Securing your WebSphere Message Broker

David Coles – WebSphere Message Broker Level 3 Service,
IBM Hursley – dcoles@uk.ibm.com

Wednesday 4th August 2010

- Welcome to this Technical Introduction to securing your WebSphere Message Broker.
- Some slides in this presentation have at least one corresponding notes slide like this one, which contains further information on the topic being discussed, and/or links to web pages.
- Only this notes slide will be shown during the presentation. To view all other notes slides, please download and view a copy of this presentation.
- The WebSphere Message Broker homepage can be found at http://www.ibm.com/software/integration/wbimessagebroker/

Agenda

- Introduction
- Administration security
  - Message Broker V7 recap
  - Security exits
  - Channel security
- Runtime security
  - Transport security
  - Database security
- Message flow security
  - Security Manager
  - WS-Security
- Demo / Sample
- Summary

Notes: Agenda

- This presentation is divided into several sections. We'll begin by describing what security is and why it is important. We'll then relate this to the different areas of Message Broker.

- Message Broker exposes three important concepts relating to security. We'll introduce administration security, runtime security and message flow security, looking into each concept explaining how it applies to Message Broker, giving an overview of the functionality and an introduction its configuration.

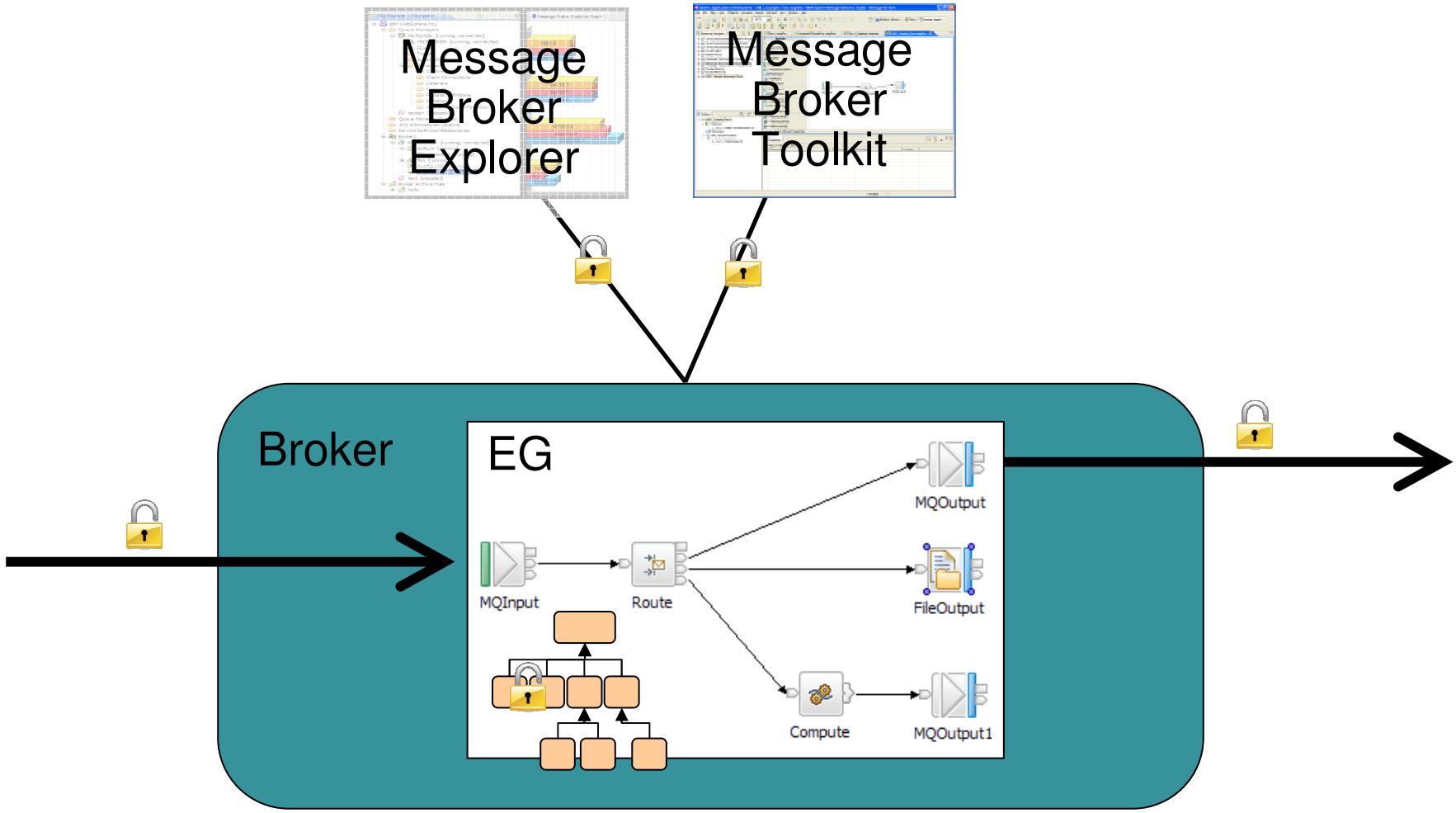
- Finally we'll highlight a technology sample that is supplied with Message Broker that really showcases the exciting new functionality available for message flow security.

Introduction - Security Overview

- Security is about preventing *unauthorised* access
  - The need to know
  - Covers multiple aspects of Message Broker configuration and usage

- Computer security generally refers to the 3A's
  - Authentication
    - Is the user who they say they are
  - Authorization
    - Is the user allowed to perform the given action
  - Accounting
    - Keeping track of who is accessing a resource and when

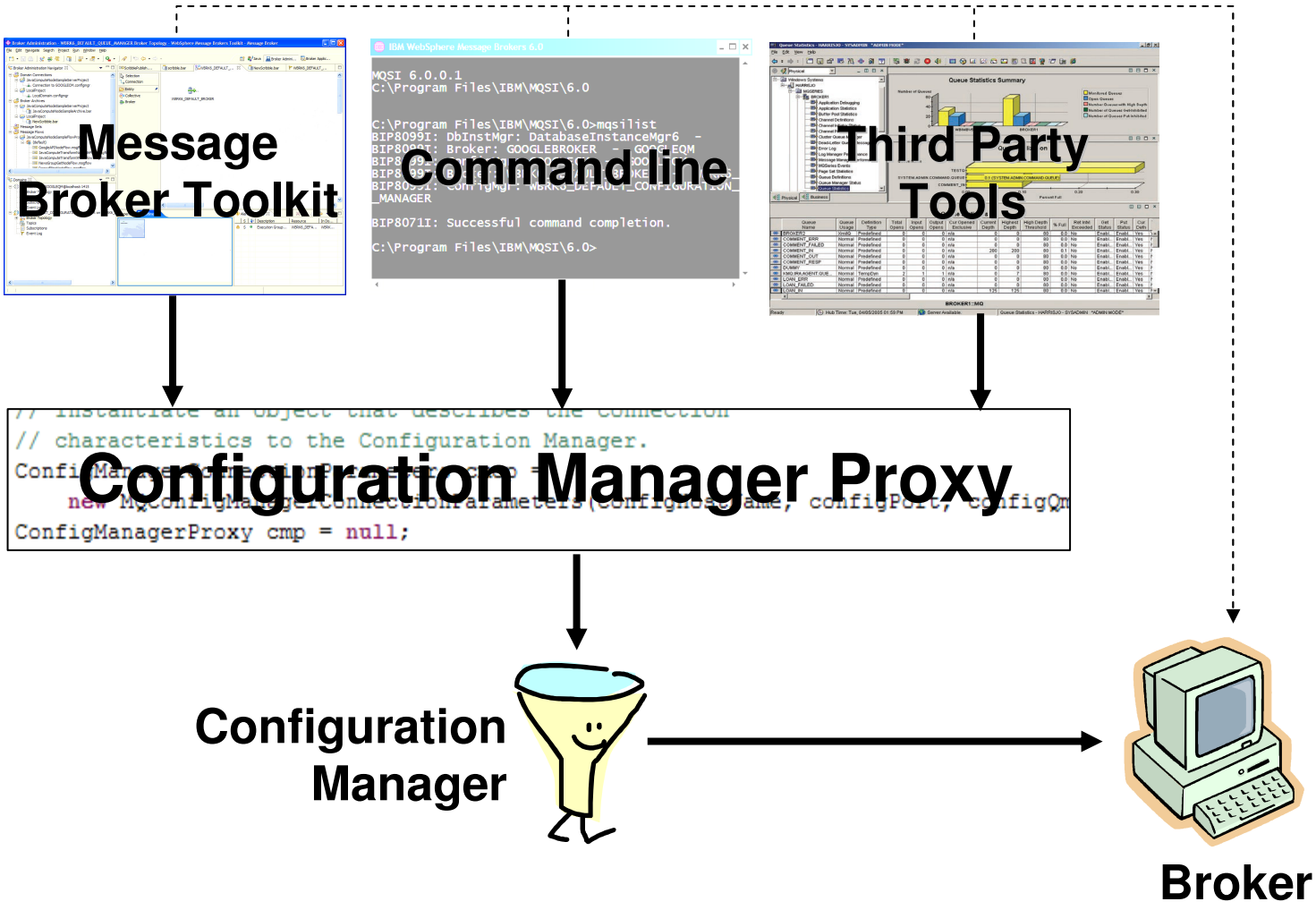
Introduction - Security Overview



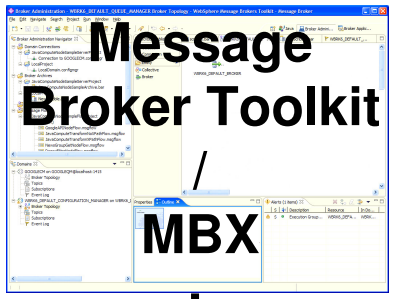
Notes: Introduction - Security Overview

- Three main areas to Message Broker security
- Administration security
  - Who is authorized to perform administrative actions on a Broker
- Runtime security
  - Who is authorized to submit messages to a Broker
- Message Flow Security
  - End-to-end processing of the message on the behalf of the identity in the message

Administration Security (V6.X Recap)

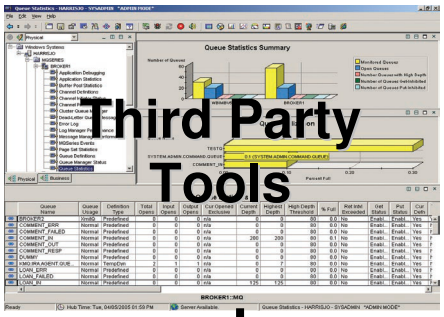


Administration Security (V7)



Message Broker Toolkit / MBX

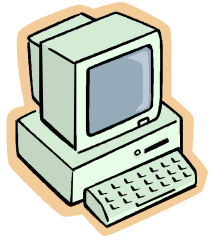
Command line



Third Party Tools

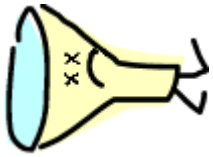
```
// characteristics to the broker.
BrokerConnectionParameters bcp
    new MQBrokerConnectionParameters(brokerHostName, brokerPort, brokerQmg
BrokerProxy b = null;
```

CMP API



Broker

Configuration Manager



Notes: Administration Security

- Restriction of user access
  - Who is authorized to deploy resources to Brokers
  - Who is authorized to run Broker administrative commands
  - Access controlled by WMQ access control model

- Preventing unauthorized access to deployment messages
  - Security exits
  - Channel security

Configuration Manager Removal - Benefits

- The Broker environment will be a lot easier to manage
- One view of the world
- More information returned to tools
- Much improved connect and deploy times
- Long-standing niggles have been eliminated. V7 has:
  - One-step broker creation (i.e. no CM association step)
  - No “Deployment already in progress” messages
  - No CM/Broker Synchronization problems
    - Cancel Deployment
    - Performance
  - As well as:
    - No service user ID requirement on non-Windows platforms
    - No default execution groups (i.e. to host pub/sub)

Configuration Manager Responsibilities

	In V7
Interaction with Tools (CMP apps)	**Broker handles admin connections**
Deployment	**Broker handles BAR file deployment**
Owner of a domain of brokers	**Domains concept has been removed**
Managing administrative security	**Security managed using MQ**
Enforcing administrative security	**Broker is Policy Enforcement Point**
Manages the pub/sub topology	**Pub/Sub managed using MQ v7 tools**
Managing subscriptions	**Pub/Sub managed using MQ v7 tools**
Manages the topics hierarchy	**Pub/Sub managed using MQ v7 tools**

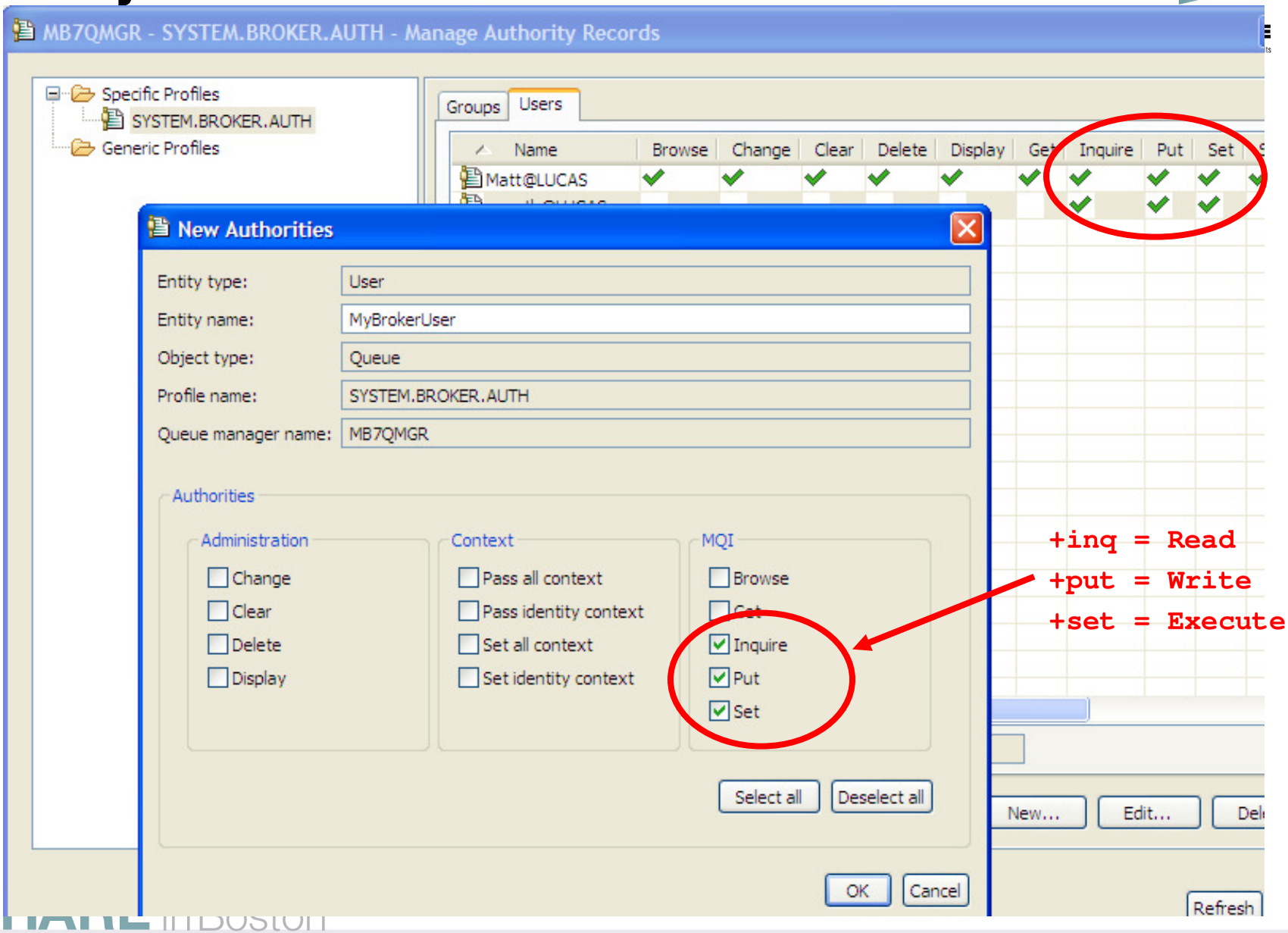
Administrative Security

- Simplified administrative security in V7 allows 3 levels of authorisation for administrative actions:
  - Reading
  - Writing
  - Executing (i.e. starting and stopping)
- On two object types:
  - Broker
  - Execution Group
- Administrative security is not enabled by default
- Access controlled using MQ queues on the Broker's queue manager
- Guidance provided for migration from CM ACLs
  - Though there is not a one-to-one mapping

Security Queues

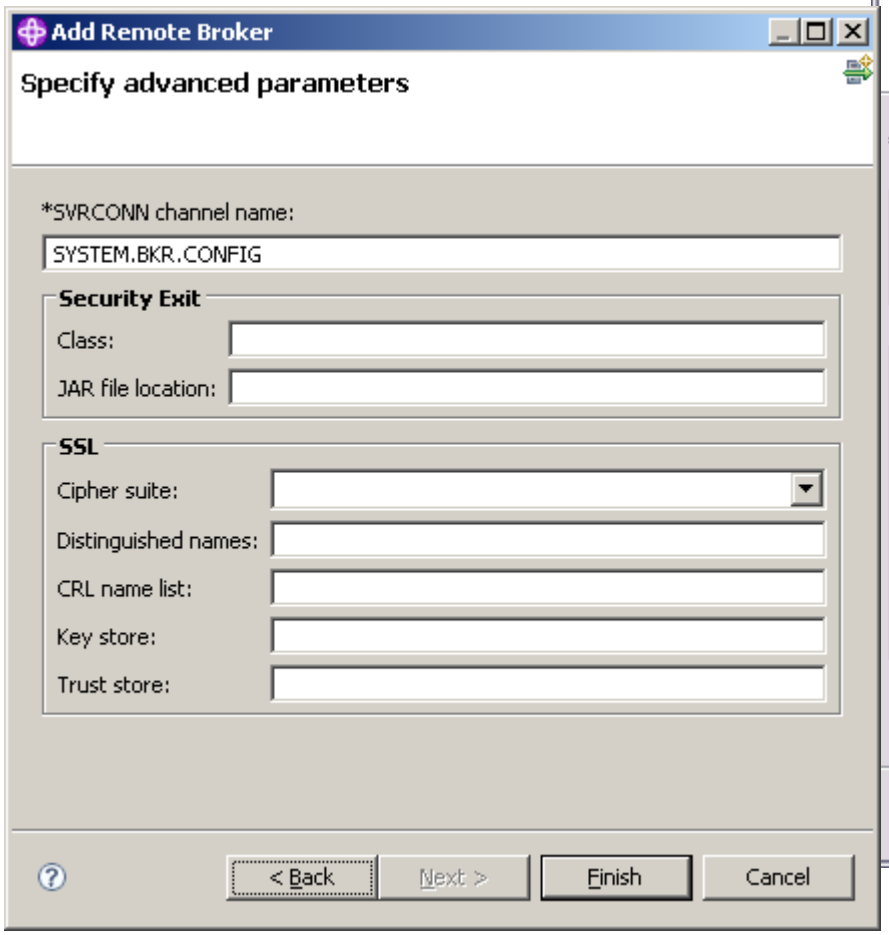
SYSTEM.BROKER.AUTH

SYSTEM.BROKER.AUTH.<egname>

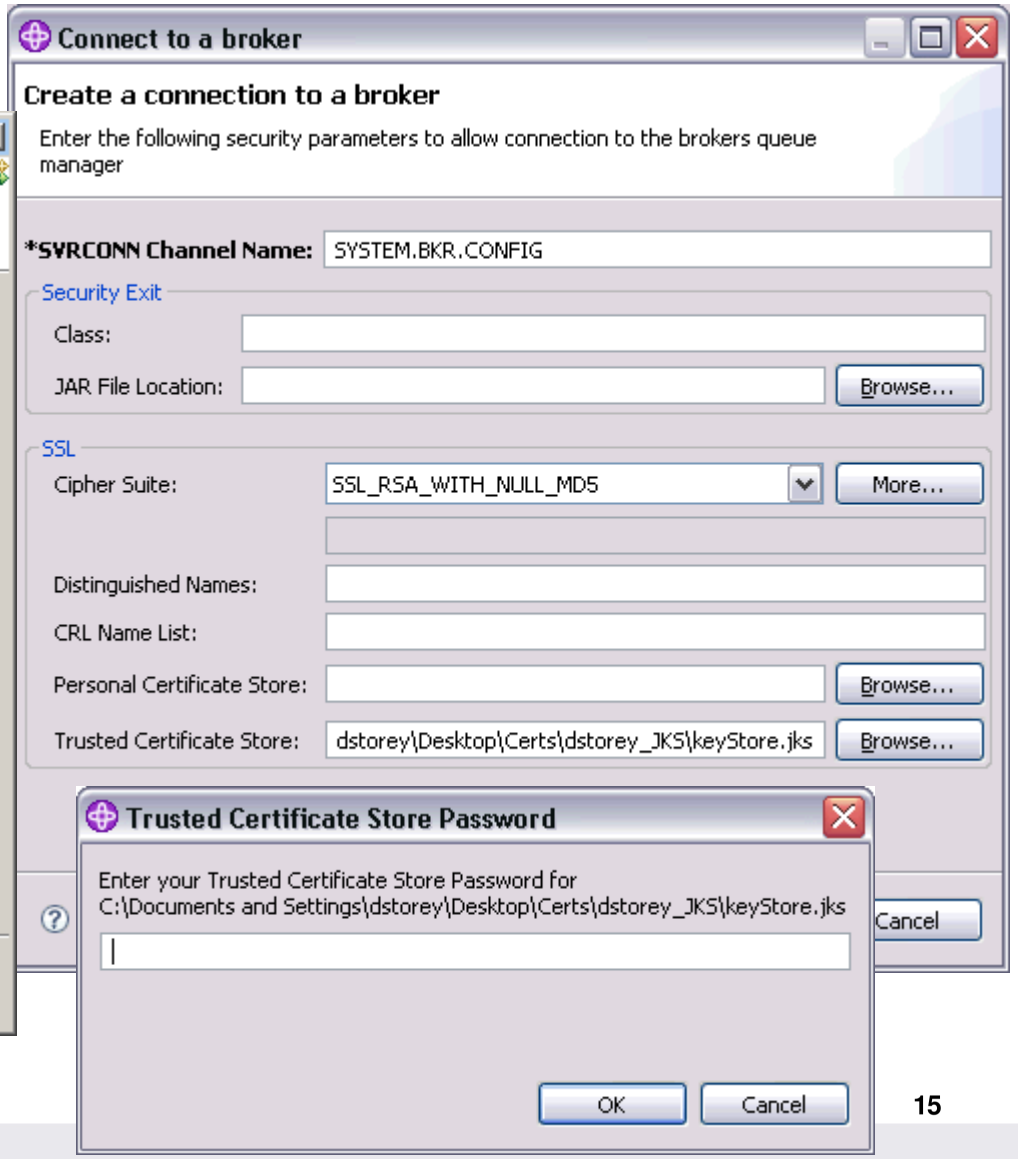


Security Exits / Channel security

- Message Broker Toolkit



- Message Broker Explorer



Notes: Security Exits

- Used to verify that the partner at the other end is genuine
- Use MQ security exits to secure access to the Broker from the WebSphere Message Broker Toolkit, WebSphere Message Broker Explorer or client programs
- You can enable a security exit at each end of the connection between your client session and the Broker:
- Set up a security exit on the channel at the Broker end. This security exit has no special requirements; you can provide a standard security exit.
- Set up a security exit in the WebSphere Message Broker Toolkit or WebSphere Message Broker Explorer. Identify the security exit properties when you connect to the broker.
- The security exit is a standard WebSphere MQ security exit, written in Java™.
- See http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/topic/com.ibm.etools.mft.doc/ap12500_.htm and http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/topic/com.ibm.etools.mft.doc/ap12510_.htm for more details

Notes: Channel security

- Secure the server conn channel used to connect the WebSphere Message Broker Toolkit, WebSphere Message Broker Explorer or client programs to the Message Broker
- Keystores and truststores specified when you configure the connection
- Passwords prompted for when you initiate the connection
- Same design as used with MQExplorer to connect to Queue Managers
- See http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/topic/com.ibm.etools.mft.doc/ap12232_.htm for more details.

Runtime Security

- Who is authorized to submit a message to a message flow
  - Delegated to the transport
- Can be offloaded to DataPower appliance
- What resources can be accessed by that message flow
- Transport Security – SSL
- Database Security

SSL (Secure Sockets Layer)

- Transport layer protocol for data encryption
- Protocol based on SSL Certificate enables encryption of sensitive information during online transactions
- Each SSL Certificate contains unique, authenticated information about the certificate owner
- A Certificate Authority verifies the identity of the certificate owner when it is issued
- Each certificate consists of a public key and a private key
  - Public key is used to encrypt information
  - Private key is used to decipher it
- A certificate owner keeps its private key and the public key is distributed
- An SSL handshake authenticates the server and the client
  - An encryption method is established with a unique session key and secure transmission can begin

SSL (Secure Sockets Layer)

- Inbuilt SSL Support in multiple Broker nodes
  - HTTP/SOAP Nodes (HTTPS connections)
  - CICSRequest
  - IMSRequest
  - Also JMS Nodes with compliant provider
- Hierarchical configuration for keystores and truststores
- Java JKS keystore format supported
- Support for Server auth and Client auth
  - Server auth
    - The client trusts the server
    - Server's public cert is in the client's truststore
  - Client auth
    - Builds on server auth and the server also trusts the client
    - Client's public cert is in the server's truststore

Notes: SSL (Secure Sockets Layer)

- Example SSL configuration for the HTTP listener
  - Create a key store to hold the brokers certificates using keytool
  - Configure the broker to use SSL on a particular port
    - Turn on SSL support in message broker, by setting a value for **enableSSLConnector** mqsichangeproperties *broker name* -b httplistener -o HTTPListener -n enableSSLConnector -v true
    - Choose the keystore file to be used, by setting a value for **keystoreFile** mqsichangeproperties *broker name* -b httplistener -o HTTPSConnector -n keystoreFile -v *fully qualified file path to keystore file*
    - Specify the password for the keystore file, by setting a value for **keystorePass** mqsichangeproperties *broker name* -b httplistener -o HTTPSConnector -n keystorePass -v *password for keystore*
    - Specify the **port** on which WebSphere Message Broker should listen for HTTPS requests mqsichangeproperties *broker name* -b httplistener -o HTTPSConnector -n port -v *Port to listen on for https*
  - Configure the message flow to process HTTPS requests
    - Specify PathSuffix for HTTPInput node
    - Select UseHTTPS box on the HTTPInput node
- More details on implementing SSL authentication can be found here: http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/topic/com.ibm.etools.mft.doc/ap12230_.htm

Database Security
- Broker database removal

- At WMB v7 the Broker no longer uses a system database
  - Configuration is now stored exclusively on the filesystem
  - WMB does not ship with a database product
  - User database access unaffected
- Additionally, the Windows registry is no longer used to hold configuration information
- New *mqsibackupbroker* and *mqsirestorebroker* commands to backup and restore (for DR)
- Migration will copy any system database and registry configuration to the filesystem

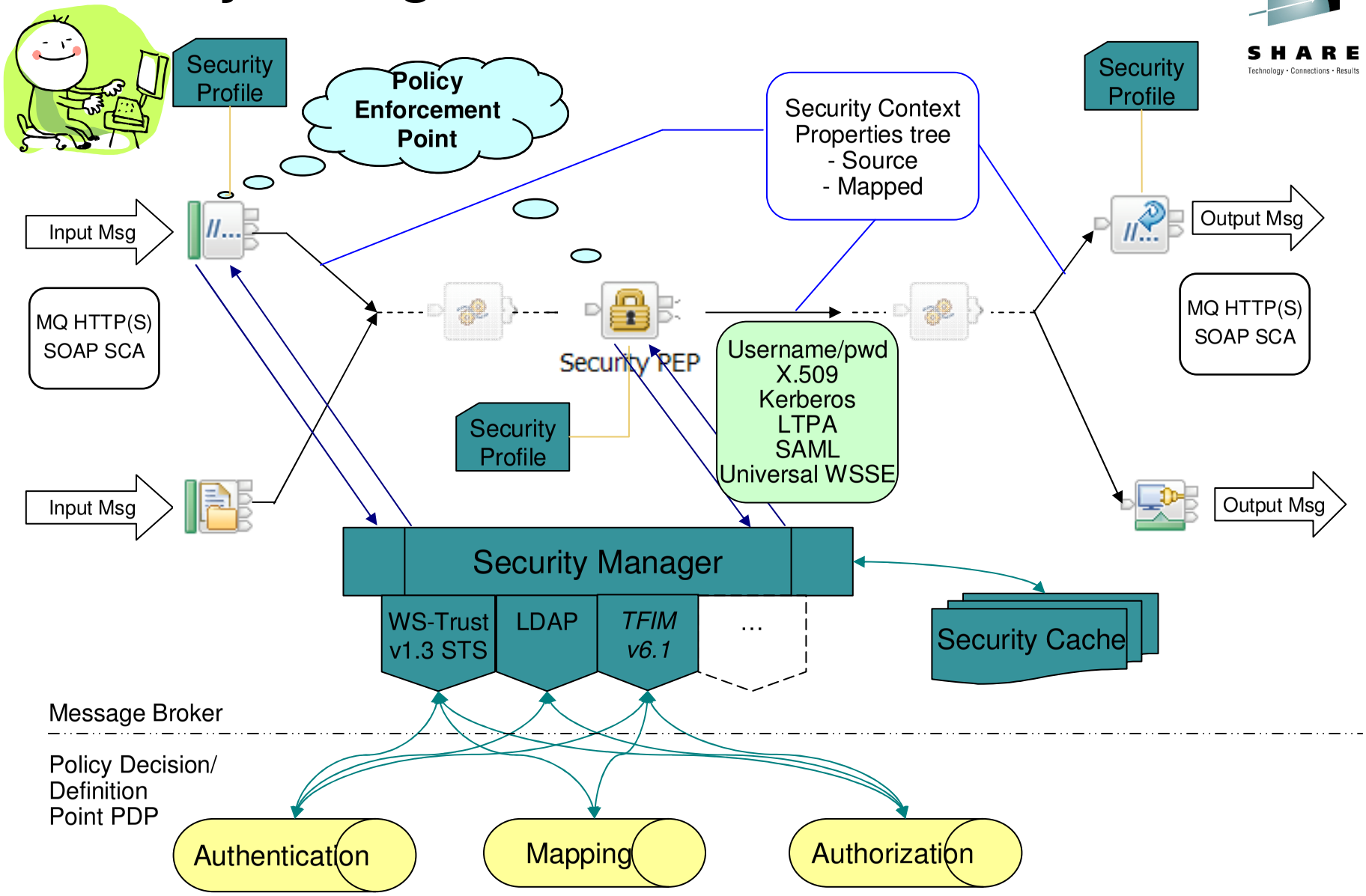
Database security - UserIds

- Database UserID and Password
  - No longer used on *mqsicreatebroker* – flags ignored
  - Use *mqsisetdbparms* to control default ODBC and JDBC access control
  - Any v6.x defaults are migrated
- Service UserID and Password
  - No longer used on non-Windows platforms
  - Still required on Windows, but can now specify LocalSystem
- The userid that starts the broker no longer requires *mqm* authority

Message Flow Security

- Default security means transport defaults are in effect
  - Broker service identity will be used as proxy id for all messages
- Security manager enables end-to-end security processing
  - Uses identity in the message – security on a per message basis at runtime
    - Identity authentication
    - Identity mapping
    - Identity authorization (policy enforcement)
    - Identity propagation
  - Data format and transport independent
- Configurable by administrator
  - Using 'security profiles'
- Able to exploit centralized security provider
  - LDAP for authentication and authorization
  - IBM **T**ivoli **F**ederated **I**dentity **M**anager (TFIM) for authentication, authorization and mapping
  - WS-Trust v1.3 compliant security token server (STS)

Security Manager Overview



Notes: Security Manager overview #1

- The first step in configuring the security manager is to create a security profile. This is done using either
  - Message Broker Explorer
  - *mqsicreateconfigurableservice* command

  This enables the administrator to define any of three possible security operations and provide the required configuration to define the external security policy decision point that will be invoked
- The next step is to associate the security profile with a node to enforce the security, Policy Enforcement Point, either a input node or a SecurityPEP node. This is done using the BAR file Editor.
- The flow developer may need to specify the type and location of the security tokens in the message on the Input or Security PEP node using XPaths or ESQL expressions or for SOAP nodes a Policy Set and Binding will define the token types
- At runtime the security manager extracts the identity information from the input message and sets it in a group of Source Identity elements in the Properties folder.
- If authentication was specified in the security profile, the security manager calls the provider to authenticate the identity. A failure results in a SecurityException being thrown.

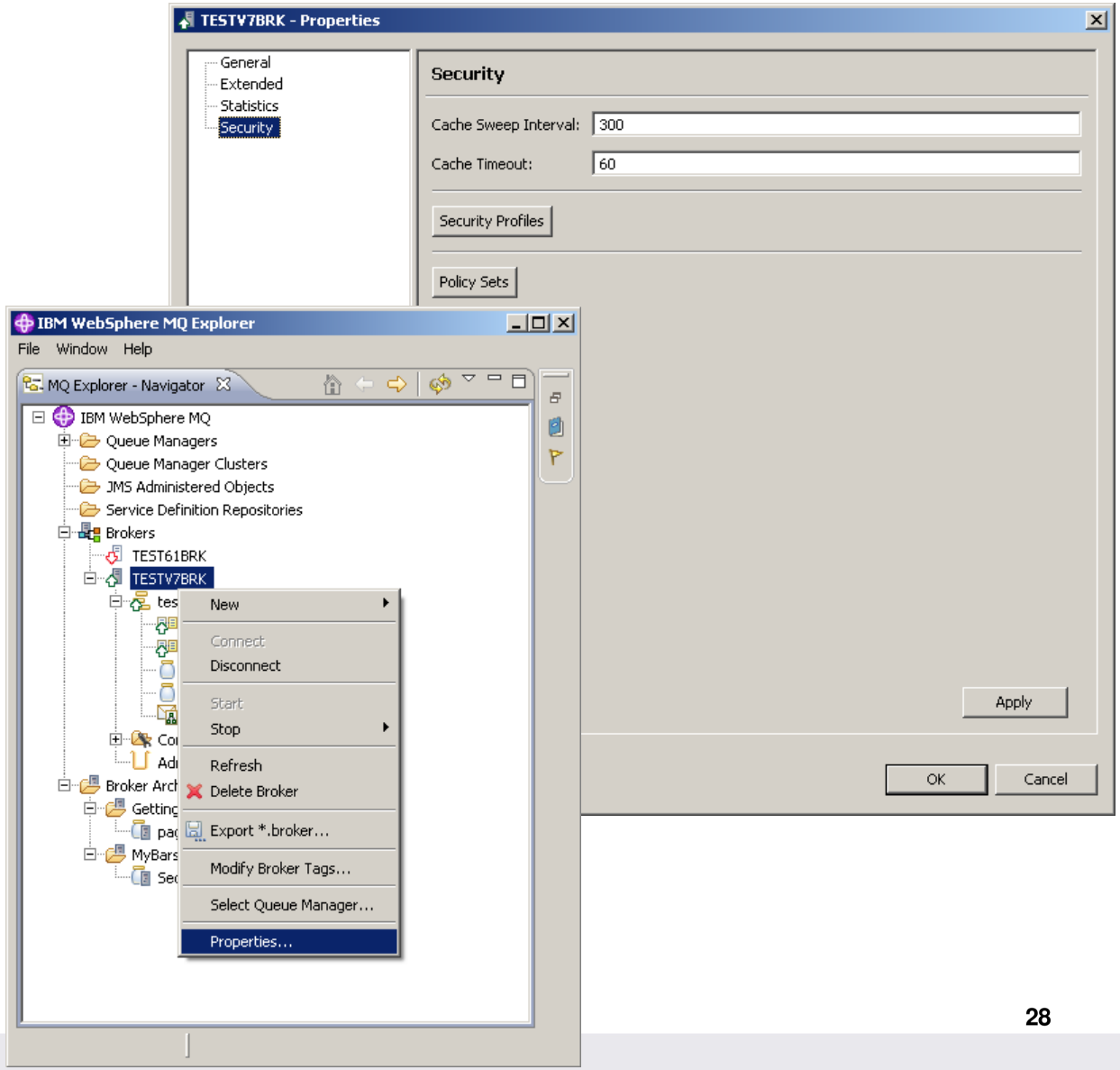
Notes: Security Manager overview #2

- If identity mapping was specified in the security profile, the security manager calls the provider to map. A failure results in a SecurityException being thrown. Otherwise the 'mapped' identity is set in Mapped Identity elements in the Properties folder.
- If authorization was specified in the security profile, the security manager calls the provider to authorize that the identity has access to this message flow. A failure results in a SecurityException being thrown.
- Note if the Security provider is a Security Token Server then all operations are performed in a single invocation
- The message, including the Properties folder and its source and mapped identity information, is propagated down the flow.
- When the message reaches an output node, a security profile can be used to indicate the identity is to be propagated in the message. The mapped identity is used, or if that is not set, the source identity is used. If no identity is set a SecurityException is thrown.
- To improve performance, authentication, authorization and mapping information from the providers is cached for re-use. The operation of the cache is automatic, but it can be tuned if needed using the *mqsichangeproperties* and *mqsireloadsecurity* commands.

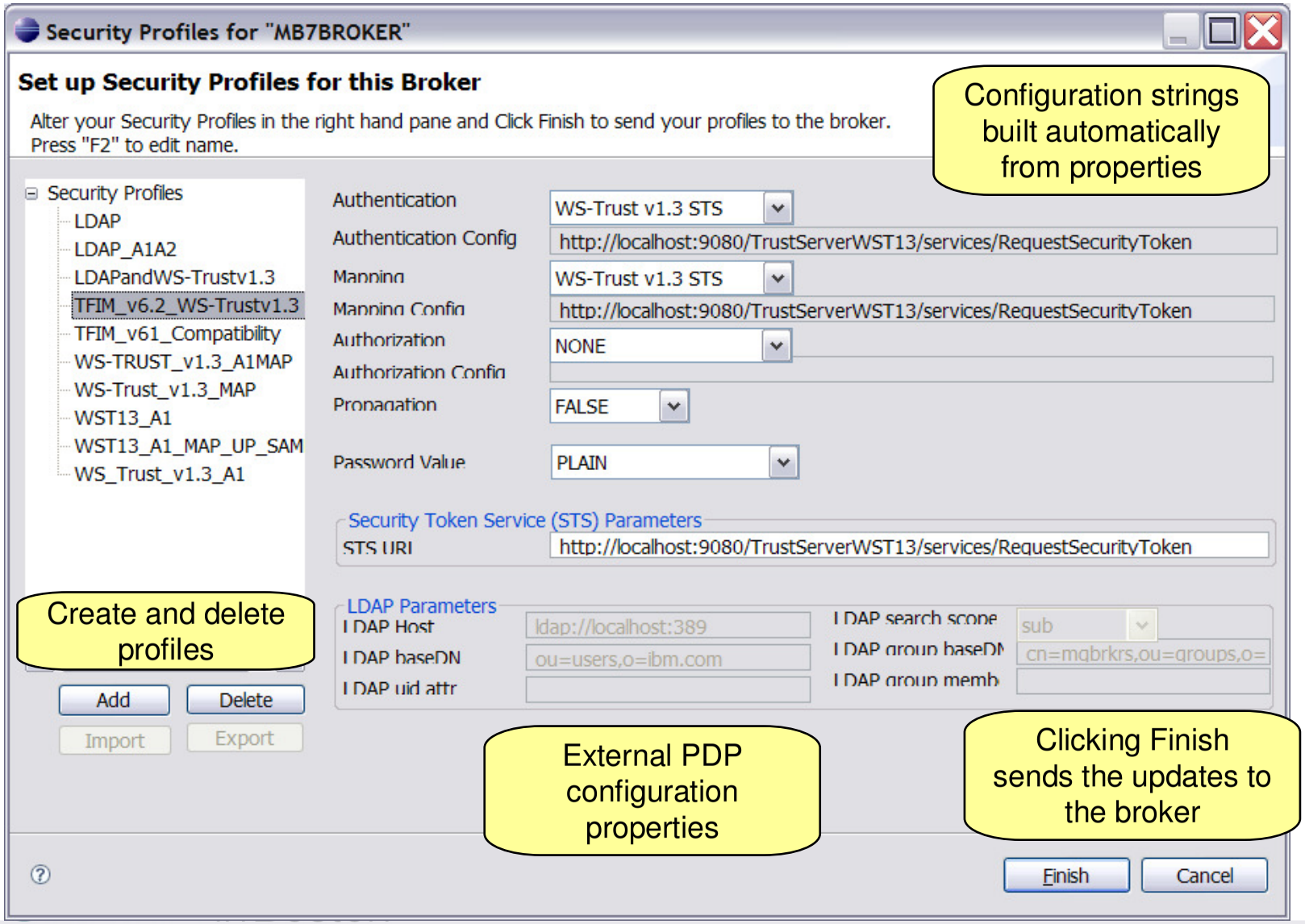
- More details on message flow security can be found here: http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/topic/com.ibm.etools.mft.doc/ap04090_.htm

Creating security profiles using the editor

- Security profiles are configured in MBX
- Right clicking on your broker and selecting ‘Properties’ will open the Broker properties pane.
- From there select ‘*Security Profiles’* in the ‘Security’ tab to open the Security Profiles editor.
- You can also load the ‘Policy Sets’ editor from here.



Creating security profiles using the editor



Security profiles

- A security profile contains the following settings
  - authentication = {NONE, LDAP, WS-Trust v1.3 STS,…}
  - authenticationConfig = …
  - mapping = {NONE, WS-Trust v1.3 STS,…}
  - mappingConfig = …
  - authorization = {NONE, LDAP, WS-Trust v1.3 STS,…}
  - authorizationConfig = …
  - passwordValue = {PLAIN, MASK, OBFUSCATE}
  - propagation = {TRUE, FALSE}

Policy enforcement on behalf of Configured Policy Decision Point

In properties tree

Propagation

-Input node, just extract the tokens

- Output/Request nodes, forward the token

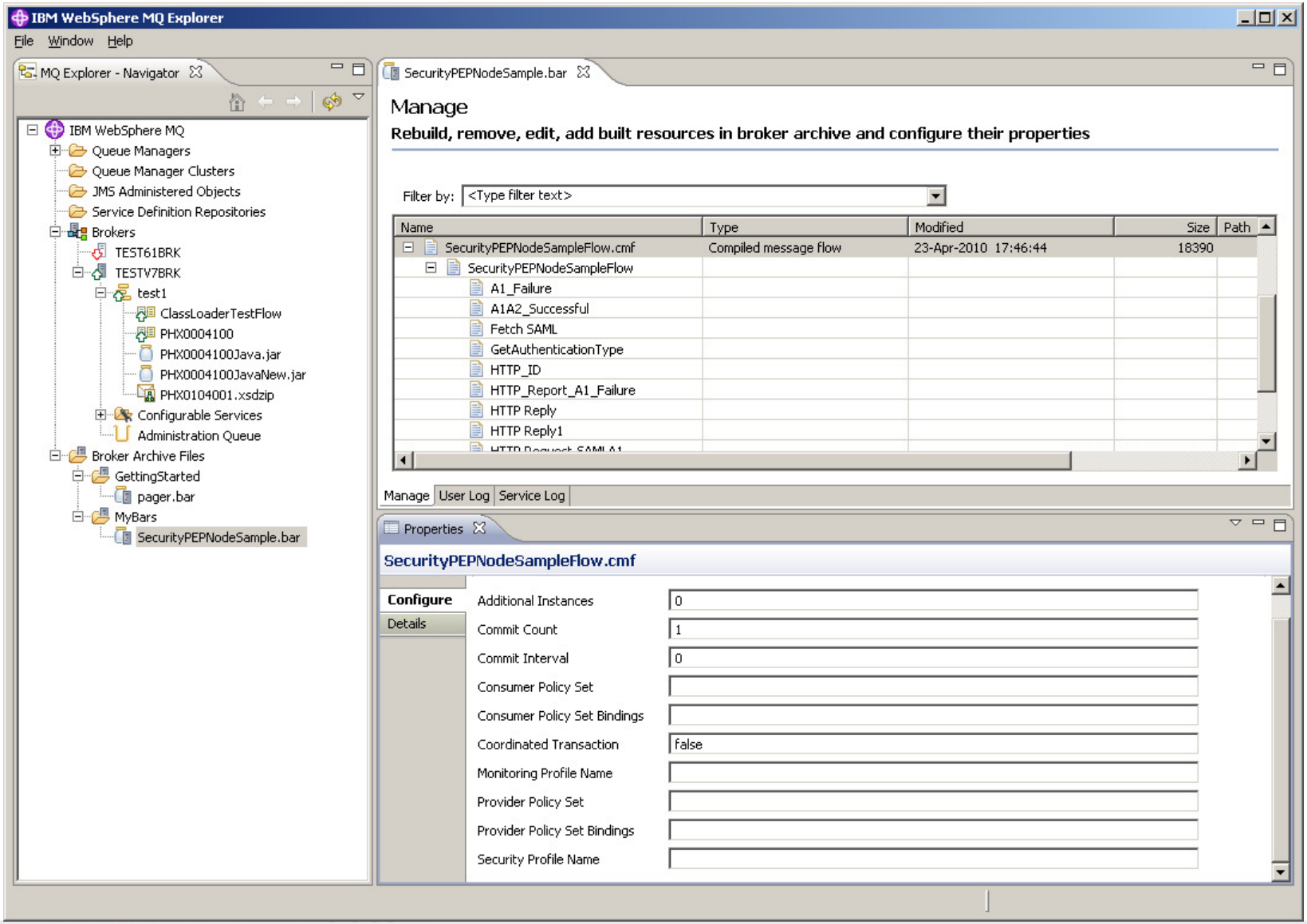
Notes: Security profiles

- A security profile consists of two kinds of information:
  - Policy enforcement (PEP) information. Whether to authenticate, authorize or map an identity along with the provider to use and associated configuration string
  - Propagation information. Whether to propagate the identity with an output message.
- Security profiles may be created, deleted, viewed and edited using a security profile editor, part of the broker toolkit administration perspective. This assists with the building of the sometimes complex configuration strings needed by the providers. Clicking on the Finish button of the editor sends the updates direct to the broker. Security profiles are *not* deployed in the .bar file.
- Alternatively security profiles may be created, deleted, viewed and deleted using the broker *mqsicreateconfigurableservice, mqsideleteconfigurableservice, mqsichangeproperties* and *mqsireportproperties* commands, or their CMP API equivalent.
- Further details and information on security profiles and their configuration can be found here: http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/topic/com.ibm.etools.mft.doc/ap04070_.htm

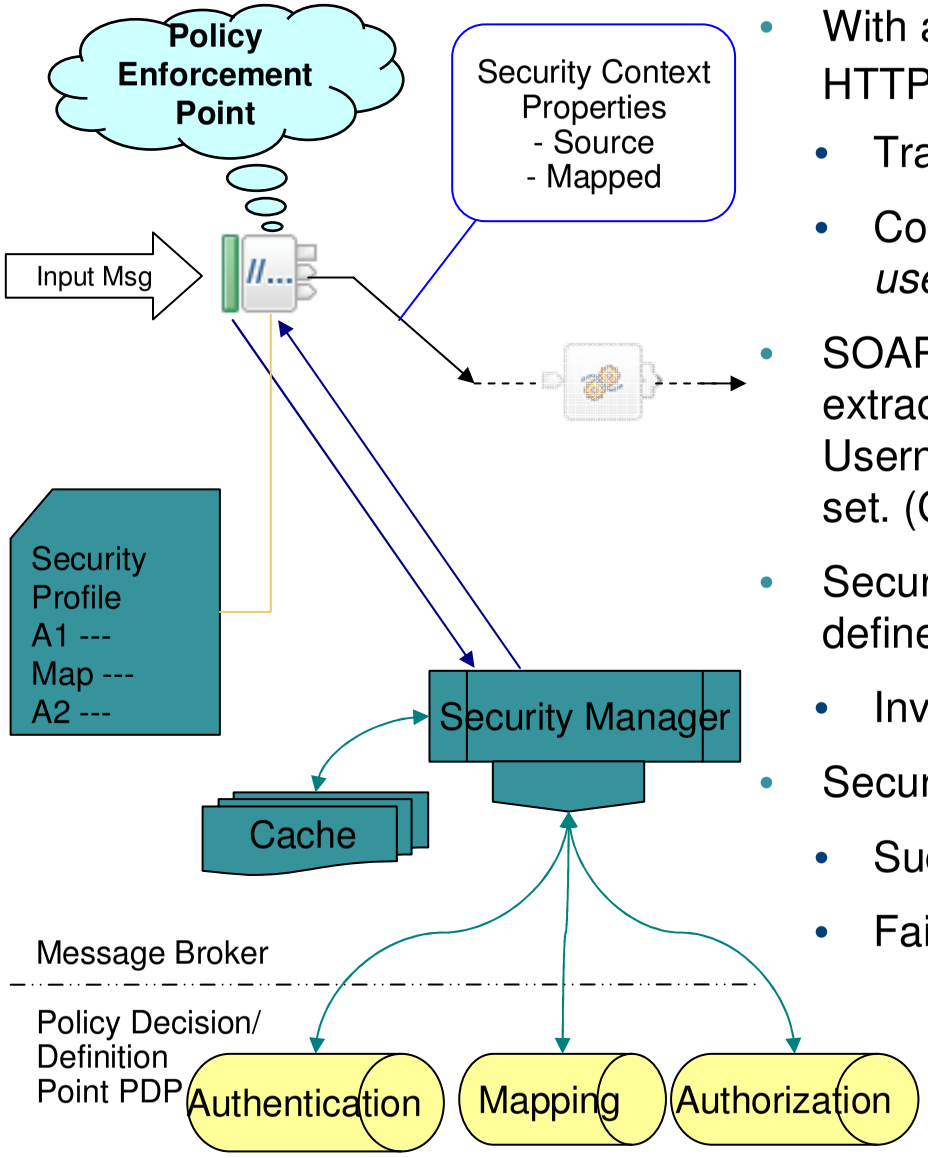
Creating security profiles using commands

- To create a new security profile
  - mqsicreateconfigurableservice <broker> -c SecurityProfiles
    –o <profile-name> -n <property-name-list> -v <property-value-list>
- To delete a security profile
  - mqsideleteconfigurableservice <broker> -c SecurityProfiles
    –o <profile-name>
- To change the values in a security profile
  - mqsichangeproperties <broker> -c SecurityProfiles
    –o <profile-name> -n <property-name-list> -v <property-value-list>
- To report the values in a security profile
  - mqsireportproperties <broker> -c SecurityProfiles –o <profile-name> -r
  - mqsireportproperties <broker> -c SecurityProfiles
    –o allReportableEntityNames -r

Associating security profiles with flows



PEP Enabled Input node operation summary



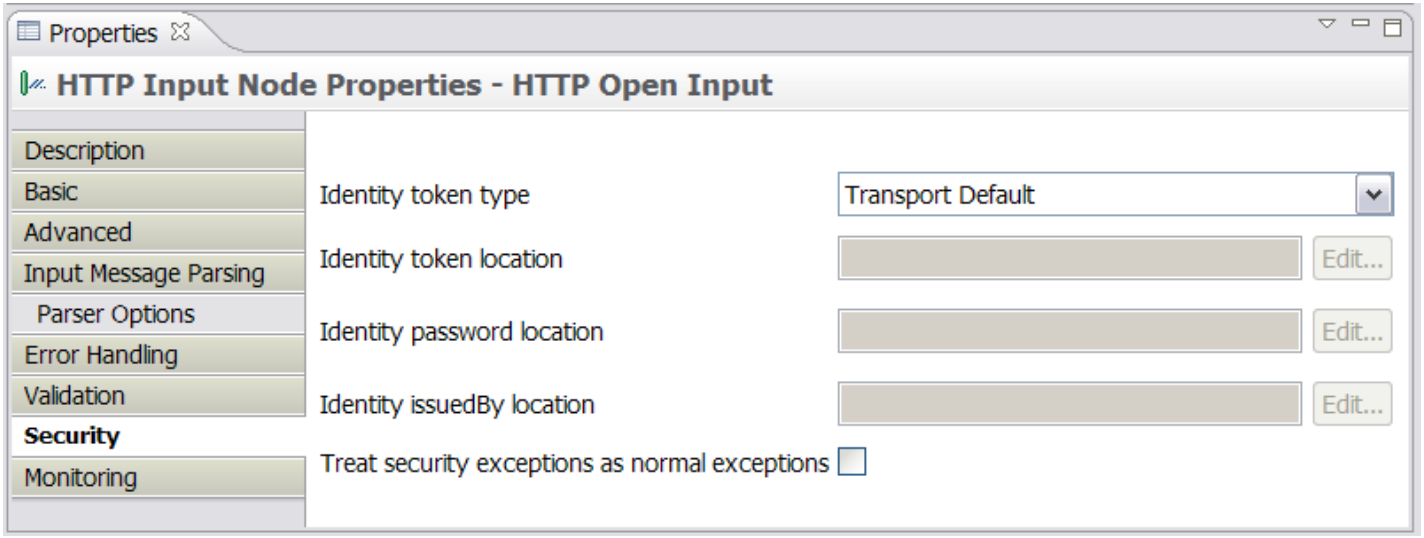
- With a Security Profile associated MQ, HTTP, SCA Input nodes extract tokens,
  - Transport Default, MQ UserID, HTTP BasicAuth
  - Configured Xpath/ESQL locations for *username, usernameAndPassword, X.509, SAML*
- SOAP nodes with a Security Profile associated extract the WS-Security token according to the Username, SAML or LTPA Policy Set and Bindings set. (Can use Transport Default if no policy)
- Security Manager enforces security operations defined in Security Profile
  - Invoke external PDP or retrieve cached decision
- Security Manager returns decision to input node
  - Success, propagate with Security context
  - Failure
    - Transport defined rejection of input message
    - Optional “Treat Security Exceptions as normal”

Notes: MQ, HTTP, SOAP nodes

- The security manager means that an input node can act as a Policy Enforcement point (PEP).
- The default locations from where to obtain the token, password and issuedBy information are transport dependent and are shown on the slides. To override the default locations, use the node location properties to specify an ESQL path or XPath to the actual location in the message header or body
- The behaviour when handling a SecurityException is transport dependent and is shown on the slides
- Note that the use of an HTTPInput node with username and password from the HTTP Authentication header and a suitable profile is equivalent to “HTTP Basic Auth” functionality
- The SOAP nodes behave in two different ways depending on whether the WS-Security protocol is being used by the message.

PEP Enabled MQ / SCA / HTTP Input node operation

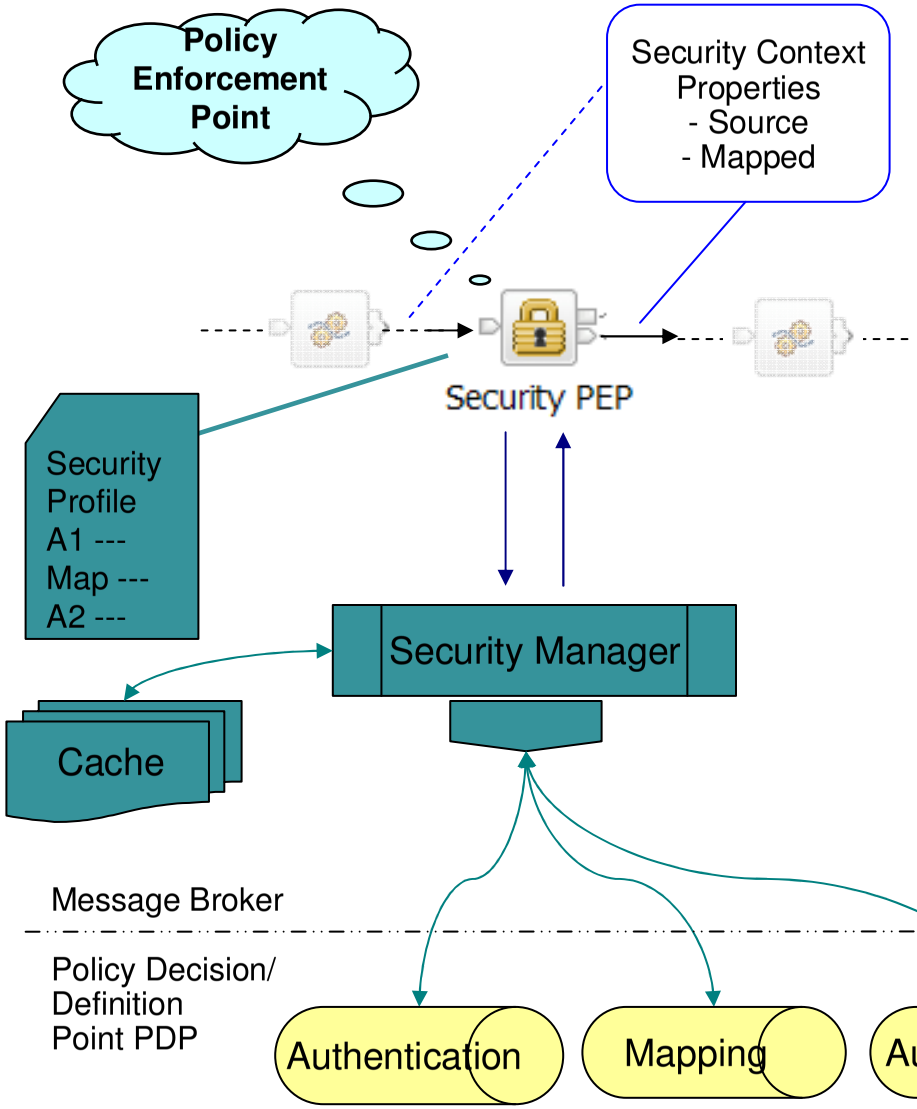
- Security Properties Page allows for configuration of
  - Token type
    - Transport Default (HTTP Basic-Auth, MQ User)
    - Username, Username + Password, SAML Assertion , X.509 Certificate
      - *Xpath/ESQL Token location of supported token type, use ‘ ’ to set a literal value*
  - Treat security exceptions as normal, causes negative security decision to propagate to failure terminal rather than inbuilt transport rejection



Notes: Node properties

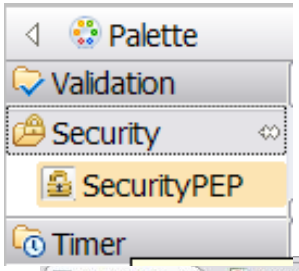
- Security properties are carried on two kinds of node, input nodes and output/request nodes.
- For input nodes, whether runtime security is configured for the node is determined by the *Security profile* property. If no security profile is specified then security is not configured. Otherwise it is the security profile that says which combination of authentication, authorization and mapping is to be performed with the identity in the message.
- The *Identity token type* property specifies how the identity appears in the message. It can be one of *Username, Username + Password*, SAML Assertion, or *X.509 Certificate* then security is configured.
- The default location in the message of the token, password and issuer is transport dependent. However the location may be overridden using the *Identity token location, Identity password location* and *Identity issuedBy location* properties.
- If a SecurityException is thrown as a result of an authentication, authorization or mapping failures, the default behaviour is that it can not be caught by exception handlers, such as wired Catch terminals. Instead the exception is always returned to the input node, where the behaviour is transport dependent. This can be overridden by the *Treat Security exceptions as normal exceptions* property, which if checked allows security failures to be handled using the usual exception handlers.
- Note that the Identity fields in the Properties folder are only set if a security profile is present for the input node.
- For output/request nodes, whether the identity is propagated with the outbound message is determined by the security profile given by the *Security profile* property. A pre-configured profile for use by output/request nodes is shipped with the broker which specifies propagation.
- Note that the *Security profile* property is ‘hidden’ but ‘configurable’ meaning that it can only be set in the broker archive (bar) file at deploy time by an administrator. It is *not* visible on the node itself. There is also a *Security profile* property on the message flow itself, which acts as a default for all nodes in the message flow that do not specify a security profile explicitly. When the flow-level property is set a node can still be configured to not have a profile (i.e. not use the flow-default value) by choosing “No Security” on it.

New Security PEP node operation summary



- With a Security Profile associated Security PEP node can be configured to use
  - Current tokens in Security Context
  - Extract from Xpath/ESQL location, for *username, usernameAndPassword, X.509, SAML, kerberosTicket, LTPA, universalWsse*
- Security Manager enforces security operations defined in Security Profile
  - Invoke external PDP or retrieve cached decision
- Security Manager returns decision to SecurityPEP node
  - Success, propagate to out terminal with Security context updated
  - Failure, propagate to failure terminal with wrapped security exception

MB 7.0.0.1 New Security PEP Node

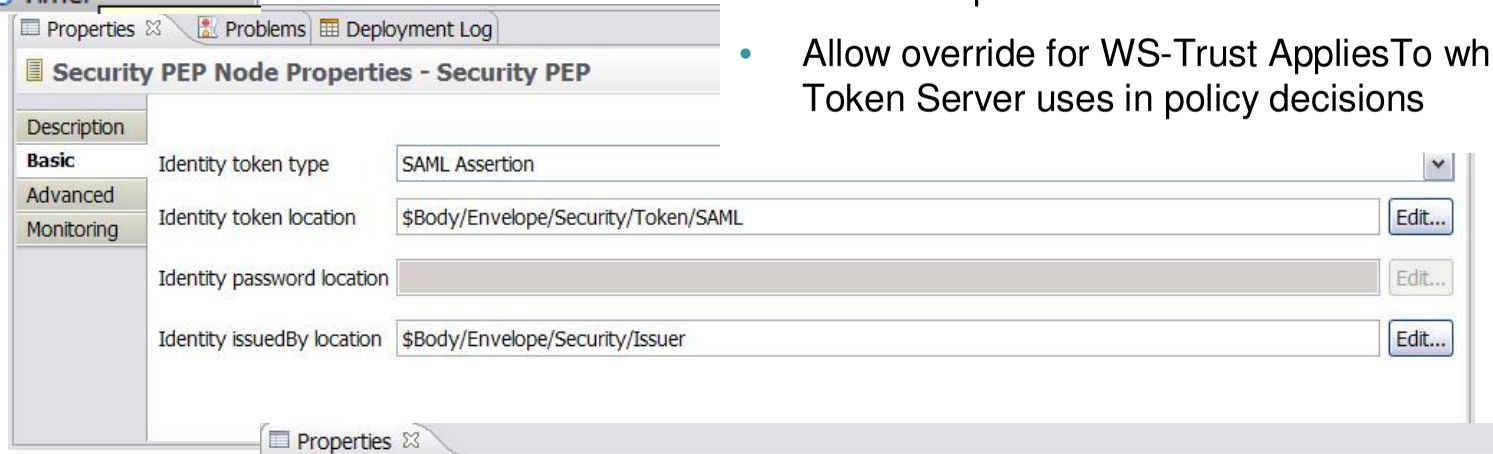


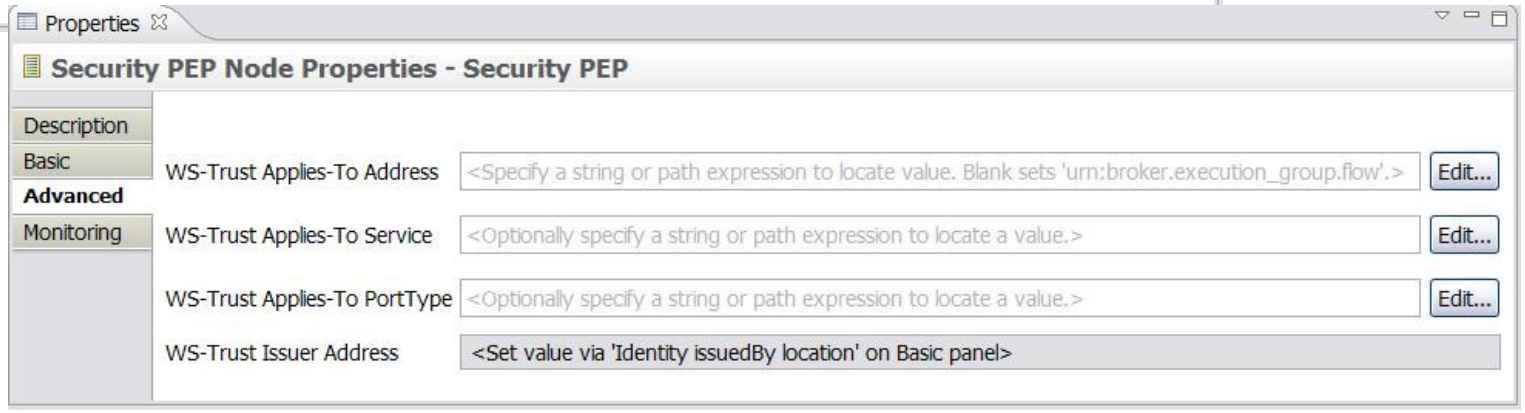
Basic Properties

- Use Current tokens or extract token using design time Xpath/ESQL locations

Advanced Properties

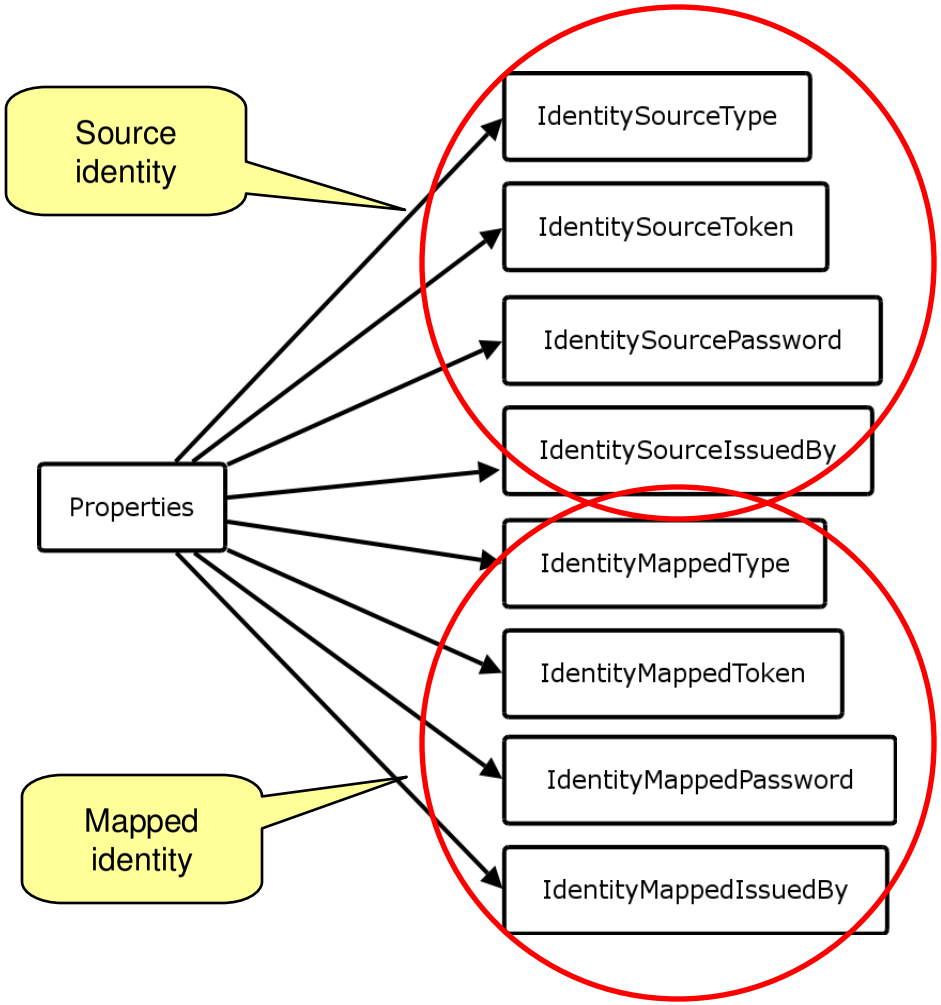
- Allow override for WS-Trust AppliesTo which a Security Token Server uses in policy decisions





- Use at any point in any flow to enforce security policy

Properties folder & Identities



- *Type contains*
  - *none, username, usernameAndPassword, X.509, SAML, kerberosTicket, LTPA, universalWsse*
- *Token* contains
  - String: username
  - Base 64 string: *X.509, kerberosTicket, LTPA*
  - *String serialization: SAML, universalWsse*
- *Password* contains
  - String: password or RACF passticket, which might be plain, masked or obfuscated
- *IssuedBy* contains
  - String: where the token was created
- Mapped used in preference to Source

Notes: Properties folder & Identities

- An identity is a piece of information which can uniquely identify an individual or object. Within the Broker identity is held in the Properties folder of the broker message tree.
- There are eight fields in the Properties folder, between them defining two identities; ‘source’ and ‘mapped’. For each of these identities, Type, Token, Password and IssuedBy fields are held.
  - The *Type* field defines the format of the Token
  - The *Token* field holds the actual token data
  - In the case of a *Username + Password* token the *Password* field will additionally contain the associated password. This could equally be a RACF Pass Ticket
    The value might be masked or obfuscated
  - The *IssuedBy* field defines where the Token was created.
- The values in the Properties are writeable, for example from ESQL

LDAP support

- Requires either
  - IBM Tivoli Directory Server
  - OpenLDAP
  - Microsoft Active Directory
- If anonymous login not permitted
  - mqsisetdbparms –n ldap::LDAP –u <username> –p <password>
  - mqsisetdbparms –n ldap::<servername> –u <username> –p <password>
- Supported token types
  - *Username*
  - *Username + Password*
- Use of security profile editor recommended

Notes: LDAP support

- Requires either IBM Tivoli Directory Server or OpenLDAP or Microsoft Active Directory .
- If your LDAP server does not permit anonymous login, you need to use the *mqsisetdbparms* command to set up the username (fully qualified) and password to be used.
- LDAP support is for token types of *Username* and *Username + Password*.
- Building LDAP configuration strings is quite complicated. If you are using commands, there are three cases. The syntax is shown for each along with real examples from IBM Blue Pages.
- Authentication only
  - *Syntax*: ldap[s]://server[:port]/baseDN[? [uid_attr] [? [base | sub]]]
  - *Example*: ldaps://bluepages.ibm.com:999/ou=bluepages.ibm.com?emailaddress
- Authentication & Authorization
  - *Syntax authn*: As above
  - Syntax *authz*: ldap[s]://server[:port]/groupDN [? member_attr]
  - *Example authz*: ldaps://bluepages.ibm.com:999/cn=HURLAB MQESB-JHRPT,ou=memberlist, ou=ibmgroups,o=ibm.com?uniquemember
- Authorization only
  - *Syntax*: ldap[s]://server[:port]/groupDN [? [member_attr] [? [base | sub] [? [x-userBaseDN=baseDN,x-uid_attr=uid_attr]]]]
  - *Example*: ldaps://bluepages.ibm.com:999/cn=HURLAB MQESB-JHRPT,ou=memberlist, ou=ibmgroups,o=ibm.com???x-userBaseDN=ou=bluepages%2co=ibm.com, x-uid_attr=emailaddress
  - Note that any commas within baseDN and uid attribute need to be replaced with "%2c".

TFIM support

- TFIM 6.1 required
- Create TFIM custom Trust Service module chains
  - Authenticate, authorize, map as necessary
  - Chain selected by *IssuedBy* value and message flow name
- Supported token types
  - *Username*
  - *Username + Password*
  - *X.509 Certificate*
- Supported identity mappings
  - *Username* to *Username*
  - *X.509 Certificate* to *Username*

Notes: TFIM support

- TFIM 6.1 is required.
- It is the responsibility of the user to customize TFIM to perform the required action against the identity. This is performed using Trust Service module chains to authenticate or authorize or map the identity.
- The chain to use is determined by a combination of the source identity *issuedBy* value and the name of the message flow, expressed as *<broker-name>.<exec-grp-name>.<msg-flow-name>*.
- TFIM support is for token types of *Username, Username + Password* and *X.509 Certificate*.
- As far as identity mapping is concerned, it is possible to map a username to another username, and an X.509 certificate to a username. But it is not possible to map a username to an X.509 certificate (TFIM does not issue X.509 certificates).
- When mapping from an X.509 certificate, TFIM can validate the certificate, but can not be used to verify the identity of the original sender. This would have to be done elsewhere, for example, using WS-Security support for digital signatures using a SOAPInput node.

WS-Trust v1.3 Security Token Server

- Requires any WS-Trust v1.3 compliant provider
  - TFIM 6.2 supported and tested
- Supported operations
  - Identity Authentication or token Validation
  - Identity Mapping or token Issuance/Exchange
  - Authorization
- Supported token types - all
  - Username + Password, SAML, Kerberos, LTPA, RACF PassTicket, X509
  - Universal WSSE .. Any token that can be put in a WSSE header sub tree
- Securing the STS connection
  - SSL and/or Basic-Auth

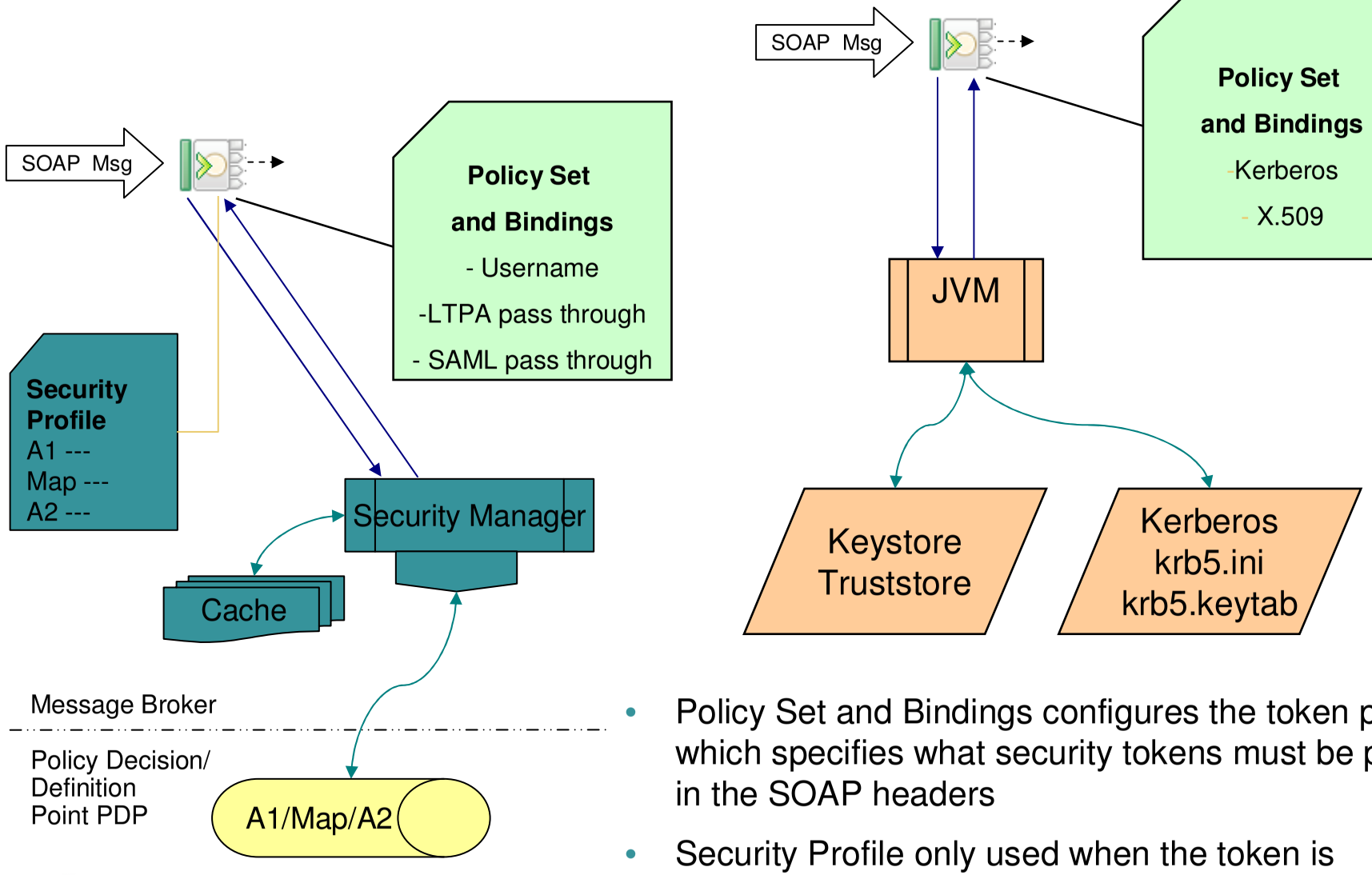
WS-Security

- Message based security
  - Fine granularity
- Parts of the message may be encrypted in different ways with different keys
- Parts of a message may be (multiply) encrypted and signed
  - On a need to know basis
- WS-Security can be used in insecure transports
- SOAP nodes support WS-Security
  - Configured using policy sets and bindings

WS-Security

- Key areas covered for WS-Security
  - Authentication (Tokens)
  - Message Part Protection
    - XML Signature (Signed)
      - *To ensure data integrity*
      - *Message can be read but not changed without detection*
    - XML Encryption (Encrypted)
      - *To ensure confidentiality*
      - *Message can not be read or changed*

SOAP Input node operation summary



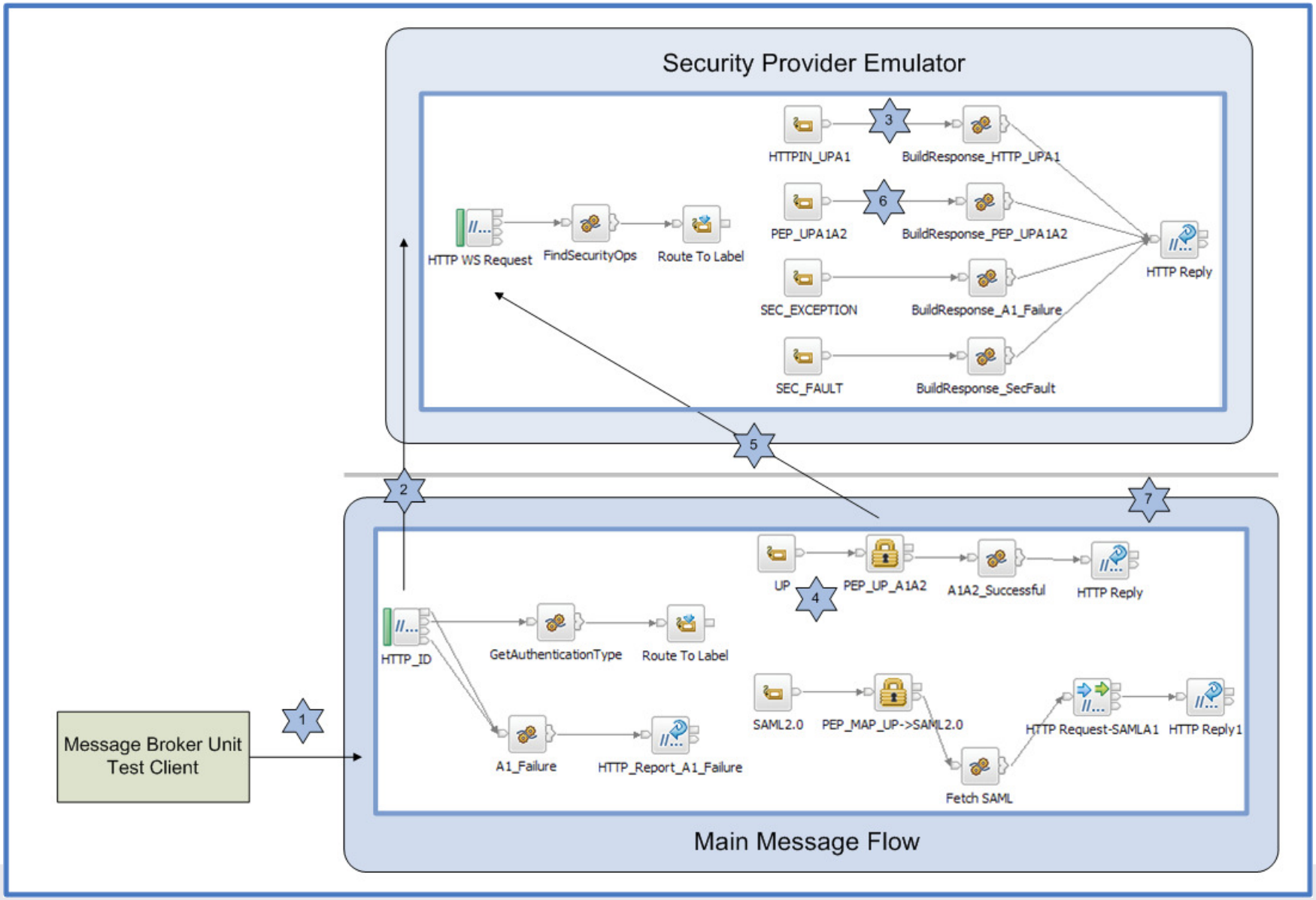
- Policy Set and Bindings configures the token profile which specifies what security tokens must be present in the SOAP headers
- Security Profile only used when the token is processed by the Security Manager

Notes: PEP Enabled SOAP Input node operation

- Policy set and Bindings define the WS-Security profile token binding
- Security processing is either
  - External Security Policy Decision Point
    - Username and password LDAP or WS-Trust v1.3 STS
    - SAML / LTPA pass through WS-Trust v1.3 STS

    Security profile defines the policy decision for any or all of authentication, mapping and/or authorization which are delegated to specified provider via security manager
  - Kerberos .. Direct to Key Distribution Centre via JVM
    Can propagate the service principal, but not the ticket
  - X.509 To Broker Key and Trust stores
  - If no Policy set and Bindings, default pick up HTTP Basic-Auth header
- Security rejection always handled through SOAP Fault response to client

Demo / Sample


Security Provider Emulator
HTTPIN_UPA1
BuildResponse_HTTP_UPA1
HTTP WS Request
FindSecurityOps
Route To Label
PEP_UPA1A2
BuildResponse_PEP_UPA1A2
HTTP Reply
SEC_EXCEPTION
BuildResponse_A1_Failure
SEC_FAULT
BuildResponse_SecFault
UP
PEP_UP_A1A2
A1A2_Successful
HTTP Reply
HTTP_ID
GetAuthenticationType
Route To Label
SAML2.0
PEP_MAP_UP->SAML2.0
HTTP Request-SAMLA1
HTTP Reply1
A1_Failure
HTTP_Report_A1_Failure
Fetch SAML
Message Broker Unit Test Client
Main Message Flow
1
2
3
4
5
6
7

Summary

- Multiple aspects to Message Broker Security
  - Administration security
    - Security Exits
    - Channel Security
  - Runtime security
    - Transport security - SSL
    - Database Security
  - Message flow security
    - Security Manager
      - Identity propagation
    - WS-Security

- New PEP node in v7 FP1 significantly enhances Message Flow security options

Copyright and Trademarks