



IBM SOA

WebSphere Security

What's New in v6.1



Alasdair Nottingham

WebSphere v6.1 – Philosophy

- ***Secure by Default, Secure out of the Box***
 - ▶ ***Administrative Security is enabled by default***
 - ▶ ***Most subsystems default to a reasonable level of authentication, authorization and encryption***
 - ▶ ***Certificates automatically generated and managed by default***

 - ▶ ***Some areas can still be improved***
 - ***See Security Hardening Presentation***

Agenda

- ***Enabled by Default***
- *Federated Repositories*
- *Keys and Certificates*
- *SPNEGO*
- *Fine Grained Access Control*
- *Portlet URL Security*

WebSphere v6.1 Security Features

- ***Security Enabled By Default***

- ✓ Enabled during install and profile creation to protect administrative resources. Option to disable.
- ✓ File-based repository used by default. Stores primary administrative identity and password as the initial user. Specified during install/profile creation or security enablement (if not enabled during install).
- ✓ Administrative security is split from application security enablement. Must have administrative security enabled before enabling application security. Application security enables J2EE container authentication and authorization. Administrative security protects other resources including Naming, MBeans, System Applications, etc.
- ✓ LTPA keys, self-signed certificate, keystores, server default identity are all generated automatically during profile creation.

On install...

- Administrative security is enabled.
 - ▶ Administrative security represents protection of system applications such as MBeans, scripting, AdminConsole, FileTransferServlet, Naming, etc. Other resources may still have some protection requirements to keep the WebSphere infrastructure protected. Application security is left disabled.
 - ▶ Must still enable application security and specify authorization constraints to have application security for the your applications (EJB and Web J2EE containers).
 - ▶ Creates the primary administrative user in the built-in File-Based repository.
- LTPA keys, self-signed certificate (one for the entire profile), default key and trust stores, and server identity will be automatically created.

Warning...

- It is possible to **not** enable administrative security on install
 - ▶ Please do not do this. There is no benefit.
 - ▶ No, really. Please do not do this. There is no benefit.
- Migration will preserve the state of security from the previous configuration.
 - ▶ This means that if your previous environment was mis-configured, your new environment will also be mis-configured.
 - ▶ We suggest that you do **not** use the migration tooling for this reason
 - Instead perform a new WAS 6.1 installation and manually deploy your applications to the new environment
 - This is a good idea in general – use “swing kit” for the new environment, and if something goes wrong you can easily fall back to the old environment!

Panel: Security Enablement in Profile Tool

Profile Management Tool

Administrative Security

Choose whether to enable administrative security. To enable security, supply a user name and password for logging into administrative tools. This administrative user is created in a repository within the application server. After profile creation finishes, you can add more users, groups, or external repositories.

Enable administrative security

User name: pbirk

Password: *****

Confirm password: *****

See the information center for more information.
[Online information center link](#)

Enables administrative security only (protects System applications). Application security must still be enabled post-install.

LTPA keys, self-signed certificate, and key and trust stores are automatically created during profile creation. Even when security is not enabled here.

< Back Next > Finish Cancel

Panel: Secure administration, applications, and infrastructure

The screenshot displays the 'Security Configuration Wizard' and 'Security Configuration Report' panels. The wizard is divided into several sections: Administrative security, Application security, Java 2 security, and User account repository. The Administrative security section has 'Enable administrative security' checked. The User account repository section shows 'Federated repositories' selected in the 'Available realm definitions' dropdown. The Authentication section on the right includes options for 'Web security', 'RMI/IIOP security', and 'Java Authentication and Authorization'. Callouts provide additional context: one notes that administrative security must be enabled before application or Java 2 security; another highlights the 'New Federated Repositories' selection for searching multiple registries; and a third points to the 'Authentication mechanisms and expiration' link, which is used for LTPA or SWAM configuration.

Wizard simplifies security configuration steps for first-time users.

Report displays security configuration attributes and settings in a table format.

Must enable administrative security before application security or Java 2 Security can be enabled.

New Federated Repositories selection which allows multiple File-Based and LDAP registries to be searched.

Configuration of LTPA or SWAM (deprecated).

Panel: Web Authentication General Settings

[Secure administration, applications, and infrastructure](#) > Web security - General settings

Specifies the settings for Web authentication.

Configuration

General Properties

Web authentication behavior

- Authenticate only when the URI is protected
- Use available authentication data when an unprotected URI is accessed
- Authenticate when any URI is accessed

Default to basic authentication when certificate authentication for the HTTPS client fails

Apply

This allows the default behavior to always authenticate web requests whether they are protected or not.

In cases where CERT_AUTH is the configured authentication mechanism, this option allows a fallback to basic auth if a client certificate is not present.

This is the default J2EE behavior.

This option persists the identity (if one exists in the request) and the URI is unprotected.

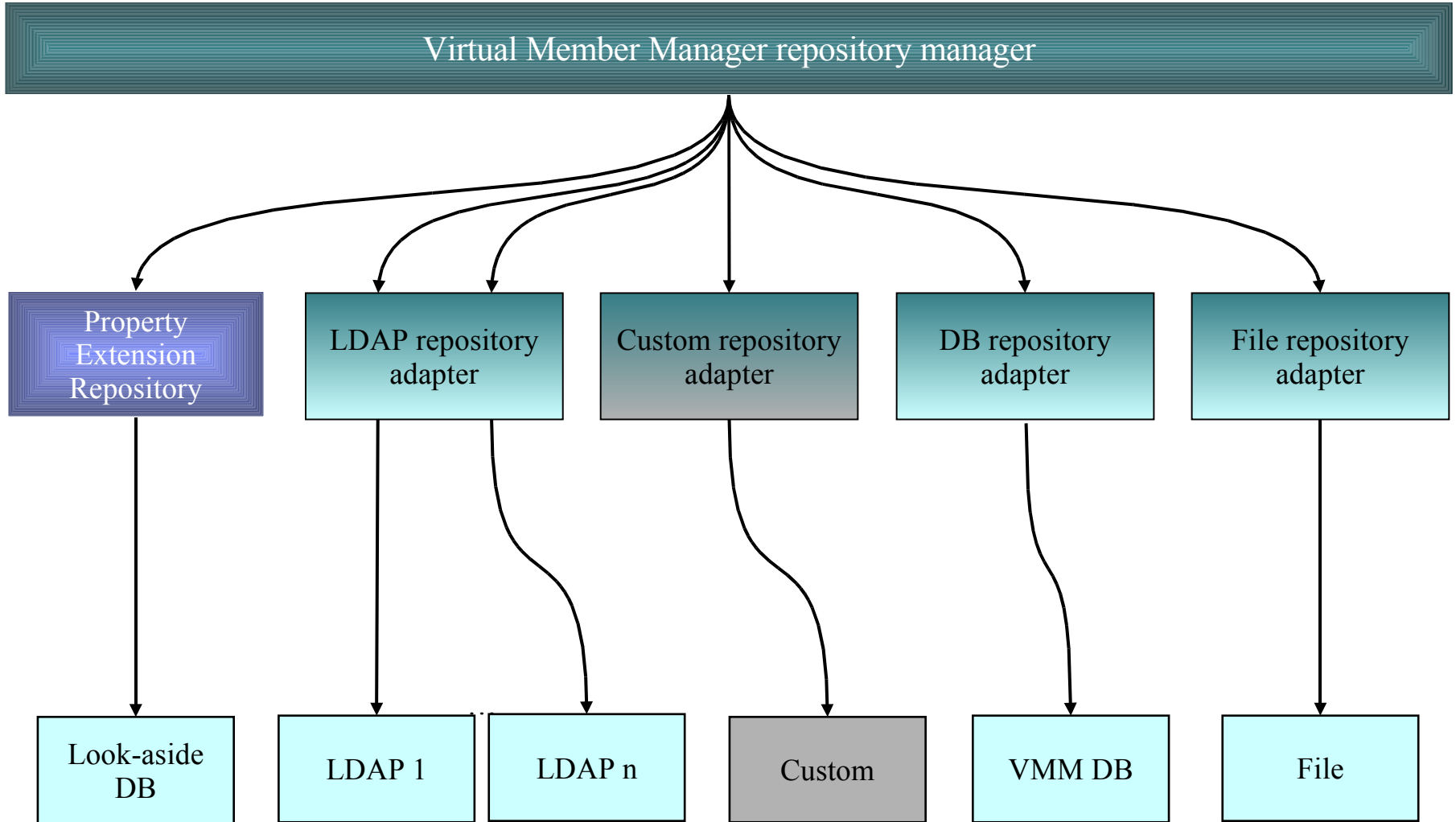
Agenda

- *Enabled by Default*
- ***Federated Repositories***
- *Keys and Certificates*
- *SPNEGO*
- *Fine Grained Access Control*
- *Portlet URL Security*

The Virtual Member Manager

- VMM is a New Feature in WAS 6.1 supporting Federated Repositories
- Prior to this release, only a single registry per cell, consisting of either:
 - ▶ A single LDAP directory
 - ▶ A single implementation of the Custom Registry interface
 - ▶ A local OS Registry (not supported for ND)
- Virtual Member Manager supports multiple individual repositories combined into a cell-wide federation.
- Three types of repository are supported in a Federation
 - ▶ File Based Repository (default)
 - ▶ Database Repository (using provided extensible schema)
 - ▶ One or more LDAP directories (or LDAP sub-trees). As always, LDAP is preferred.
- Custom repositories are supported as of 6.1.0.9 but not covered here
 - ▶ <http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/>
 - ▶ <http://www.ibm.com/developerworks/websphere/library/samples/vmms>

VMM



User and Group Management

- WAS Federated Repository support provides the ability to manage users and groups – i.e. provides read and write access to the federated repository.
- Admin console panels for searching across all configured repositories
- Can only add and update members from one repository in the federation
 - ▶ By default, this is the File Repository
 - ▶ Can be configured to manage another repository, such as a Database repository or an LDAP directory
- There are also other ways to manage users and groups:
 - ▶ Using wsadmin commands
 - ▶ Using programmatic APIs

Virtual Realms

- In previous versions of WAS, Security Realm mapped one-to-one with User Registry
- E.g. for an LDAP directory the Realm was identified by Host and Port
- In WAS 6.1 we have concept of a Virtual Realm
 - ▶ No longer maps to a single repository
 - ▶ Realm spans all repositories in Federation
 - ▶ Realm name is defined within WAS
 - ▶ Each repository in the federation is mapped to a “base entry” within the federated repository, providing a single logical namespace
 - ▶ In WAS 6.1 only a single virtual realm is supported in WAS
 - Multiple realms are expected to be supported in a future release

Realm Structure

- A Realm is a collection of independent Repositories. Each repository has:
 - ▶ A Repository Name, which can be anything you choose
 - ▶ A Base Entry in the Realm. This is a logical root entry for this particular repository within the federated repository (virtual realm)
 - This is the name in VMM for this repository
 - ▶ A Base Entry in the Repository. Optionally, the repository that is being federated may actually be a sub-tree of some other directory, so this defines the starting point, or root entry for the set of items that are part of the virtual realm.
 - E.g., for LDAP this is the baseDN in LDAP (where searches are rooted)
 - ▶ Example
 - Base entry in realm: `o=ldap,o=vmm`
 - Base entry in LDAP repository: `dc=example.com`
 - User in LDAP with DN of `uid=bob,ou=users,dc=example.com`
 - DN in VMM is then `uid=bob,ou=users,o=ldap,o=vmm`
 - if we were to change the base entry in LDAP to be `ou=users,dc=example.com`, then the VMM DN would be `uid=bob,o=ldap,o=vmm`

Federated Repository Searching

- All Repositories in the Federation are searched sequentially
 - ▶ User identities (User ID and DN) must be unique across all repositories within the federation
 - ▶ If a User ID is found in more than one repository, a Duplicate ID error will be thrown
 - ▶ It is not safe to make assumptions as to the order in which the repositories will be searched

- Groups are generally defined at the individual repository level, not the Federation level. Obtaining members of a group will be performed within the specific repository in which the group is defined.

- Performance for repository access may be improved by configuring an Entry Mapping Repository, but this does not have any impact on the search that is performed across the entire federation at login time.

Limitations

- Single Virtual Realm per cell
 - ▶ Federated Repository defined at cell level – shared by all applications
 - ▶ Cannot have both a Federation and a standalone registry active at the same time

- All repositories in the federation must be active and available in order for user authentication to be possible

- User IDs must be unique across all repositories in the federation

Mixed Cells

- Some limitations on mixed cells (i.e. cells including pre WAS 6.1 servers)
 - ▶ Can only configure one LDAP
 - ▶ Must be an LDAP type supported natively by older WAS version
 - ▶ Realm name must be LDAP host and port
 - ▶ Must configure stand-alone LDAP registry as well as the federation, with identical information. This is the information that will be replicated to back-level nodes.

- Given these limitations and the error-prone configuration requirements, there is probably no good reason to attempt to use federated repository in a mixed cell.

Panel: Federated Repositories Configuration

Single realm name for entire cell even though multiple repositories can be configured.

* Realm name
defaultWIMFileBasedRealm

* Primary administrative user name
pbirk

Server user identity

Automatically generated server identity

Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node

Password

Ignore case for authorization

Automatically generated server identity is an LTPA token protected identity for internal system calls. This is the default selection in v6.1.

Must use this server ID and password choice for mixed-version Cells due to previous version requirements. Password is optional though for v6.1.

Repositories in the realm:

Select	Base entry	Repository identifier	Repository type
<input type="checkbox"/>	o=defaultWIMFileBasedRealm	InternalFileRepository	File

- Additional Properties**
- Property extension repository
 - Entry mapping repository
 - Supported entity types
- Related Items**
- Manage repositories

Used to store attributes about users that cannot be stored in existing repositories.

Supported entity types must be configured before you can write to this repository via Manage Users/Groups.

Supports entry-level join of multiple repositories.

List of File-based and LDAP registries configured. Database registry may be configured through scripting.

Panel: Adding a Repository

General Properties

* Repository identifier
chiawang.austin.ibm.com:389

LDAP server

* Directory type
IBM SecureWay Directory Server

* Primary host name Port
chiawang.austin.ibm.com 389

Failover server used when primary is not available:

Delete		
Select	Failover host name	Port
	None	

Add

Support referrals to other LDAP servers
ignore

Security

Bind distinguished name

Bind password

Login properties
uid

Certificate mapping
EXACT_DN

Certificate filter

Require SSL communications

Centrally managed
Manage endpoint security configurations

Use specific SSL alias
CellDefaultSSLSettings SSL configurations

Typical LDAP configuration attributes.

Failover support.

Whether to ignore or follow referrals to other LDAP servers.

Additional Properties

- Performance
- LDAP entity types
- Group attribute definition

LDAP performance settings.

Group attributes

Centrally managed or direct selection for SSL configuration.

Panel: Supported LDAP Server List

General Properties

* Repository identifier

LDAP server

* Directory type

- IBM SecureWay Directory Server
- IBM SecureWay Directory Server**
- * IBM Tivoli Directory Server Version 4
- IBM Tivoli Directory Server Version 5.1
- IBM Tivoli Directory Server Version 5.2
- IBM Tivoli Directory Server Version 6
- z/OS Integrated Security Services LDAP Server
- IBM Lotus Domino Version 5
- IBM Lotus Domino Version 6
- IBM Lotus Domino Version 6.5
- Novell Directory Services
- Sun ONE
- Microsoft Windows 2000 Server Active Directory
- Microsoft Windows Server 2003 Active Directory
- Microsoft Active Directory Application Mode
- Custom

Support referrals to other LDAP servers

ignore

Security

Bind distinguished name

Bind password

Login properties

Certificate mapping

EXACT_DN

Certificate filter

Require SSL communications

Centrally managed

[Manage endpoint security configurations](#)

Use specific SSL alias

CellDefaultSSLSettings

[SSL configurations](#)

Panel: Manage Users (for File-based Repository)

Manage users and groups for federated registries.

Search for Users

Search by * Search for * Maximum results

1 users matched the search criteria.

Select	User ID	First name	Last name	E-mail	Unique Name
<input type="checkbox"/>	pbirk	Peter	Birk	pbirk@us.ibm.com	uid=pbirk,o=defaultWIMFileBasedRealm

Page 1 of 1 Total: 1

Agenda

- *Enabled by Default*
- *Federated Repositories*
- ***Keys and Certificates***
- *SPNEGO*
- *Fine Grained Access Control*
- *Portlet URL Security*

Keys and Certificates

- **KeyStore and Certificate Management**

- ▶ No longer using the dummy certificate!
 - Self-signed certificate created per profile (properties to define values).
- ▶ IKeyMan-like certificate management capabilities built into Admin Console.
- ▶ Removed JVM system property requirement for SOAP and URL connections using SSL
- ▶ More detailed error messages for SSL handshake failures
- **Warning:** Many of the new certificate management features apply only to newly created cells. Cells and nodes that are migrated from previous versions will not benefit from many of these

Keys and Certificates

- **SSL Configuration Management**
 - ▶ Dynamic SSL configuration updates (all outbound, CFW inbound)
 - ▶ Pluggable key and trust manager support
 - ▶ Support for certificate revocation checking using PKIX trust manager
 - ▶ Multiple SSL configuration selection types with precedence rules
 - Programmatic selection (thread-based)
 - Dynamic outbound selection (outbound protocol, target host and port)
 - Direct selection (for backwards compatibility)
 - Scoped selection (centrally managed)

Keys and Certificates

- **KeyStore and Certificate Management (server-side)**
 - Federation of Base AppServer exchanges signer with Deployment Manager
 - Common TrustStore used by default in Cell
 - Webserver plugin CMS (.kdb) key store automatically maintained in config repository.
 - Expiration monitoring with notification and auto-replacement of expiring self-signed certificates
 - Hierarchically scoped SSL and KeyStore configurations

- **KeyStore and Certificate Management (client-side)**
 - Signer exchange prompt for easy exchange (browser-like, can disable prompt)
 - New SSL configuration properties file, ssl.client.props, supports multiple SSL configurations
 - Same SSL configuration support as server-side, including pluggable trust/key managers

Keys and Certificates

- ***Key Lifecycle Management***
 - ✓ Generic key and key pair generator interfaces
 - ✓ Automatically deletes old keys when new key generated and maximum number of managed keys reached
 - ✓ LTPA implements key set and key set groups to manage auto-generation of LTPA keys
 - ✓ Keys stored in keystores (can be hardware keystore)

Key Stores

- Key stores & trust stores contain
 - ▶ Signers
 - ▶ Personal certificates – certificate + corresponding private key
 - ▶ Personal certificate requests
 - ▶ By convention trust stores contain only signers while key stores contain only personal certificates
 - ▶ There are *usually* no default signers. Exceptions are “compatibility signers” (dummy signers for earlier versions of WAS), and CMS files still have all the CA's (should be removed)
- Storage types are
 - ▶ JCEKS, JKS – Java file formats
 - ▶ PKCS12 – standard file format (.p12). Microsoft may call this PFX.
 - ▶ PKCS11 – hardware device (path is actually the location of the shared library to load)
 - ▶ CMSKS – older format used by IHS and plugin (.kdb)
- WAS provides all the needed key stores by default
- Key Stores and Trust Stores (even KDB files) are managed via the WAS admin console and stored in the configuration repository.
- Changes are automatically replicated throughout the cell

SSL configuration

- New configuration model for SSL.
- Centrally managed option instead of specifying SSL alias all over the place.
- Integration of SSL runtime with SSL configuration for everything SSL (including URLs, Admin SOAP, Bus, IIOp, HTTP, etc).
- Ability to determine visibility of configuration by management scope. By default, Cell scope
- Certificate expiration monitoring capabilities.
 - ▶ Can replace self-signed certificates once they reach threshold of expiration.
 - ▶ Replace function will replace signers that exist in keystores throughout the entire Cell configuration.
- Warning: Be careful with changes made when “dynamic” updates are enabled. This can cause immediate outages if not careful, should test changes in test environment before making them in production.
- Note: Admin SOAP inbound and IIOp inbound (on distributed) do not accept dynamic changes. This is due to their use of socket factories instead of the channel framework (which can dynamically change SSL settings for inbound without affecting existing connections).

SSL Configurations

- SSL Configurations contain/specify
 - ▶ Key store – private key info
 - Certificate aliases – which private key/cert pair should be used from this store
 - ▶ Trust store – trusted signers
 - ▶ Trust manager – manages validating certificates from peer
 - ▶ Key manager – manages getting keys as needed
 - ▶ Quality of Protection – SSL strength, etc.
- IBM provides two trust managers
 - ▶ IbmX509 (the default) – checks certificate signature and expiration. Optionally can require hostname verification for certificates on URL connections.
 - ▶ IbmPKIX – same plus more
 - CRL distribution point processing – contact CA for revocation based on info in certificate
 - Optionally supports Online Certificate Status Protocol (OCSP)
 - ▶ You can write your own trust manager if you want
 - You could implement your own arbitrary complex validation rules
- IBM provides one key manager
 - ▶ IbmX509
 - ▶ You can write your own key manager if you want

Default SSL Configurations

- WAS creates a number of default SSL configurations
 - ▶ Each node has (by default) a node specific SSL configuration: NodeDefaultSSLSettings
 - ▶ The cell has a cell default SSL configuration: CellDefaultSSLSettings
- You can create your own SSL configurations if needed
 - ▶ Perhaps you'll dedicate an SSL configuration for the LDAP server, or a Web Container, or just a particular application server

Panel: SSL, KeyStore, and Certificate Management

The screenshot displays the IBM SOA console interface. On the left is a navigation pane with a 'View: All tasks' dropdown and a tree structure including 'Welcome', 'Guided Activities', 'Servers', 'Applications', 'Resources', 'Security' (with sub-items like 'Secure administration, applications, and infrastructure', 'SSL certificate and key management', 'Bus Security', 'Web services'), 'Environment', 'System administration', 'Users and Groups', 'Monitoring and Tuning', 'Troubleshooting', 'Service integration', and 'UDDI'. The main content area is titled 'SSL configurations' and contains text explaining the Secure Sockets Layer (SSL) and how configurations are managed. Below this is a 'Configuration settings' section with two options: 'Use the United States Federal Information Processing Standard (FIPS) algorithms...' (unchecked) and 'Dynamically update the run time when SSL configuration changes occur' (checked). On the right, there is a 'Related Items' list with links to 'SSL configurations', 'Dynamic outbound endpoint SSL configurations', 'Key stores and certificates', 'Key sets', 'Key set groups', 'Key managers', and 'Trust managers'. Three callout boxes are overlaid on the image: one pointing to the 'SSL configurations' text, one pointing to the 'Dynamically update...' checkbox, and one pointing to the 'Key stores and certificates' link.

View: All tasks

- Welcome
- Guided Activities
- Servers
- Applications
- Resources
- Security
 - Secure administration, applications, and infrastructure
 - SSL certificate and key management
 - Bus Security
 - Web services
- Environment
- System administration
- Users and Groups
- Monitoring and Tuning
- Troubleshooting
- Service integration
- UDDI

SSL configurations

The Secure Sockets Layer (SSL) secures communications between processes or endpoints. SSL is used for establishing communications for inbound and outbound from an endpoint. For secure communications, a certificate configuration must be specified.

In previous versions of this product, it was necessary to manually configure each endpoint for Secure Sockets Layer (SSL). In this version, you can define a single configuration for the entire serving environment. This allows you to centrally manage security configurations. In addition, trust zones can be defined for node environments by using a single level SSL configuration.

If you have migrated from a previous version using the migration tool, you may need to re-configure SSL configurations for centralized management.

Configuration settings

- [Manage endpoint security configurations](#)
- [Manage certificate expiration](#)

Use the United States Federal Information Processing Standard (FIPS) algorithms. Note: This option requires the TLS handshake protocol, which some browsers do not enable by default.

Dynamically update the run time when SSL configuration changes occur

Related Items

- [SSL configurations](#)
- [Dynamic outbound endpoint SSL configurations](#)
- [Key stores and certificates](#)
- [Key sets](#)
- [Key set groups](#)
- [Key managers](#)
- [Trust managers](#)

Cell-scoped configurations.

Show topology view for finer-grained configuration scopes.

Periodic task monitoring for certificate expiration.

Careful: Dynamically updates the runtime after changes saved and sync'd.

Panel: EndPoint SSL Selection Choice

SSL settings

Centrally managed

- [Manage endpoint security configurations](#)

Use specific SSL alias

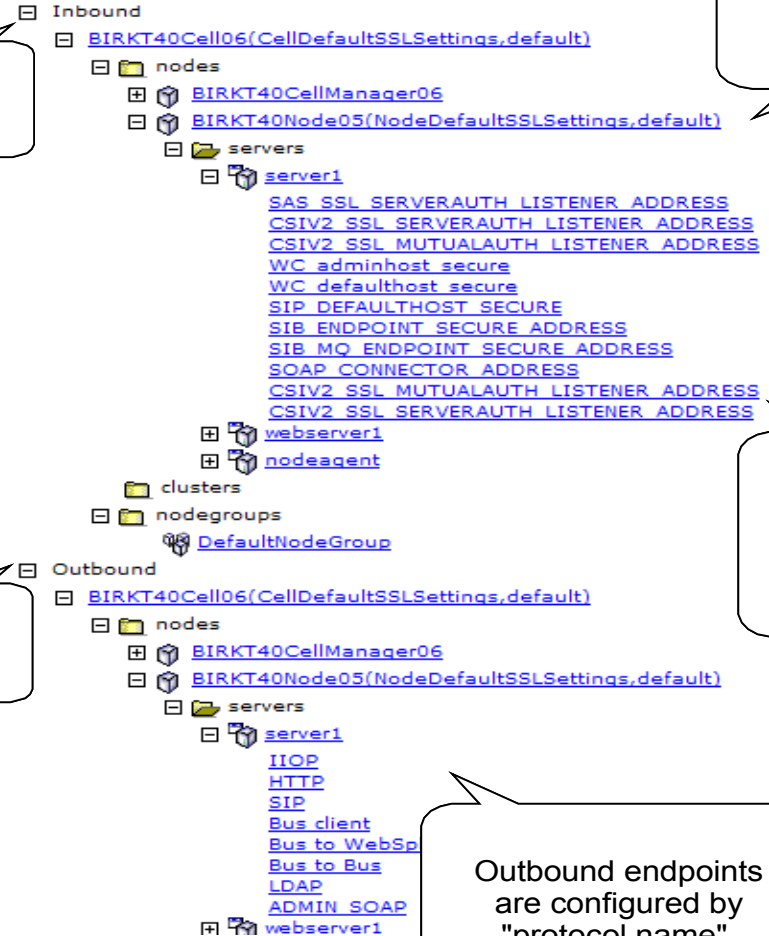
▼

- [SSL configurations](#)

- Allows choice at any EndPoint configuration to either make a direct selection (as done in previous releases, and by migration) or choose to Centrally Manage the SSL configurations (default in v6.1).
- It's highly recommended to encourage the movement of Centrally Managed configurations for ease-of-maintenance. Changes in SSL configurations can be done much more rapidly from one location which only affects the security.xml. The JSSEHelper knows how to interpret these changes in runtime, when called from a particular scope.

Panel: SSL Configurations Scoped by Topology

Inbound SSL configurations.



Outbound SSL configurations.

Node association overrides Cell association. Everything below the Node, inherits this SSL configuration.

Specific endpoints for "server1". Note: No associations currently made here, inherits from Node.

Outbound endpoints are configured by "protocol name".

Topology view

- Topology view is for the Centrally Managed option. Endpoints can configure themselves as either Centrally Managed (default in v6.1) or Direct Selection (default when migrated).
- Topology view is how to scope the visibility of SSL, keystore, keyset, etc. configurations to a specific Node, NodeGroup, Cluster, Server, or EndPoint. When associated with one of these management scopes, visibility is limited to that scope and below.
- Be aware, a configuration created at Node A will not show up in the list for Node B or Cell scopes, etc. This may cause some confusion at first, but is working as designed.
- The SSL configuration chosen is the one defined at the lowest scope for the EndPoint requesting the SSL configuration.

Panel: Override SSL Configuration at EndPoint Scope

[SSL certificate and key management](#) > [Manage endpoint security configurations](#) > WC_adminhost_secure

Displays Secure Sockets Layer (SSL) configurations for selected scopes, such as a cell, node, server, or cluster.

Configuration

General Properties

Name

WC_adminhost_secure

Direction

Inbound

Inherited SSL configuration

Inherited SSL configuration name

NodeDefaultSSLSettings

Inherited certificate alias

default

Specific SSL configuration for this endpoint

Override inherited values

SSL configuration

NodeDefaultSSLSettings

Update certificate alias list

Manage certificates

Certificate alias in key store

test certificate

Endpoint name.

Inherited SSL configuration and certificate alias.

Configure objects visible only to this particular endpoint.

Override above settings from Node scope with these settings for the particular endpoint.

Related Items

- [SSL configurations](#)
- [Dynamic outbound endpoint SSL configurations](#)
- [Key stores and certificates](#)
- [Key sets](#)
- [Key set groups](#)
- [Key managers](#)
- [Trust managers](#)

SSL configuration override

- When you click on a scope link, it brings you to this panel. It shows you the “inherited SSL configuration” from a higher up scope.
- To override, click the “Override inherited values” check box and select a new SSL configuration.
- You must selection the “Update certificate alias list” push button before you can save (unless you leave it as ‘(none)’).
- Related items links will create new configuration objects at this particular management scope to reduce visibility.

Panel: Dynamic Outbound SSL Configuration Association

[SSL certificate and key management](#) > [Manage endpoint security configurations](#) > [server1](#) > [Dynamic outbound endpoint SSL configurations](#) > [New](#)

Dynamic endpoint configuration scopes represent an association between an Secure Sockets Layer (SSL) configuration and target protocol, host, and port. When an outbound connection is attempted, this association is verified ahead of the SSL configuration scope association. Based on the protocol, host, port target, the outbound SSL configuration might be different than the default that is specified in the SSL scope configuration

Configuration

General Properties

* Name

Description

Connection information

Add connection information

*

SSL configuration

Certificate alias

Related Items

- [SSL configurations](#)

Any http or iiop requests to the domain of .ibm.com (any port) will use the SSL configuration and certificate alias configured below..

Dynamic selection

- This overrides both the “Centrally Managed” and “Direct Selection” methods for configuration. This is a “Dynamic Selection” method for configuring SSL. It dynamically associates an SSL configuration to outbound connection information such as target host and port or the outbound protocol type.
- To avoid overriding other methods of SSL selection in all servers, you may want to scope this change to the server or endpoint that absolutely needs it.

Panel: Keystore Collection

Select two keystores and exchange signers.

Management > Manage endpoint security configurations > BIRKT40Node05 > Key stores and certificates

including cryptography, RACF(R), CMS, Java(TM), and all TrustStore types.

Select	Name	Path	Remotely managed	Host list
<input type="checkbox"/>	CellDefaultKeyStore	\${CONFIG_ROOT}/cells/BIRKT40Cell06/key.p12	false	
<input type="checkbox"/>	CellDefaultTrustStore	\${CONFIG_ROOT}/cells/BIRKT40Cell06/trust.p12	false	
<input type="checkbox"/>	CellLTPAKeys	\${CONFIG_ROOT}/cells/BIRKT40Cell06/ltpa.jceks	false	
<input type="checkbox"/>	NodeDefaultKeyStore	\${CONFIG_ROOT}/cells/BIRKT40Cell06/nodes/BIRKT40Node05/key.p12	false	
<input type="checkbox"/>	NodeDefaultTrustStore	\${CONFIG_ROOT}/cells/BIRKT40Cell06/nodes/BIRKT40Node05/trust.p12	false	

Total 5

Default keystores are managed in the configuration repository and synchronized to Nodes.

Panel: Exchange Signers

[SSL certificate and key management > Manage end stores and certificates > Exchange signers](#)

[RKT40Node05 > Key](#)

Extract a Personal Certificate from one key store and

Signer Certificate.

Extract personal certificate and move over to the other keystore as a signer.

Configuration

General Properties

Signers to exchange

<p>CellDefaultKeyStore personal certificates</p> <ul style="list-style-type: none"> default test certificate 	<p>Add >></p> <p><< Remove</p>	<p>NodeDefaultTrustStore signers</p> <ul style="list-style-type: none">
<p>NodeDefaultTrustStore personal certificates</p> <ul style="list-style-type: none"> 	<p>Add >></p> <p><< Remove</p>	<p>CellDefaultKeyStore signers</p> <ul style="list-style-type: none">

Apply OK Reset Cancel

Panel: KeyStore (Managed on the Node)

[SSL certificate and key management](#) > [Key stores and certificates](#) > [HardwareKeyStore](#)

Defines KeyStore types, including cryptography, RACF(R), CMS, Java(TM), and all TrustStore type

Configuration

General Properties

Name
HardwareKeyStore

Path
c:\mydevice\luna.cfg

Change password
••••••

Confirm password
••••••

Type
Cryptographic Token Device (PKCS11)

Remotely managed

Host list
host1.austin.ibm.com

Read only

Initialize at startup

Additional Properties

- [Signer certificates](#)
- [Personal certificates](#)
- [Personal certificate requests](#)
- [Custom properties](#)

Keystore type.

Certificate management links.

Remotely managed indicates the keystore physically resides on a Node. An MBean request is sent for the certificate management updates.

Hardware devices used for acceleration would benefit from immediate initialization.





Panel: Personal Certificate Collection

[SSL certificate and key management](#) > [Key stores and certificates](#) > [CellDefaultKeyStore](#) > [Personal certificates](#)

Manages personal certificates.

⊞ Preferences

Create a self-signed certificate Delete Receive a certificate from a certificate authority Replace Extract Import Export

Select	Alias ▾	Issued by ▾	Issued to ▾	Serial number ▾	Expiration ▾
<input type="checkbox"/>	default	CN=BIRKT40.austin.ibm.com, O=IBM, C=US	CN=BIRKT40.austin.ibm.com, O=IBM, C=US	1146156212	Valid from April 27, 2006 to April 27, 2007.
<input type="checkbox"/>	test certificate	CN=birkt40.austin.ibm.com, OU=WebSphere, O=IBM, L=Austin, ST=Texas, POSTALCODE=78751, C=US	CN=birkt40.austin.ibm.com, OU=WebSphere, O=IBM, L=Austin, ST=Texas, POSTALCODE=78751, C=US	1146403508	Valid from April 30, 2006 to April 30, 2007.

Total 2

- Same IKeyMan-like function except for the advanced “Replace” function. This will allow the selection of certificate to replace with a new one. It replaces all old signers. This is the same function used by the certificate expiration monitor to replace an expiring certificate.

General Properties

Old certificate
default

Replace with
test certificate ▾

Delete old certificate after replacement


Delete old signers





Apply OK Reset Cancel

Panel: Signer Certificate Collection

[SSL certificate and key management](#) > [Manage endpoint security configurations](#) > [server1](#) > [Key stores and certificates](#) > [NodeDefaultTrustStore](#) > [Signer certificates](#)

Manages signer certificates in key stores.

 Preferences

Select	Alias 	Issued to 	Fingerprint (SHA digest) 	Expiration 
<input type="checkbox"/>	www.ibm.com	CN=www.ibm.com, OU=HPODS, O=IBM, L=Research Triangle Park, ST=NC, C=US	52:D1:45:73:04:E7:20:4A:0F:C1:9D:E4:91:86:F8:82:D6:64:8D:57	Valid from February 2, 2006 to May 5, 2007.
<input type="checkbox"/>	default	CN=BIRKT40.austin.ibm.com, O=IBM, C=US	16:5D:C0:5F:C3:29:7E:3B:4B:5E:9E:63:89:90:6D:F2:69:8E:27:13	Valid from March 8, 2006 to March 8, 2007.
<input type="checkbox"/>	dummyclientsigner	CN=jclient, OU=SWG, O=IBM, C=US	0B:3F:C9:E0:70:54:58:F7:FD:81:80:70:83:A6:D0:92:38:7A:54:CD	Valid from July 30, 2003 to October 13, 2021.
<input type="checkbox"/>	dummyserver signer	CN=jserver, OU=SWG, O=IBM, C=US	FB:38:FE:E6:CF:89:BA:01:67:8F:C2:30:74:84:E2:40:2C:B4:85:65	Valid from July 30, 2003 to October 13, 2021.
<input type="checkbox"/>	pete	CN=pete, OU=, O=ibm, L=, ST=, POSTALCODE=, C=US	EB:FE:7B:D2:19:DD:7B:09:17:EB:2C:94:31:22:E1:60:F6:5F:D4:94	Valid from March 8, 2006 to March 8, 2007.
<input type="checkbox"/>	default_1	CN=BIRKT40.austin.ibm.com, O=IBM, C=US	AB:89:87:D7:5E:FA:F2:B0:37:EF:58:40:CE:C0:E6:B1:5E:42:88:D6	Valid from April 6, 2006 to April 6, 2007.
Total 6				

Panel: Retrieving a Signer from Remote Port

[SSL certificate and key management](#) > [Manage endpoint security configurations](#) > [server1](#) > [Key stores and certificates](#) > [NodeDefaultTrustStore](#) > [Signer certificates](#) > Retrieve from port

Makes a test connection to a Secure Sockets Layer (SSL) port and retrieves the signer from the server during the handshake.

Configuration

General Properties

* Host

* Port
 Specifies the host name to which you connect when attempting to retrieve the signer certificate from the Secure Sockets Layer (SSL) port.

SSL configuration for outbound connection
 ▼

* Alias

- This allows the retrieval and storage of a signer from a remote SSL port. When connecting to an SSL port, the handshake sends the signer from the server's personal certificate into the trust manager, this allows us to store it during the connection attempt.

Panel: Scoped SSL Configuration

[SSL certificate and key management](#) > [Manage endpoint security configurations](#) > [server1](#) > [SSL configurations](#) > **NodeDefaultSSLSettings**

Defines a list of Secure Sockets Layer (SSL) configurations.

Configuration

General Properties

* Name

NodeDefaultSSLSettings

Trust store name

CellDefaultTrustStore

Keystore name

NodeDefaultKeyStore

Get certificate aliases

Default server certificate alias

pete

Default client certificate alias

pete

Management scope

(cell):BIRKT40Cell01:(node):BIRKT40Node06

Apply

OK

Reset

Cancel

Additional Properties

- [Quality of protection \(QoP\) settings](#)
- [Trust and key managers](#)
- [Custom properties](#)

Related Items

- [Key stores and certificates](#)

Scoped configurations

- Keystore and truststore references with ability to choose alias from keystore for client and server identity. When you choose a different keystore, you must refresh the alias lists by clicking on “Get certificate aliases” push button.
- Pluggable trust and key managers.
- Configurable IbmPKIX trust manager provides CRL validation when certificates have a CRL distribution point extension (common for most CAs).
- Management scope shown for SSL configurations. This determines visibility based on topological inheritance.

Panel: Certificate Expiration Monitor

Start now

General Properties

* Expiration notification threshold
60 days

Enable checking

Expiration checking

Scheduled time of day to check for expired certificates

21 : 30 A.M. P.M. 24-hour

Check by calendar

Weekday * Repeat interval
Sunday 4 weeks

Check by number of days

* Repeat interval
7 days

Specifies that you want to schedule a specific number of days between each run of the expiration monitor.

Next start date

Sunday, June 25, 2006 9:30 PM

Expiration check notification

MessageLog

Automatically replace expiring self-signed certificates

Delete expiring certificates and signers after replacement

Related Items

- [Notifications](#)

Expiration Monitor

- Checks certificates within all keystores in the Cell configuration.
- “Expiration notification threshold” determines how much lead time you want to be notified (default 30 days).
- Capability to “replace” self-signed certificates that meet this threshold.
- Replaces all the associated signer certificates with the expiring self-signed to preserve existing trust.
- Clients will need to re-import a server signer after it has been replaced, however, the signer exchange prompt and/or retrieveSigners script can handle this.
- Important Note: Some inbound endpoints do not perform dynamic certificate updates, including SOAP connector inbound and IOP inbound. The server needs to be recycled for this “Server Sockets” once the expired certificates are replaced. SSLInboundChannels do not have this restriction.

Panel: WebServer Plugin SSL Configuration

[Web servers](#) > [webserver1](#) > **Plug-in properties**

Use this page to configure a Web server plug-in. The plug-in passes HTTP requests from a Web server to WebSphere(R) Application Servers.

Runtime Configuration

Plug-in properties

Ignore DNS failures during Web server startup

* Refresh configuration interval
 seconds

Repository copy of Web server plug-in files:

- * Plug-in configuration file name
- Automatically generate the plug-in configuration file
- Automatically propagate plug-in configuration file
- * Plug-in key store file name

Additional Properties

- [Request and Response](#)
- [Caching](#)
- [Request Routing](#)
- [Custom Properties](#)

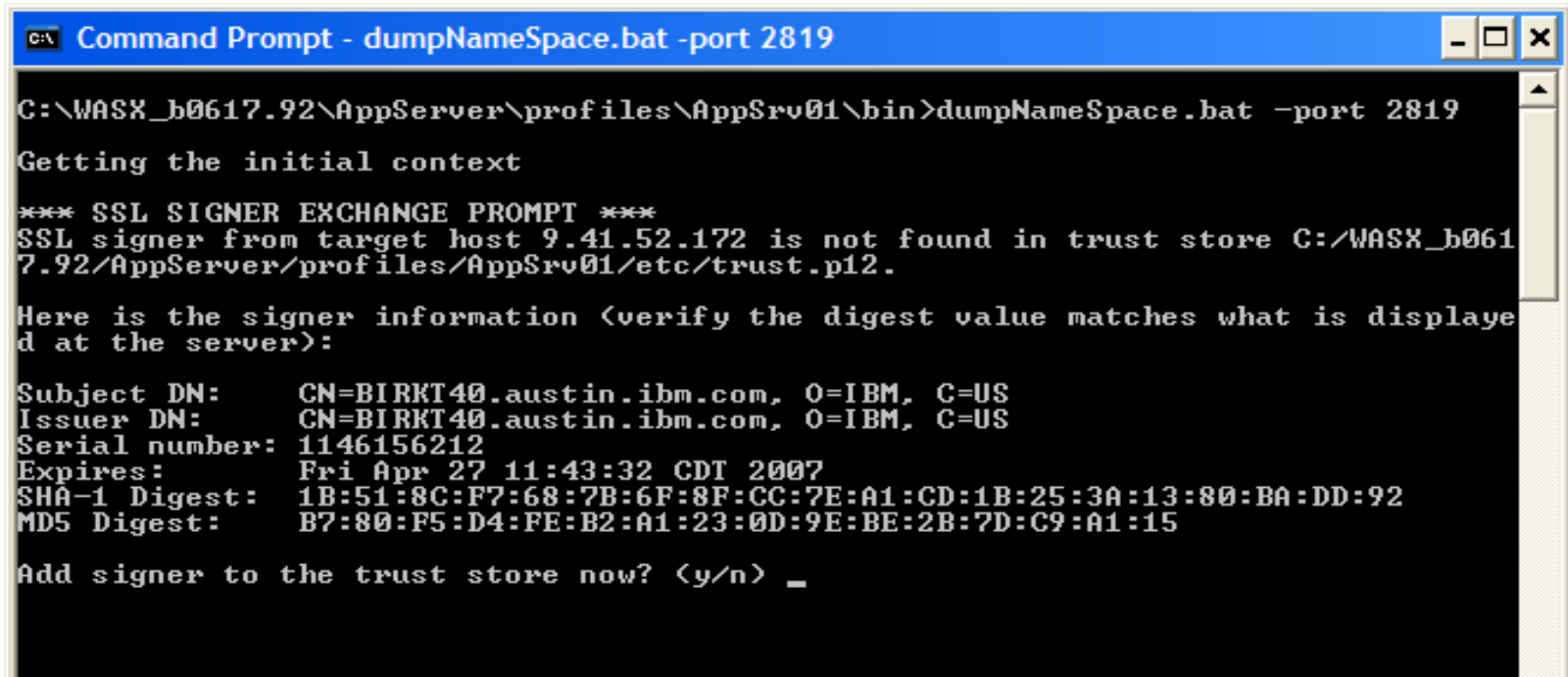
Plugin SSL configuration

- Plugin-key.kdb is generated during web server creation using the same self-signed from the profile. This enables trust on both sides of the connection between plugin and application server, out of the box.
- Can link to the same certificate management panels to modify the KDB file and then propagate to the WebServer similar to how the WebServer configuration file is propagated.
- iSeries has hard link to internal password (no .sth file) so once KDB is propagated, the password needs to be reset to establish the link.
- zSeries does not have writeable keystores, so we cannot add the signer from the self-signed in the KDB back into the RACF keystores. However we can extract the RACF signer into the KDB so that server authentication succeeds (which is the default setup).

Client Signer Management

- New `ssl.client.props` should be added to your scripts for the new configuration to be effective. SSL configurations in the `sas.client.props` and `soap.client.props` are still effective, but won't have the enhanced function unless properties are added.
- `RetrieveSigners` script the recommended method for downloading signers from the server to prepare clients. Programmatic techniques available to `retrieveSigners`, `autoAcceptForThisConnectionOnly`, `autoAcceptAndStoreInTrustStore` using the `com.ibm.wsspi.ssl.RetrieveSignersHelper` SPI.
- `RetrieveSigners` can be used for Cell->Cell interop by changing the client truststore (in `/etc`) to point the server truststore (in `/config/cells/<cellname>`) for the local truststore.
- Signer exchange prompt simplifies handshake failures by allowing real-time acceptance. This can cause hang-like symptoms for clients that perform background connections. In these cases, the signer exchange prompt should be disabled.

Signer Exchange Prompt



```
Command Prompt - dumpNameSpace.bat -port 2819

C:\WASX_b0617.92\AppServer\profiles\AppSrv01\bin>dumpNameSpace.bat -port 2819

Getting the initial context

*** SSL SIGNER EXCHANGE PROMPT ***
SSL signer from target host 9.41.52.172 is not found in trust store C:/WASX_b0617.92/AppServer/profiles/AppSrv01/etc/trust.p12.

Here is the signer information (verify the digest value matches what is displayed at the server):

Subject DN:      CN=BIRKT40.austin.ibm.com, O=IBM, C=US
Issuer DN:       CN=BIRKT40.austin.ibm.com, O=IBM, C=US
Serial number:   1146156212
Expires:        Fri Apr 27 11:43:32 CDT 2007
SHA-1 Digest:   1B:51:8C:F7:68:7B:6F:8F:CC:7E:A1:CD:1B:25:3A:13:80:BA:DD:92
MD5 Digest:     B7:80:F5:D4:FE:B2:A1:23:0D:9E:BE:2B:7D:C9:A1:15

Add signer to the trust store now? (y/n) _
```

- Enabled or disabled from the **com.ibm.ssl.enableSignerExchangePrompt** property in properties/ssl.client.props file.
- Note: Can cause hang-like symptoms when enabled in background threads and the signer exchange prompt is

RetrieveSigners Utility

```
Command Prompt

C:\WASX_b0617.92\AppServer\profiles\AppSrv01\bin>retrieveSigners.bat
CWPKI0300I: Use the -listRemoteKeyStoreNames and -listLocalKeyStoreNames
           options to get list of names for <remoteKeyStoreName> and
           <localKeyStoreName>, respectively.

Usage: retrieveSigners
       <remoteKeyStoreName> <localKeyStoreName> [options]
options:
  [-profileName <profileName>] [-remoteAlias <aliasFromRemoteStore>]
  [-localAlias <storeAsAlias>] [-listRemoteKeyStoreNames]
  [-listLocalKeyStoreNames] [-autoAcceptBootstrapSigner]
  [-uploadSigners] [-host <host>] [-port <port>] [-contentType
  <RMI!SOAP>] [-user <user>] [-password <password>] [-trace] [-logfile
  <filename>] [-replacelog] [-quiet] [-help]

C:\WASX_b0617.92\AppServer\profiles\AppSrv01\bin>retrieveSigners.bat CellDefault
TrustStore ClientDefaultTrustStore -autoAcceptBootstrapSigner
CWPKI0308I: Adding signer alias "CN=BIRKT40.austin.ibm.com, O=IBM, C=US" to
           local keystore "ClientDefaultTrustStore" with the following SHA
           digest: 1B:51:8C:F7:68:7B:6F:8F:CC:7E:A1:CD:1B:25:3A:13:80:BA:DD:92
CWPKI0308I: Adding signer alias "dummyclientsigner" to local keystore
           "ClientDefaultTrustStore" with the following SHA digest:
           0B:3F:C9:E0:70:54:58:F7:FD:81:80:70:83:A6:D0:92:38:7A:54:CD
CWPKI0308I: Adding signer alias "dummyserversigner" to local keystore
           "ClientDefaultTrustStore" with the following SHA digest:
           FB:38:FE:E6:CF:89:BA:01:67:8F:C2:30:74:84:E2:40:2C:B4:B5:65

C:\WASX_b0617.92\AppServer\profiles\AppSrv01\bin>_
```

Panel: Key Lifecycle Management - Key Set

[SSL certificate and key management](#) > [Key sets](#) > CellLTPAKeyPair

Manages key sets that control key instances of the same type for use in cryptographic operations.

Configuration

General Properties

* Key set name

* Key alias prefix name

* Key password

* Confirm password

Key generator class name



Delete key references that are beyond the maximum number of keys

* Maximum number of keys referenced

Key store



Generates key pair

Additional Properties

▪ [Active key history](#)

Related Items

▪ [Key stores and certificates](#)

KeySets

- A KeySet manages the lifecycle of a specific key type.
- Can be configured to dynamically generate keys when a configured key or key pair generation class is implemented.
- Manages the number of active keys to remember, wraps old keys off as new keys are generated.
- Uses keystore to store keys so hardware device may be used here.
- Can simply reference existing keys in a keystore instead of generating them in software and adding them into the keystore.
- `com.ibm.websphere.ssl.KeySetHelper` API helps applications obtain key or KeySet references. This API is Java 2 Security protected.

Panel: Key Lifecycle Management - Key Set Group

Configuration

General Properties

* Key set group name

Key sets

*

CellLTPAKeyPair
 CellLTPASecret

Related Items

- [Key sets](#)

Key generation

Automatically generate keys

Scheduled time for generation

:
 A.M.
 P.M.
 24-hour

Generate on a specific day

Weekday:
 Repeat interval: weeks

Generate at an interval

Repeat interval: days

Next start date:

KeySetGroups

- A KeySetGroup manages one or more KeySets.
- Enables dynamic generation of multiple KeySets to synchronize the key versions.
- Can generate based on a configurable schedule.
- Note: One can disable the automatic generation of keys. In some cases this is desirable since we may want to maintain a static list for some period of time.



Panel: Key Lifecycle Management – Active Key History

[SSL certificate and key management](#) > [Key sets](#) > [CellLTPAKeyPair](#) > Active key history

Manages key alias references in key stores.

+ Preferences

Add key alias reference Delete Generate key

Select	Alias reference
<input type="checkbox"/>	LTPAKeyPair_2
<input type="checkbox"/>	LTPAKeyPair_3

Total 2

Sorted in ascending order. Click to sort in descending order.

- This shows the current key aliases that are tracked by a specific KeySet.
- The aliases can be generated dynamically (when keys are generated dynamically) or they can be references to already existing aliases in the KeyStore.

Panel: LTPA Use of KeySetGroup for Key Management

Configuration

Key generation
Authentication data is encrypted and decrypted by using keys that are kept in one or more Key stores.

Key set group
CellLTPAKeySetGroup

[Key set groups](#)

Authentication expiration
Authentication information persists in the system for a limited amount of time before it expires and must be refreshed.

Authentication cache timeout
 minutes seconds

Timeout value for forwarded credentials between servers
 minutes

Cross-cell single sign-on
Single sign-on across cells can be provided by sharing keys and passwords. To share the keys and password, log on to one cell, specify a key file, and click Export keys. Then, log on to the other cell, specify the key file, and click Import keys.

* Password

* Confirm password

Fully qualified key file name

LTPA using KeySetGroup

- LTPA uses this new technology for key lifecycle management to ensure keys are changed frequently enough.
- By default, two versions of keys tracked (the latest is used to generate/validate recent keys, the oldest is used to validate older tokens).
- Dynamic generation has implications to cross-cell SSO. When importing a key from another Cell, it becomes the default key for both Cells (last keys imported/generated are the ones used for token generation). If you don't want to lose these keys due to wrapping, either increase the "maximum number of keys referenced" (could cause performance degradation) or disable dynamic generation to keep the list static.
- Since keys have moved into keystores, it's highly recommended to remove scripting that updates the old LTPA key location in security.xml. This effectively disables the new function.
- Password no longer required for LTPA keys explicitly, except when importing/exporting keys (required for KeyStore though).

Migration/Mixed Version/Mixed Platform Issues

- Migration will preserve keystores in entirety. If v6.02 is migrated to v6.1 and it used the Dummy certificates, these will be preserved. It's difficult to determine if some customization has occurred to all or part of an SSL configuration and safer to migration this way. These can more easily be converted to self-signed certificates in the new certificate management panels.
- Migrating the existing keystores allows a mixed version Cell to work properly as back-level servers do not have the advanced functions to handle signer exchange. Default certificates can more easily be replaced using the new certificate management panels.
- Back-level clients will have difficulty communicating with a v6.1 server (using self-signed certs). The client will need to be reconfigured to use the v6.1 key/trust stores or add the necessary signers using IKeyMan.
- When Dmgr is z/OS (using RACF keystores) and Node is distributed (using self-signed cert inside PKCS12 keystores), a manual signer exchange should occur prior to performing an addNode.
- When performing an addNode from a v6.02 Node to a v6.1 Dmgr, the v6.02 Node needs to exchange signers with the v6.1 Dmgr prior to being able to federate.
- Migration disables "Web inbound attribute propagation" whenever a v5.1 or prior Node is part of a mixed-version Cell due to problems interoperating with LtpaToken2 (new version of LtpaToken introduced in v5.1.1).

z/OS Specific Changes

- CSlv2 implementation for IIOp security now mostly common-code with distributed (Java-based).
- Support for HW Crypto exploitation for LTPA and web services security for soft non hw managed keys.
- Enhanced Sync-to-thread to move the enablement decision into SAF.
- Added support to optionally utilize the new RACF mixed case password.
- All but the daemon SSL has moved to JSSE.

Agenda

- *Enabled by Default*
- *Federated Repositories*
- *Keys and Certificates*
- ***SPNEGO***
- *Fine Grained Access Control*
- *Portlet URL Security*

Single sign-on (SSO) for HTTP using SPNEGO TAI

- Kerberos: from Greek mythology, Cerberus/Cerberos/Kerberos was the 3 headed dog guarding the gates to Hades.
- Kerberos is a 3 party security system – The requestor of a service, the service itself, and a trusted 3rd party
- Cryptographic Tokens are exchanged, not userids/passwords (passwords only flow when users change them)
- **Generic Security Services API** – a C API that abstracts security services. Kerberos is reference implementation.
 - ▶ Java SDK implements **Java GSS API**.
 - ▶ WebSphere Application Server (at sufficient service level) includes JGSS SPNEGO Provider for parsing SPNEGO tokens
- **Simple and Protected GSS-API Negotiation Mechanism**
- ▶ Defined IETF RFC 2478
- ▶ SPNEGO over HTTP was defined by Microsoft for exchanging credentials to a web server via HTTP (the focus of the TAI)
- ▶ SPNEGO token wraps a Kerberos Token

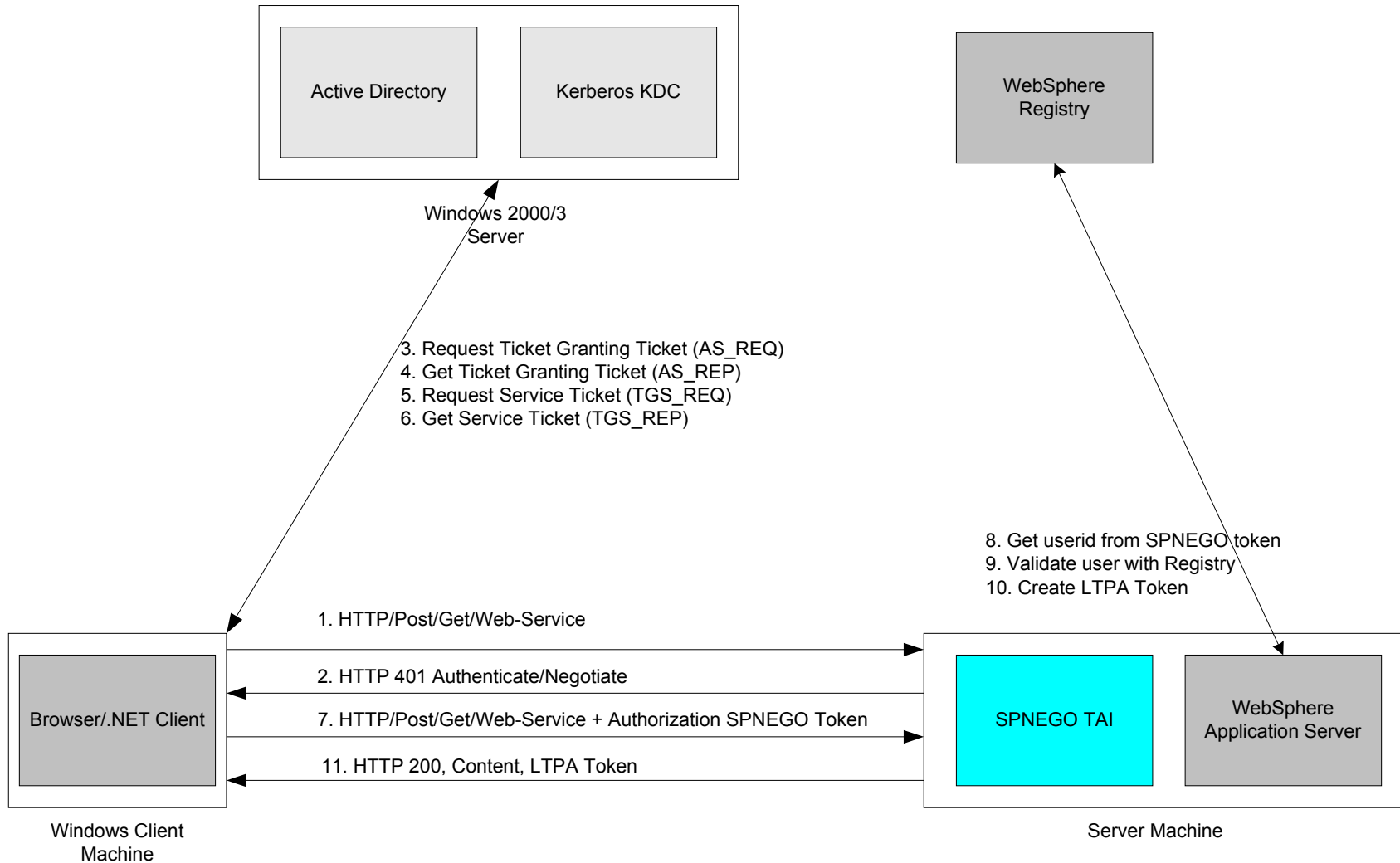
Single sign-on (SSO) for HTTP using SPNEGO TAI

- Windows Server 200x Active Directory Domain internally has a Kerberos core
- Domain Controllers act as Kerberos Key Distribution Centres (KDC), Ticket Granting Servers, LDAP directory and more
- WebSphere Application Server provides a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for protected resources in WebSphere Application Server.
- The same code “base” as the ISSW provided SPNEGO TAI, **but** it is not customizable (you do not get the source code)
- The 6.1 SPNEGO TAI is part of the product and fully supported by IBM support

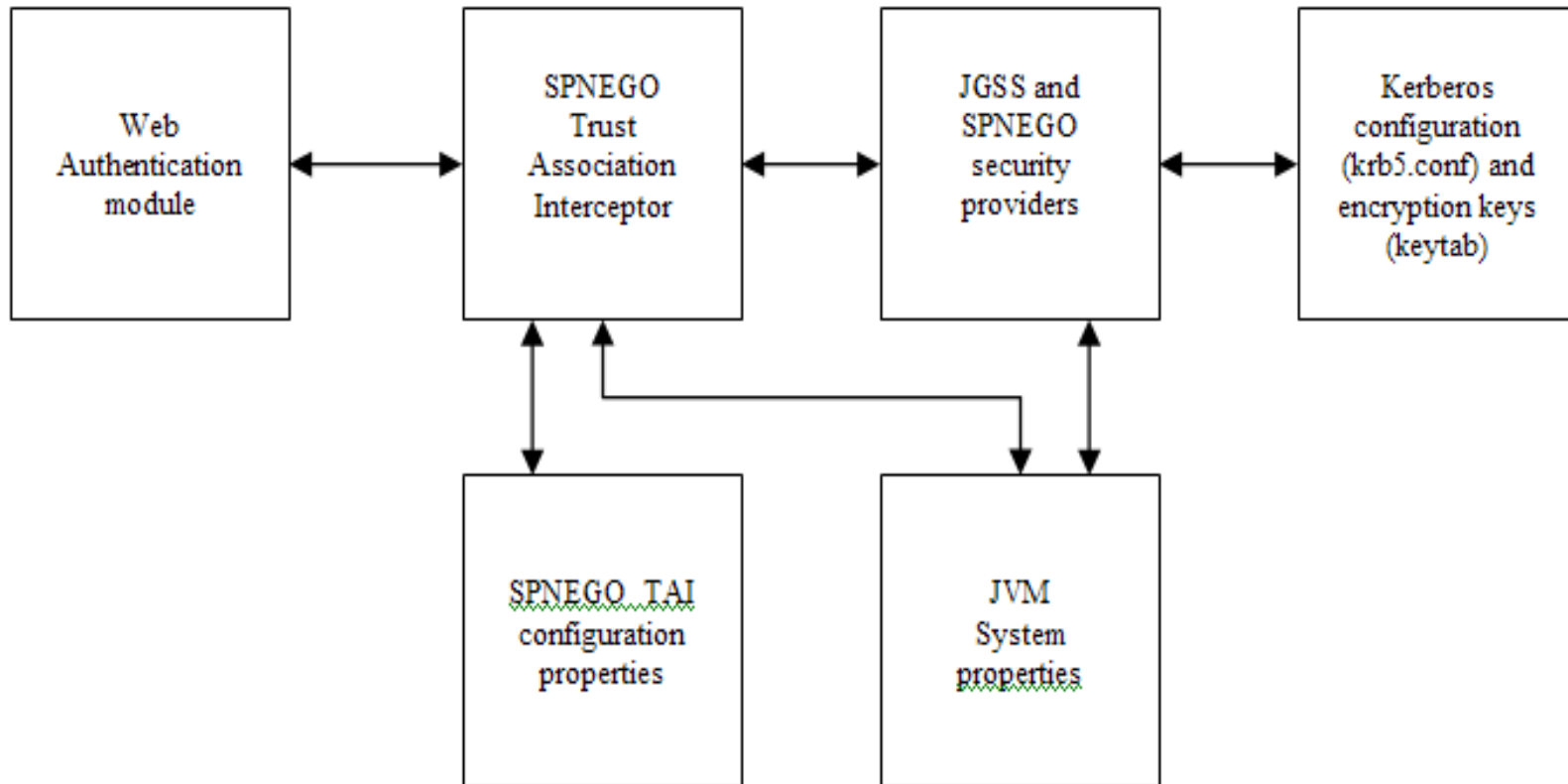
Single sign-on (SSO) for HTTP using SPNEGO TAI

- With SPNEGO TAI support, after a user login to the MS domain controller, the Web browser client does not have to provide a user ID and password again to access protected resources in WebSphere Application Server.
- Using the JAAS custom login module to map the client Kerberos principal name bob@kerberosRealmName to WebSphere user name.
- Support all User Registries and platforms that are supported by WebSphere Application Server.
- Support web browsers:
 - ▶ Microsoft Internet Explorer V6.0 SP1
 - ▶ Mozilla V1.7.8
 - ▶ Firefox V1.5
- Support one or more Microsoft (MS) domain controllers within the same forest.

Challenge-responses process between web browser and SPNEGO TAI



SPNEGO TAI configuration elements



Configure SPNEGO TAI for WebSphere Application Server

- Panel: Enable Trust Association

The screenshot shows the 'Trust association' configuration panel in the WebSphere Administration Console. The panel title is 'Secure administration, applications, and infrastructure > Trust association'. Below the title, there is a description: 'Enables trust association. Trust association is used to connect reversed proxy servers to the application server.' The panel is divided into two sections: 'General Properties' and 'Additional Properties'. In the 'General Properties' section, the 'Enable trust association' checkbox is checked. In the 'Additional Properties' section, there is a link for 'Interceptors'. At the bottom of the panel, there are four buttons: 'Apply', 'OK', 'Reset', and 'Cancel'.

Secure administration, applications, and infrastructure ?

[Secure administration, applications, and infrastructure](#) > **Trust association**

Enables trust association. Trust association is used to connect reversed proxy servers to the application server.

Configuration

General Properties

Enable trust association

Additional Properties

■ [Interceptors](#)

Apply OK Reset Cancel

Configure SPNEGO TAI for WebSphere Application Server (Continue)

- Panel: Enable SPENGO TAI through JVM system property
 - Application servers > server1 > Process Definition > Java Virtual Machine

Debug Mode

Debug arguments

Generic JVM arguments

Executable JAR file name

Disable JIT

Operating system name

Configure SPNEGO TAI for WebSphere Application Server (Continue)

- Configure SPNEGO TAI properties using wsadmin command. The following example set up the Service Principal Name (SPN) w2003secdev.austin.ibm.com, remove the Kerberos realm name from the Kerberos principal name for WebSphere registry lookup and intercept all HTTP request that have request-url partially equal to “snoop”.

```
wsadmin>$AdminTask addSpnegoTAIProperties -interactive
Add SPNEGO TAI properties

This command adds SPNEGO TAI properties in the security configuration.

*Host name in Service Principal Name (host): w2003secdev.austin.ibm.com
Service Principal Name identifier (spnId): 1
HTTP header filter rule (filter): request-url%=snoop
Name of class used to filter HTTP requests (filterClass):
SPNEGO not supported browser response (noSpnegoPage):
NTLM Token received browser response (ntlmTokenPage):
Remove Kerberos realm name from Kerberos principal name (trimUserName): true

Add SPNEGO TAI properties

F (Finish)
C (Cancel)

Select [F, C]: [F] f
WASX7278I: Generated command line: $AdminTask addSpnegoTAIProperties {-host w200
3secdev.austin.ibm.com -spnId 1 -filter request-url%=snoop -trimUserName true}
com.ibm.ws.security.spnego.SPN1.trimUserName=true
com.ibm.ws.security.spnego.SPN1.filter=request-url%=snoop
com.ibm.ws.security.spnego.SPN1.hostName=w2003secdev.austin.ibm.com
wsadmin>$AdminConfig save
```


Configure SPNEGO TAI for WebSphere Application Server (Continue)

- Configure a Kerberos client by creating the Kerberos configuration file (krb5.ini or krb5.conf).

```
wsadmin>$AdminTask createKrbConfigFile -interactive
Create Kerberos configuration file

This command creates a Kerberos configuration file (krb5.ini or krb5.conf).

*Filesystem location of the Kerberos configuration file (krbPath): c:\winnt\krb5
.ini
*Kerberos realm name in Kerberos configuration file (realm): WSSEC.AUSTIN.IBM.CO
M
*Host name of the Kerberos Key Distribution Center (kdcHost): axel.austin.ibm.c
m
Port number of the Kerberos Key Distribution Center (kdcPort):
*Default name of the Domain Name Service (dns): austin.ibm.com
*Filesystem location of the keytab file (keytabPath): c:\winnt\krb5.keytab
Encryption type (encryption):

Create Kerberos configuration file

F (Finish)
C (Cancel)

Select [F, C]: [F]
WASX7278I: Generated command line: $AdminTask createKrbConfigFile {-krbPath c:/w
innt/krb5.ini -realm WSSEC.AUSTIN.IBM.COM -kdcHost " axel.austin.ibm.cm" -dns au
stin.ibm.com -keytabPath c:/winnt/krb5.keytab}
c:\winnt\krb5.ini has been created.
wsadmin>
```

Configure SPNEGO TAI for WebSphere Application Server (Continue)

- Create a Kerberos key tab file from the Active Directory (AD) machine
 - ▶ Create a user name w2003secdev in AD and check the option **Use DES encryptions types for this account.**
 - ▶ Use MS setspn tool to map the user name to the SPN format **HTTP**/**<hostname>**
 - C:\MS SDK>setspn -a **HTTP**/w2003secdev.austin.ibm.com w2003secdev
 - ▶ Use MS ktpass tool to generate the Kerberos keytab file krb5.keytab for the SPN
 - ktpass -out c:\temp\krb5.keytab -princ **HTTP/w2003secdev.austin.ibm.com@WSSEC.AUSTIN.IBM.COM** -mapUser w2003secdev -mapOp set -pass security -crypto DES-CBC-MD5 +DesOnly
- Copy the krb5.keytab file to the WebSphere Application Server machine at the location which specify in the Kerberos configuration file (krb5.ini or krb5.conf).
- Note: The Windows 2003 server ktpass support both DES and RC4-HMAC

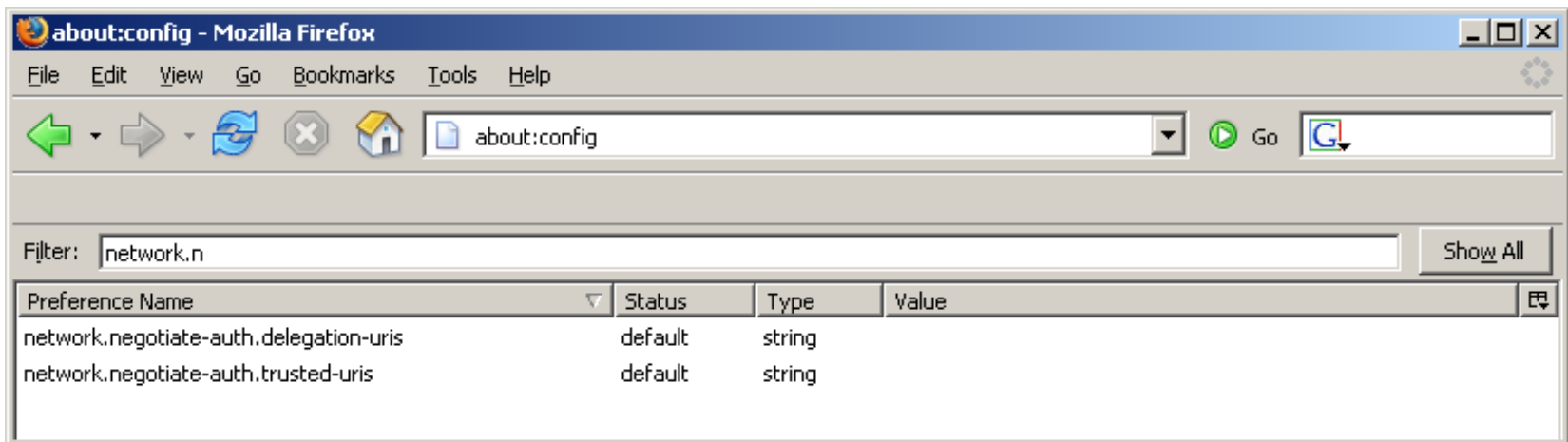
Configure MS IE browser to use SPNEGO authentication

Make sure the client machine is part of a domain for which SSO has been defined. In the following example, the machine `w2003secdev.austin.ibm.com` is a member of the domain controller **wssec.austin.ibm.com**. Log on to the Windows Desktop with a user name from the domain.

- ▶ Open the browser, go to menu bar **Tools -> Internet Options**
- ▶ Select the **Security** tab.
- ▶ Select **Local intranet** icon.
- ▶ Click **Sites**.
- ▶ Click **Advanced**.
- ▶ Add the URL for the host name that SSO should be enabled for, to the list. For example:
http://w2003secdev.austin.ibm.com
- ▶ Click **OK**.
- ▶ Click **OK**.
- ▶ Select the **Advanced** tab.
- ▶ Scroll down to security section and ensure that **Enable integrated Windows Authentication(requires restart)** is checked.
- ▶ Close the browser.
- ▶ Start the browser.

Configure Mozilla or FireFox browser to use SPNEGO authentication

- ▶ Open the browser.
- ▶ At the address field, type **about:config**
- ▶ In the filter, type **network.n**



- ▶ Double click on `network.negotiate-auth.trusted-uris`. This preference lists the sites that are permitted to engage in SPNEGO Authentication with the browser
- ▶ Enter a comma delimited list of trusted domains or URLs. For example: `http://w2003secdev.austin.ibm.com`
- ▶ Close the browser.
- ▶ Start the browser.

Agenda

- *Enabled by Default*
- *Federated Repositories*
- *Keys and Certificates*
- *SPNEGO*
- ***Fine Grained Access Control***
- *Portlet URL Security*

Fine-Grained Admin Access Control

- Configurable by scripting only.
- Authorization groups setup to group resources.
- Resources include Cell, Node, ServerCluster, Server, Application, or NodeGroup.
- Cell authorization group available by default for backwards compatibility.
- A resource can only belong to a single authorization group.
- Users/groups assigned to roles and authorization groups.

Scripting Operations to Setup Fine-Grained Admin Authz

- **Create a new authorization group:** `$AdminTask createAuthorizationGroup {-authorizationGroupName authGroup1}`
- **Deleting an authorization group:** `$AdminTask deleteAuthorizationGroup {-authorizationGroupName groupName}`
- **Add resources to an authorization group:** `$AdminTask addResourceToAuthorizationGroup {-authorizationGroupName groupName -resourceName Application=app1}`
- **Remove resources from an authorization group:** `$AdminTask removeResourceFromAuthorizationGroup {-authorizationGroupName groupName -resourceName Application=app1}`
- **Add user IDs to roles in an authorization group:** `$AdminTask mapUsersToAdminRole {-authorizationGroupName groupName -roleName administrator -userids user1}`
- **Add group IDs to roles in an authorization group:** `$AdminTask mapGroupsToAdminRole {-authorizationGroupName groupName -roleName administrator -groupids group1}`
- **Remove user IDs from roles in an authorization group:** `$AdminTask removeUsersFromAdminRole {-authorizationGroupName groupName -roleName administrator -userids user1}`
- **Remove group IDs from roles in an authorization group:** `$AdminTask removeGroupsFromAdminRole {-authorizationGroupName groupName -roleName administrator -groupids group1}`

Agenda

- *Enabled by Default*
- *Federated Repositories*
- *Keys and Certificates*
- *SPNEGO*
- *Fine Grained Access Control*
- ***Portlet URL Security***

Portlet URL security

- WAS 6.1 has an embedded JSR168 Portlet Container.
- One can directly request a portlet through a URL to display its contents without portal aggregation.
- Similar to servlets one can invoke a portlet by its context root with the URL mapping /<portlet-name> that is created for each portlet.
- Portlets can be protected just like servlets when accessed using the URL.
- The information in the portlet.xml like the user-data-constraint and security-role-ref will be combined with the information in the web.xml for that portlet which can be explicitly defined or implied through an url-pattern.

Example

Portlet relevant Security Constraints in web.xml

```
-----  
<security-constraint id="SecurityConstraint_1">  
  <web-resource-collection id="WebResourceCollection_1">  
    <web-resource-name>Protected Area</web-resource-name>  
    <url-pattern>/MyPortlet1/*</url-pattern>  
    <url-pattern>/MyPortlet2/*</url-pattern>  
  </web-resource-collection>  
  <auth-constraint id="AuthConstraint_1">  
    <role-name>Employee</role-name>  
  </auth-constraint>  
</security-constraint>
```

Security Constraints in Portlet.xml

```
-----  
<security-constraint>  
  <display-name>Secure Portlets</display-name>  
  <portlet-collection>  
    <portlet-name>MyPortlet1</portlet-name>  
    <portlet-name>MyPortlet3</portlet-name>  
  </portlet-collection>  
  <user-data-constraint>  
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>  
  </user-data-constraint>  
</security-constraint>
```

Protection based on above constraints

URL	Transportation Protection	User Authentication	Authorization
MyPortlet 1/*	HTTPS	Yes	Yes (Employee)
MyPortlet 2/*	None	Yes	Yes (Employee)
MyPortlet 3/*	HTTPS	None	None
URL	Transportation Protection	User Authentication	Authorization

Legal

- **© Copyright IBM Corporation 2008. All rights reserved.**
-
- **IBM, the IBM logo, the e-business logo and other IBM products and services are trademarks or registered trademarks of the International Business Machines Corporation, in the United States, other countries or both. References in this publication to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.**
-
- **Product release dates and/or capabilities referenced in this publication may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.**
-
- **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 trademarks of Microsoft Corporation in the United States, other countries or both.**
-
-
- **All other trademarks, company, products or service names may be trademarks, registered trademarks or service marks of others.**