security related messages to syslogd. You can use the trmdstat command to produce a variety of reports.

The IKE daemon (IKED) provides support for dynamic and manual VPNs. It uses certain IPSec policies to communicate with its peers on other systems to establish secure tunnels to protect data being sent or received.

The Network Security Services server daemon (NSSD) provides certificate and network management services for IPSec, in addition to other services for XML appliances.

The Defense Manager daemon allows an external security monitor function to install temporary IP filters to block a specific attack or a pattern of attacks.

The Network SLAPM2 subagent (nslapm2) monitors the effectiveness of QoS policies.



This slide shows an example of a policy agent configuration for monitoring dependent functions. The Configuration Assistant will generate the initial set of definitions. You might want to update file locations and other fields.

The **AutoMonitorParms** statement in the policy agent main configuration file configures global application monitoring parameters.

The **MonitorInterval** parameter specifies the number of seconds in the monitor interval, which is how often policy agent checks to see if the monitored applications are active. The default is 10 seconds.

The **RetryLimitCount** and **RetryLimitPeriod** parameters work together. They indicate how many times within a given time period applications should be restarted. The default is five times within 600 seconds (10 minutes). If these limits are exceeded, policy agent stops monitoring the application, and does not try to restart it any more. After you've resolved the application problem, you can restart the application and resume monitoring using an operator command. The commands are described later in this presentation.

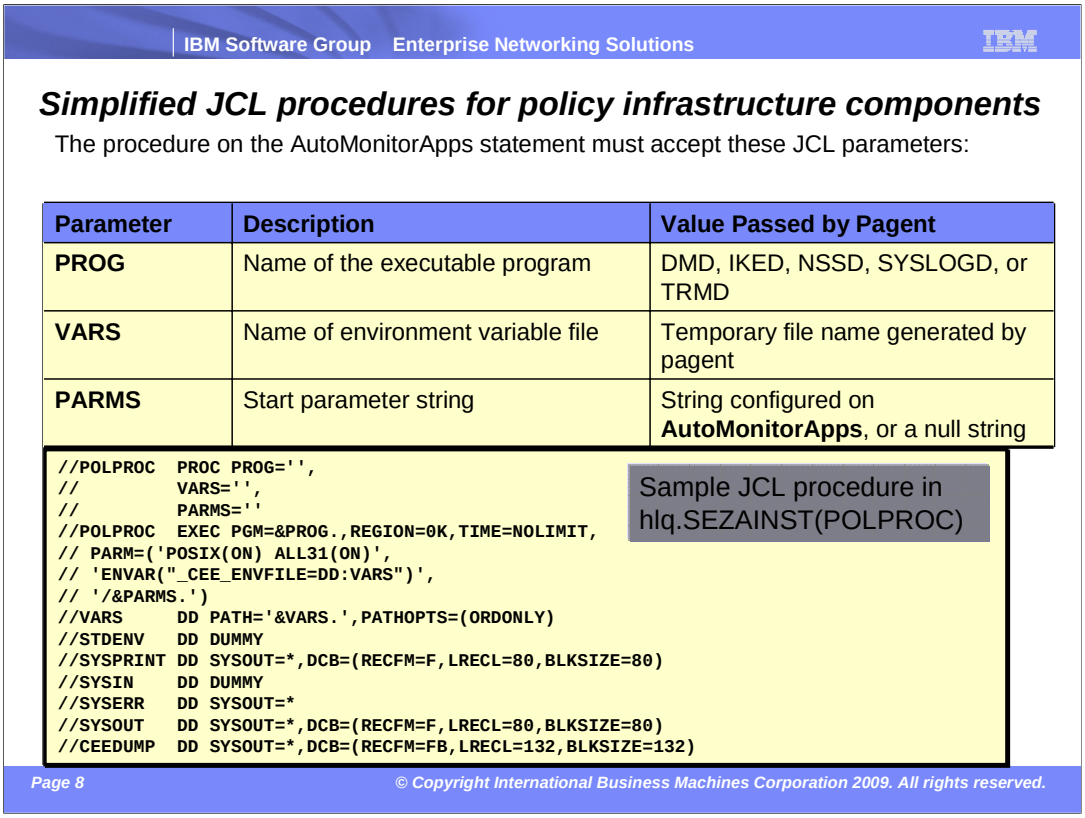
The **AutoMonitorApps** statement in the policy agent main configuration file configures which applications are to be monitored and parameters specific to those applications. You use the **AppName** parameter to specify each application, and repeat this parameter for the remaining applications you want to monitor. For those applications that run a unique instance on each TCP/IP stack, specify the **TcpImageName** parameter for the stack name, and repeat this parameter for each stack. You must also configure the **TcpImage** statement in the main configuration file for each stack configured on the **AutoMonitorApps** statement.

The **ProcName** parameter is required, and specifies the name of the cataloged procedure that starts the application. The specified procedure must accept parameters that are passed to it by policy agent. The parameters are detailed on the next slide.

The **Jobname** parameter specifies the job name that is used when the application runs. It defaults to the **AppName** parameter. You should specify a unique job name for each instance of TRMD.

The **StartParms** parameter specifies the start options for the application. The maximum length of this parameter is 45 characters, so use environment variables for long values where possible.

The **EnvVar** parameter specifies an environment variable. For example: EnvVar TZ=EST5EDT. Repeat this parameter for each environment variable you want to specify.



Three JCL parameters must be accepted by the cataloged procedure that policy agent uses to start or restart monitored applications.

The **PROG** parameter specifies the name of the executable program, and one of the supported application names is always passed by policy agent.

The **VARS** parameter is the name of a file containing environment variables. Policy agent creates a temporary file and populates it with the configured environment variables. It then passes the name of the temporary file to the procedure.

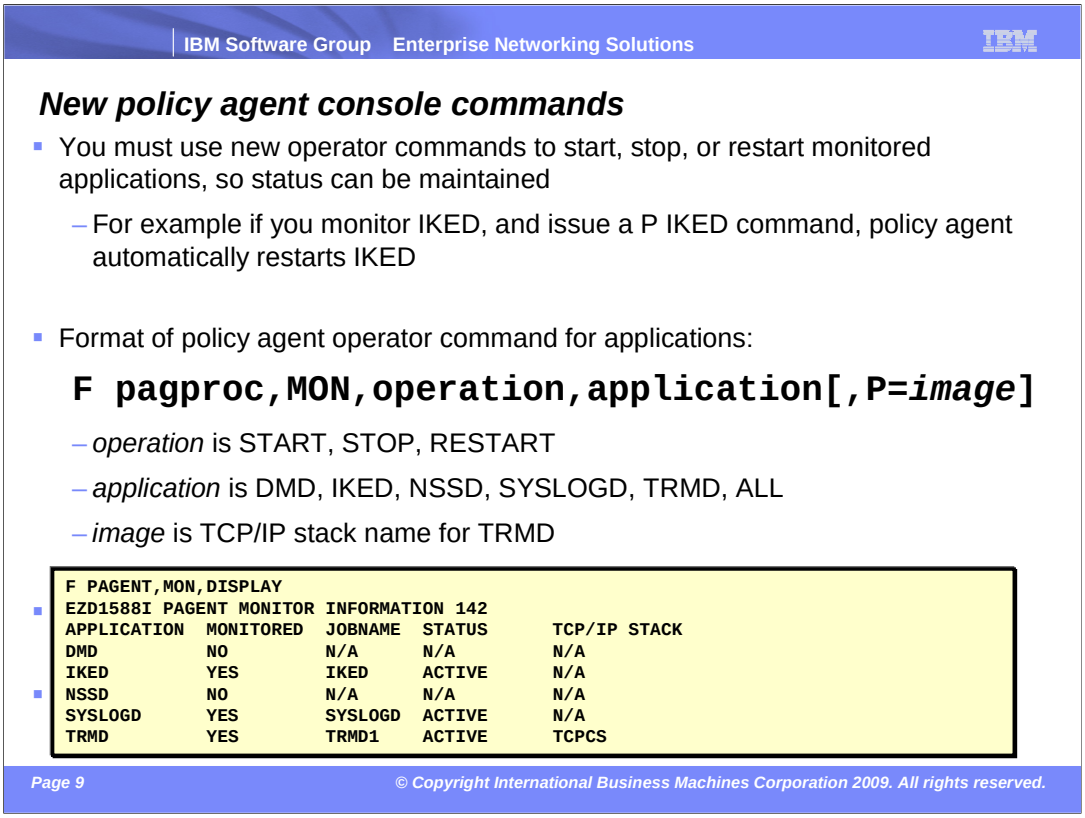
The **PARMS** parameter specifies the start parameter string. Policy Agent passes the configured string, or a null string if no start parameters are configured.

You can use the sample procedure shipped in SEZAINST(POLPROC) as a template to develop your own procedures.

When policy agent starts or restarts an application, it waits up to one minute for the application to become active. If the application is not active after one minute, policy agent restarts it. You should allow for this one minute start wait when configuring the **RetryLimit** and **RetryPeriod** parameters. For example, if you configure five retries within a three minute period, policy agent will never stop trying to restart an application. This is because it takes three minutes to restart the application three times, so you never reach five retries within that period.

When an application stops unexpectedly, policy agent is immediately informed. The application is restarted after a short delay. Notice that this occurs regardless of the configured monitor interval.

You can configure an application to be monitored, but then start the application before starting policy agent. If you do this, and the application was already running with the configured job name, there are several consequences. First, policy agent will still try to start the application, and the start will fail because the application is already active. Also, if policy agent needs to later restart the application, it uses the configured procedure name and job name. These might not match the procedure name and job name that you used to originally start the application. It's therefore best if you do not start monitored applications before starting policy agent. The one exception is syslogd, which you normally want to start very early. Notice that if you start a monitored application using a different job name than configured, policy agent is not able to successfully monitor the application because it looks for active applications using the configured job name.



When you monitor applications using the policy agent, you need to use a set of new policy agent operator commands to start, stop, or restart the applications. This allows the policy agent to keep track of the current status of the applications. One example of why this is needed is as follows. If you were to stop an application directly, for example by issuing a P IKED command, policy agent does not know you intended to stop IKED and restarts it.

The format of the new commands is shown on this slide. You can perform start, stop, and restart operations against an individual application, or all applications. For TRMD, you can select which instance by using the P=*image* parameter on the command.

If you stop the policy agent, all monitored applications remain active. If you want to stop all policy infrastructure components that are being monitored, issue the MODIFY MON,STOP,ALL command before stopping policy agent.

You can use the MODIFY MON,DISPLAY command to display the current status of all applications. This includes whether the application is monitored, the job name, and the status. For TRMD, it also includes the associated stack name.

The application status is the most important piece of information in case of problems.

STOPPED - Application has never been started, failed to start, or was manually stopped

INACTIVE - Application is temporarily inactive (for TRMD this means the stack is inactive)

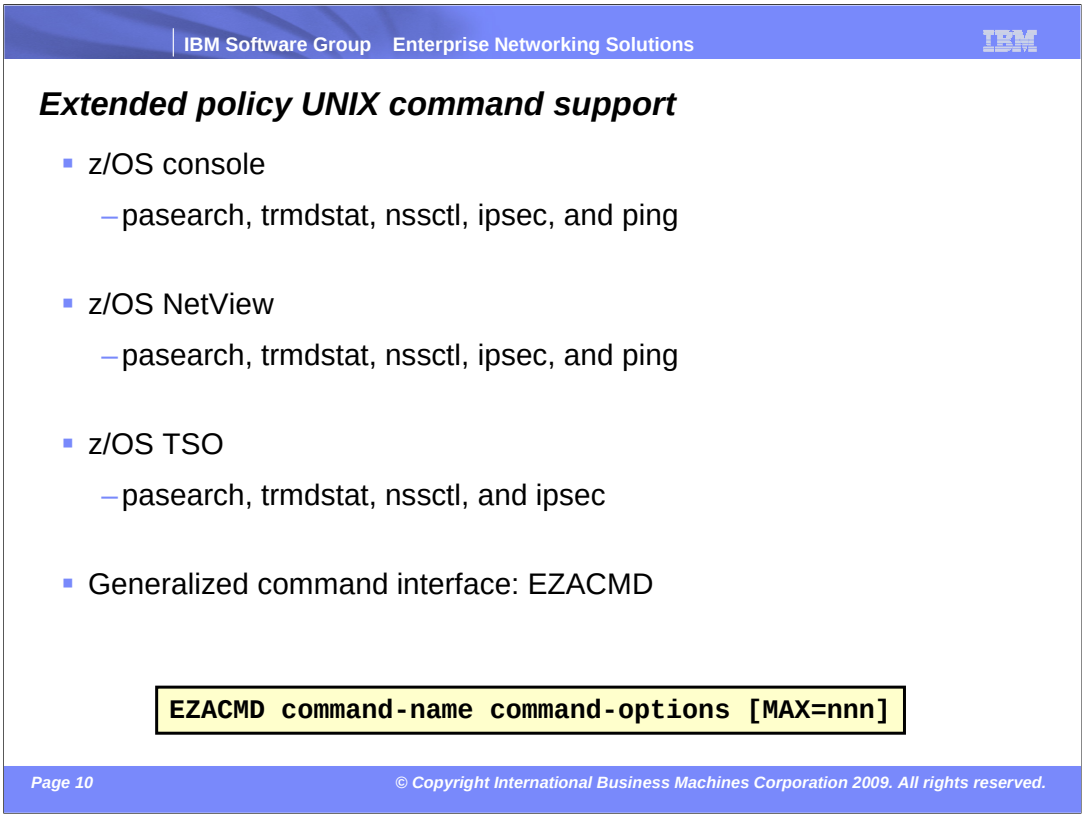
STARTING - Application has been started but is not yet active

RESTARTING - Application has been restarted but is not yet active

STOPPING - Application has been stopped but is not yet inactive

ACTIVE - Application is active

N/A - Application is not being monitored



z/OS V1R11 makes the z/OS UNIX commands listed on the slide available in three command environments: z/OS console, NetView, and TSO. Only the z/OS UNIX commands listed on the slide are available in these environments.

Ping is not a policy-related command, but customers have asked for ping from the z/OS console for many years. The infrastructure that was built for the policy-related commands was very easily expanded to also support ping. Since TSO has a native TSO ping command already, the z/OS UNIX ping was not made available in TSO.

EZACMD is a generalized interface to the selected set of z/OS UNIX commands. The same EZACMD command is used from TSO, the z/OS console, and NetView. Each environment has specific requirements and characteristics, which are discussed a little later.

EZACMD supports the command name in any case. Command options are case sensitive and must be entered exactly as documented for the z/OS UNIX command in question.

The MAX keyword specifies the maximum number of output lines. MAX is optional. The default is 100, the maximum is 64000. MAX can be entered in any case and it can be present anywhere after the command-name.

Output from the commands is displayed as-is. Some commands in some of the supported environments will produce output lines that are too long for the display environment. These long output lines wrap to the next line. No attempt is made to re-format the output from the existing z/OS UNIX commands.

EZACMD is delivered as a compiled REXX program in two different system libraries. One library is SYS1.SAXREXEC, which is the system REXX system library. This is a VB, LRECL=255 library. The second library is tcpip.SEZAEXEC, which is the z/OS Communications Server REXX library. This is an FB, LRECL=80 library.

SYS1.SAXREXEC is used from the z/OS console by means of the system REXX infrastructure, which requires a VB, 255 library.

tcpip.SEZAEXEC is used from TSO and NetView.

EZACMD z/OS console support

- The EZACMD command name and options must be entered in quotes to avoid having the input translated to upper-case

```
16.06.59  %%ezacmd 'ping -v w3.ibm.com max=20'

16.06.59  System REXX EZACMD: ping command - start - userID=USER1
16.06.59  System REXX EZACMD: ping -v w3.ibm.com

16.06.59  CS V1R11: Pinging host w3.ibm.com (9.17.137.11)
16.06.59  with 256 bytes of ICMP data
16.06.59  Ping #1 from 9.17.137.11: bytes=264 seq=1 ttl=242 time=66.17 ms
16.06.59  Ping #2 from 9.17.137.11: bytes=264 seq=2 ttl=242 time=61.77 ms
16.06.59  Ping #3 from 9.17.137.11: bytes=264 seq=3 ttl=242 time=61.06 ms
16.06.59  Ping statistics for w3.ibm.com (9.17.137.11)
16.06.59    Packets: Sent=3, Received=3, Lost=0 (0% loss)
16.06.59    Approximate round trip times in milliseconds:
16.06.59    Minimum=61.06 ms, Maximum=66.17 ms, Average=63.00 ms,
 StdDev=2.77 ms

16.06.59  System REXX EZACMD: ping command - end - RC=0
```


z/OS console support uses the z/OS System REXX environment. System REXX in z/OS V1R11 is enhanced in various areas:

Support for REXX programs that use z/OS UNIX system services

Support for multiple REXX libraries

EZACMD is delivered in the System REXX system library. There is no need for adding additional libraries to the System REXX environment.

Configure SYS1.PARMLIB member AXRnn with a System REXX command recognition string. In the examples in this presentation '%%' is used:

AXRnn definition: CPF('%%',SYSTEM)

Follow the System REXX documentation for defining JCL procedures and RACF definitions

A console operator must be logged in to use System REXX commands. To log on, type in the LOGON console command and enter the requested information (user ID and password).

EZACMD can be protected: Create an OPERCMDS resource profile with the name of MVS.SYSREXX.EXECUTE.EZACMD to protect the EZACMD command.

Only logged in console users who are authorized with READ access to that profile can use the EZACMD command from the z/OS console.

This level of security applies to the z/OS console environment only.

The normal TCP/IP-specific SERVAUTH resource definitions related to the UNIX commands are supported in all three environments:

EZB.IPSECCMD.*sysname.stackname.command_type*

EZB.IPSECCMD.*sysname*.DMD_GLOBAL.*command_type*

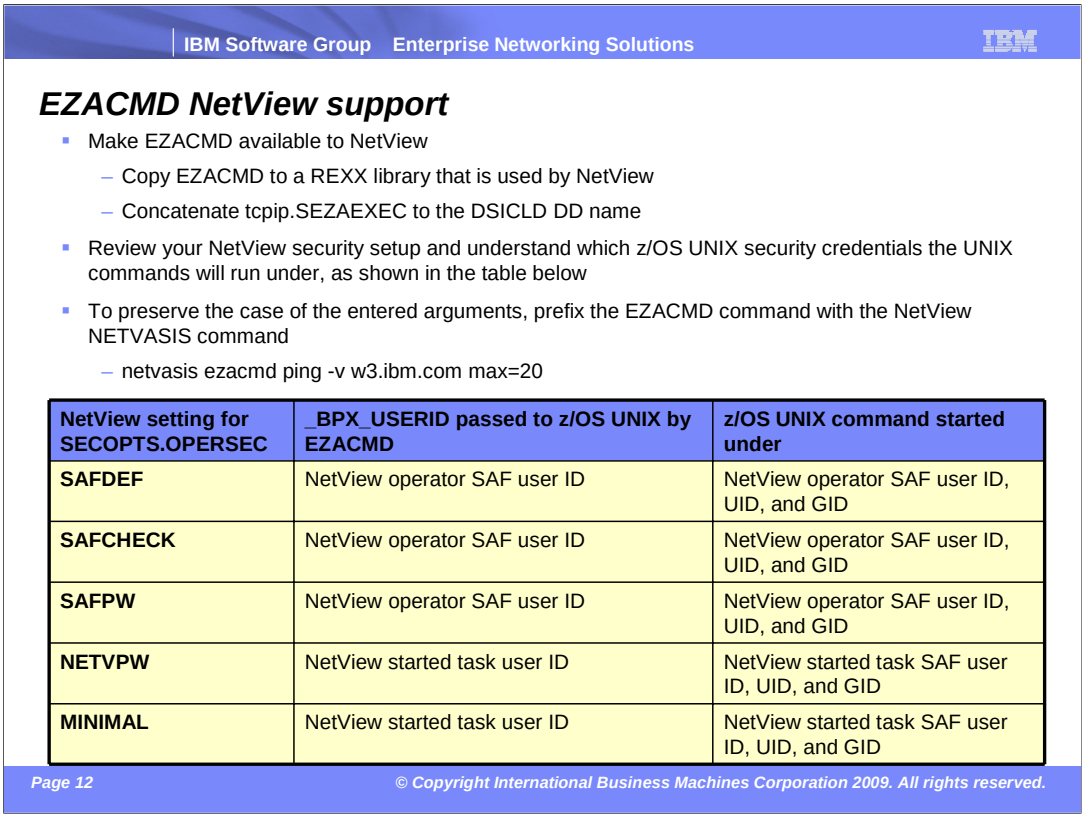
EZB.NETMGMT.*sysname.clientname*.IPSEC.CONTROL

EZB.NETMGMT.*sysname.clientname*.IPSEC.DISPLAY

EZB.NETMGMT.*sysname.sysname*.NSS.DISPLAY

EZB.NETMGMT.*sysname.sysname*.IKED.DISPLAY

EZB.PAGENT.*sysname.image.ptype*



z/OS NetView supports five different types of operator security, which are:

SAFCHECK: Logon passwords are checked by SAF. Attributes are specified in DSIOPF/DSIPRF. DATASET and OPERCMDS classes are checked at the task level.

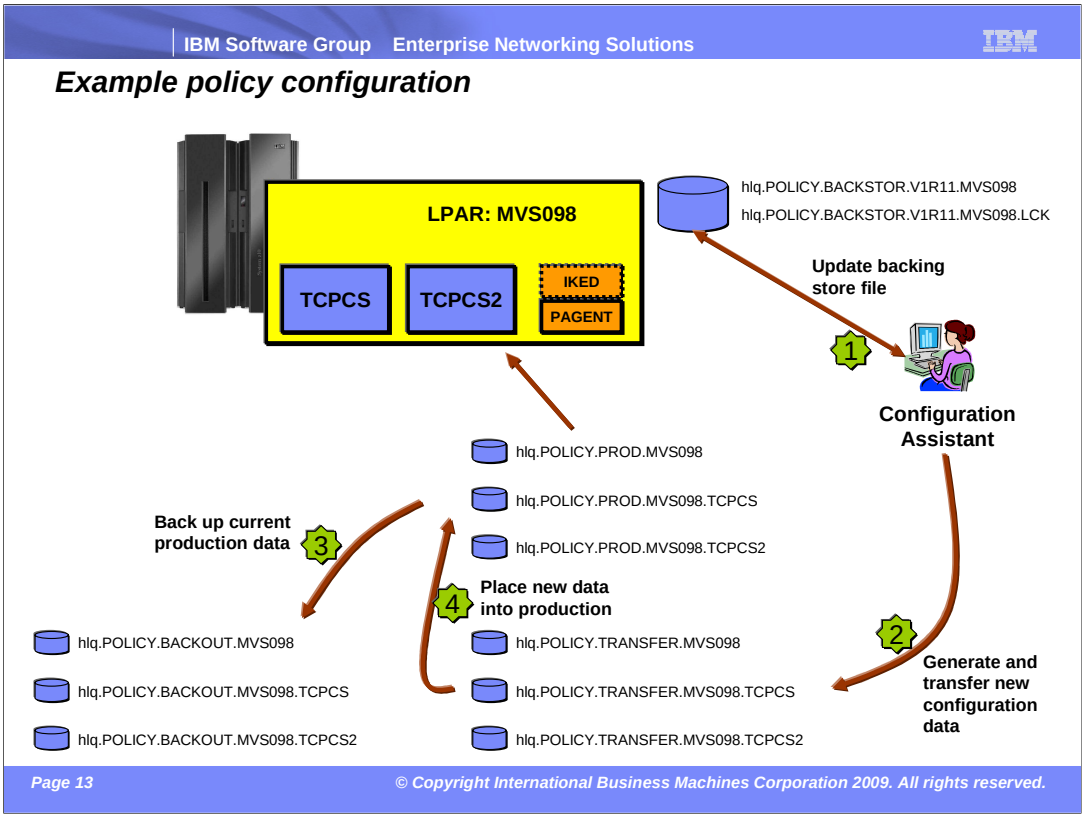
SAFPW: Logon passwords are checked by SAF. Attributes are specified in DSIOPF/DSIPRF. DATASET and OPERCMDS classes are checked at the NetView level.

SAFDEF: SAF checking is done for both logon passwords and attributes (NETVIEW segment).

NETVPW: Logon passwords are defined in DSIOPF or DSIEX12. Attributes are specified in DSIOPF/DSIPRF.

MINIMAL: NetView ignores logon passwords and attributes.

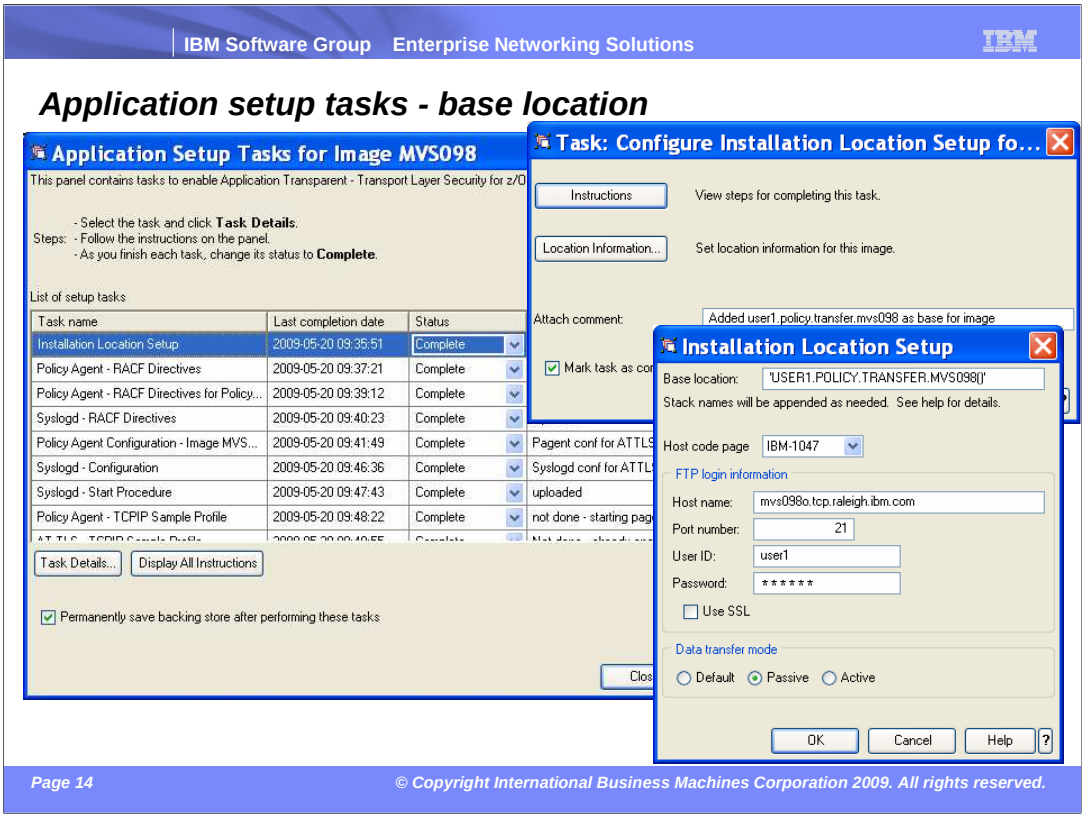
The NetView command interface will convert your input to uppercase by default. To preserve the case, use the netvasis command in front of EZACMD.



This slide shows how you can work with the Configuration Assistant and a z/OS LPAR. The example uses MVS PDS(E) libraries for all policy components, and it also uses z/OS for storing the backing store file. The backing store file is stored as a sequential MVS data set, while the policy configuration data components are stored as members of PDS(E) libraries.

The example uses two stacks (TCPCS and TCPCS2) on an LPAR (image name MVS098).

The example shows a suggested structure for honoring traditional change management procedures using 3 data sets. The transfer library set into which the Configuration Assistant stores new and changed definitions. The production library set from which the system reads the current definitions. And the backout library set for use in case of emergency backout of changed definitions. The simple method for production cutover is first to copy the content of the production library set to the backout library set. Then copy the transfer library set to the production library set. Either restart the policy components or issue modify commands to refresh their configurations.



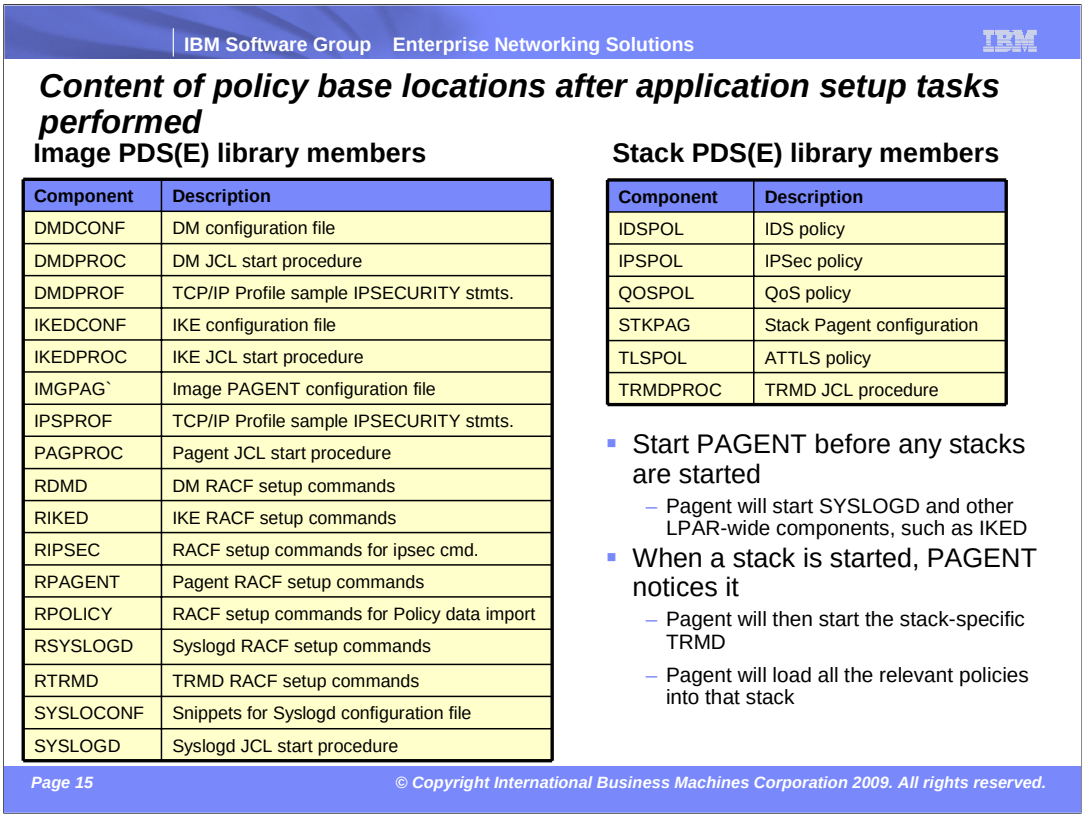
The Installation Location Setup panel is where you set a base location for the staging area. When you provide a base location for each image, the Configuration Assistant will use this to create names for the files. The files are organized by image name and then by stack name for files that are specific to a stack.

These file names can then be used when the configuration materials are transferred to the target host. You can override the provided file names at that time, but this base location provides a suggested set of names for you to use.

The base location can be a zFS file path, a dataset HLQ, or a partitioned dataset name. Whatever base location is specified, the administrator user ID needs the authority to write files at that location.

The Location Information panel also provides a place to set FTP login information to be used by the FTP delivery process. This FTP login information is the default for delivering files for the current image. At the final delivery stage, the administrator can override the file name, the installation method, or any of the FTP login details.

The Configuration Assistant will assume PDS(E) libraries have been created before using them. If the example LPAR TEST9 has a stack named STACK1 then two PDS(E) libraries must exist: one named USER1.CFGASST.TEST9 and one named USER1.CFGASST.TEST9.STACK1. If the LPAR has multiple stacks, a PDS(E) library per stack must be created.



This example has policies defined for IPSec, ATTLS, QoS, and IDS.

After defining all the policy disciplines and policies in the Configuration Assistant and performing all the suggested installation setup tasks, the sample PDS(E) libraries will contain several members. The names used here are the default names – they can be overridden.

There is still a little setup work to do on z/OS after defining everything and transferring everything to z/OS:

The RACF jobs need to be analyzed and executed.

The JCL procedures need to be copied to a valid procedure library.

The IMGPAG and STKPAG configuration files need to be checked again and modified as needed.

Configuration Assistant for z/OS Communications Server

- Windows-based stand-alone version can be downloaded from the Web:
  - Versions for z/OS V1R7, V1R8, V1R9, V1R10, and V1R11 are available for download
  - Support is “informal” (forum)
- Configuration Assistant for z/OS V1R11 Communications Server is also shipped with the z/OS MF product
  - Runs on z/OS
  - Accessed from a Web browser
  - Normal IBM support channels

The IBM Configuration Assistant for z/OS V1R11 Communications Server is on the Web at http://www.ibm.com/support/docview.wss?rs=852&context=SSSN3L&dc=D400&uid=swg24013160&loc=en_US&cs=UTF-8&lang=en&rss=ct852other

Or you can use this URL instead: http://tinyurl.com/cgoqsa

The Configuration Assistant in z/OS V1R11 is delivered both as a windows-based stand-alone application and with the new z/OS Management Facility offering on z/OS. The z/OS MF version is an IBM product fully supported by the normal IBM support channels.

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

NetView RACF z/OS

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

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

Other company, product, or 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 or changes in the products or programs 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.

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 (for example, 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.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.