4114 - Taking CICS Web Security to the Next Level

Peter Havercan

CICS & Enterprise Transformation

Objectives

This presentation describes new Security features available in CICS Transaction Server 3.1 and 3.2

Session Agenda

- Security facilities in CICS TS 3.1
  - Support for Transport Layer Security (TLS)
  - Specifying cipher suites
  - Performance enhancements
  - Changes to revocation processing
  - Support for mixed case passwords
  - Web Services Security (WSSE)
- Security facilities in CICS TS 3.2
  - Basic Authentication enhancements
  - Resource Security for Document Templates
  - Transaction user’s access to HFS files is now checked

Session Agenda - Notes

CICS TS 3.1 implemented support for the Transport Layer Security (TLS) 1.0. This new protocol, based on Secure Sockets Layer (SSL) 3.0 is provided by the System SSL component of z/OS. CICS TS 3.1 supports the 128-bit and 256-bit Advanced Encryption Standard cipher suites. CICS also allows you to specify a range of cipher suites on the TCPIPSERVICE, CORBASERVER and URIMAP resource definitions, and on the IPCONN resource definition in CICS TS 3.2.

CICS TS 3.1 made a number of changes to provide for an increased number of SSL connections and to reduced the performance cost of re-using an SSL connection. CICS now allows SSL session IDs to be shared across a sysplex. CICS has also restructured SSL processing to exploit the Open Transaction Environment (OTE) with its S8 TCBs.

CICS TS 3.1 has a new utility transaction, CCRL, which can be invoked from a terminal or started task, which will download a certificate authority's revocation lists into a local LDAP server for use by PKI processing. CICS now also checks the revoked status of a userid and its connection to its default group.

CICS TS 3.1 will support passwords with mixed case if the underlying security manager allows it.

CICS TS 3.1 delivered support for Web Services Security in a post-GA PTF.

CICS TS 3.2 has made extensive changes is support of HTTP Basic Authentication, for both client and servers.
CICS TS 3.2 now supports Resource Security for Document templates, and controls access from Web applications to UNIX files (HFS and zFS) when used to deliver static content.

CICS support for Transport Layer Security

- TLS is the latest version of the Secure Sockets Layer protocol
  - Specification documented in RFC 2246
- z/OS 1.4 (and later) System SSL incorporates:
  - TLS 1.0
  - SSL 3.0
  - SSL 2.0 (but CICS will not allow this protocol)
- CICS now uses new System SSL APIs
  - Those used in CICS TS 2.3 are now deprecated by z/OS

CICS support for Transport Layer Security - Notes

CICS TS 3.1 provides support for the Transport Layer Security (TLS) 1.0. This new protocol, based on Secure Sockets Layer (SSL) 3.0 is provided on the z/OS platform by the System SSL component.

The System SSL APIs that CICS used in previous releases are being deprecated, so CICS is using new System SSL APIs instead. This also provides better support for Certificate Revocation Lists and sysplex-wide session-id caching, which CICS now exploits.

CICS no longer supports SSL version 2. It is not secure enough for modern applications and has been superseded almost everywhere by SSL Version 3 or TLS. During testing it was discovered that some browsers may drop back to SSL V2 when they are unable to negotiate the most secure cipher suites in V3 or TLS, resulting in an unintended fallback in security when only the very highest security was intended.

Cipher Suites

- Support for AES cipher suites
  - 128-bit and 256-bit encryption
- Specification of cipher suites to be used for encryption
  - Allows for a minimum and maximum level of encryption
    - If partner doesn't support the selected choices no connection will be established
    - Specified on:
      - TCPIPSERVICE for inbound HTTP and IIOP requests
      - CORBASERVER for outbound IIOP requests
      - URIMAP for outbound HTTP requests (and EXEC CICS WEB OPEN)
      - IPCONN for IP interconnectivity (new in CICS TS 3.2)

Cipher Suites - Notes

CICS TS 3.1 now supports the 128-bit and 256- bit Advanced Encryption Standard cipher suites.

CICS will also allow you to specify a range of cipher suites on the TCPIPSERVICE (for inbound requests), CORBASERVER (for outbound IIOP) and URIMAP (for outbound HTTP) resource definitions. CIPHERS can also be specified on the new EXEC WEB OPEN command for outbound HTTP connections.

CICS will only negotiate a session with a partner that supports at least one of the selected cipher suites.

Cipher Suites

- Cipher suites are identified by hexadecimal codes:

Cipher	Encryption Algorithm	Key length	MAC	Key Exchange
01	None		MD5	RSA
02	None		SHA	RSA
03	RC4	40	MD5	RSA
04	RC4	128	MD5	RSA
05	RC4	128	SHA	RSA
06	RC2	40	MD5	RSA
09	DES	56	SHA	RSA
0A	3-DES	168	SHA	RSA
2F	AES	128	SHA	RSA
35	AES	256	SHA	RSA
30	AES	128	SHA	ephemeral-DH with RSA cert
31	AES	128	SHA	fixed-DH with RSA cert
36	AES	256	SHA	fixed-DH with DSA cert

Cipher Suites - Notes

- Each cipher suite code represents a combination of an encryption algorithm, an encryption keylength, a MAC (message authentication code) hashing algorithm, and a key-exchange technique,
  - RC2, RC4 are Rivest cryptographic algorithms
  - DES is the Data Encryption Standard
  - 3-DES or Triple DES is DES applied three times
  - AES is the Advanced Encryption Standard
  - RSA is the Rivest, Shamir, and Adleman algorithm
  - DSA is the Digital Signature Algorithm
  - DH is Diffie-Hellman key exchange
    - can use fixed or ephemeral keys and RSA or DSA certificates
- Cipher suite 00 is supported by System SSL. This cipher suite indicates no encryption and no MAC algorithm. This is allowed by the SSL and TLS protocol, but is **not recommended**. CICS does not allow you to specify this cipher suite code.
- The full range of cipher suites is available on z/OS 1.7 is:
  - 0504353637383922F303132330A1613100D0915120F0C03060201

Cipher Suites…

- Range of available cipher suites for CICS to use is specified in the system initialization parameter
  - ENCRYPTION={<u>STRONG</u> | MEDIUM | WEAK}
    - For compatibility with CICS TS V2
      - ENCRYPTION=NORMAL will be treated as ENCRYPTION=MEDIUM (but is no longer the default)
- Selection of cipher suites and order of preference
  - Specified in CIPHERS attribute
    - Two-digit hexadecimal codes indicate cipher suites
    - Order determines preference
      - e.g. `CIPHERS(352F0A0504)`
    - Replaces the PRIVACY parameter

Cipher Suites - Notes

The default range of cipher suites available to CICS to use when negotiating a session is specified in the Systems Initialization Table (SIT) by the ENCRYPTION parameter.

The default has changed in CICS TS 3.1 from ENCRYPTION=NORMAL to ENCRYPTION=STRONG.

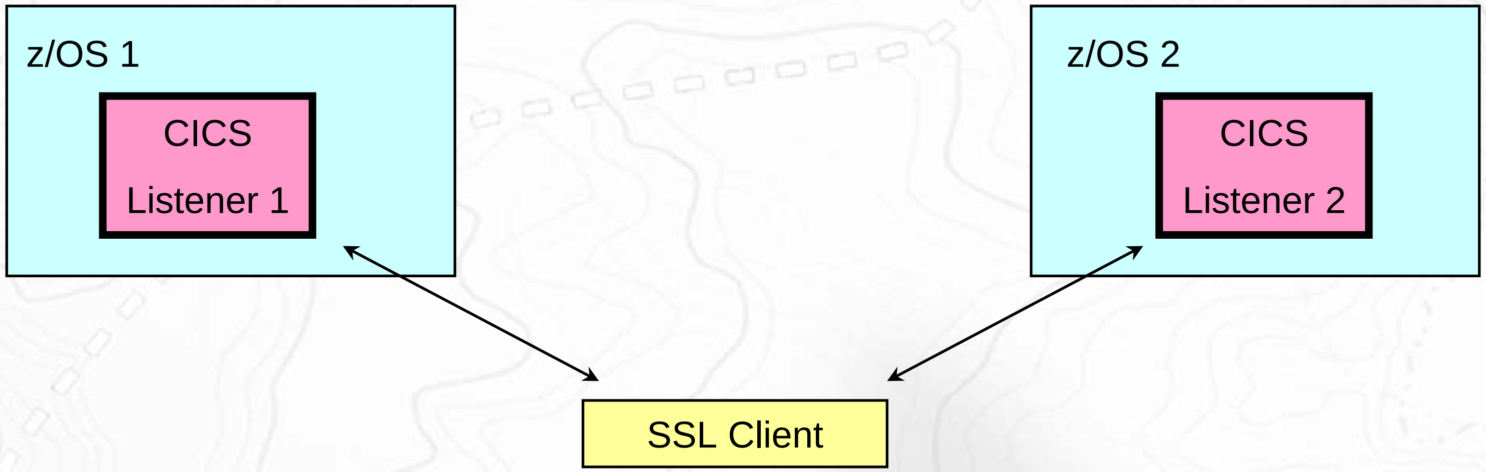
ENCRYPTION=MEDIUM has replaced ENCRYPTION=NORMAL. To maintain compatibility with prior releases a specification of ENCRYPTION=NORMAL will be accepted as ENCRYPTION=MEDIUM. The actual ranges of cipher suites belonging to each level are listed on the following pages.

The actual selection of the cipher suites to be used is specified on the resource definitions: TCPIPSERVICE, CORBASERVER and URIMAP. The CIPHERS parameter is a list of two-digit hexadecimal codes identifying the specific cipher suites. The codes are the same as those specified in the TLS specification. The order in which the codes are listed in the CIPHERS parameter indicates your preference.

The CIPHERS parameter replaces the PRIVACY parameter. The PRIVACY parameter will still be shown on the RDO panels, but it will not be possible to modify it except by changing the CIPHERS list. INQURE TCPIPSERVICE PRIVACY will still function.

Performance Enhancements

- SSL Sysplex Caching
  - In CICS TS V2 the SSL session id is cached locally in each CICS region
    - If the same client connects to a different CICS region a full SSL handshake is required
      - Impacts cloned CICS listener regions



Performance Enhancements - Notes

In the earlier CICS SSL implementation, a client who successfully connects to one CICS regions and then connects to another CICS region will be required to go through a full SSL handshake in both cases. This is because the SSL cache is local to a CICS address space.

Performance Enhancements

- SSL Sysplex Caching…
  - Makes server session information across a sysplex
    - Requires all systems in the sysplex to use the same ESM
    - Requires SSL Started Task (GSKSRVR) to be implemented
      - Supports TLS 1.0 and SSL 3.0 protocols
      - GSKSRVR Environment Variables
        - GSK_LOCAL_THREADS: number of threads
        - GSK_SIDCACHE_SIZE: sysplex session cache size in megabytes
        - GSK_SIDCACHE_TIMEOUT: session cache entry timeout in minutes
    - Enable CICS to use sysplex caching
      - SIT parameter
        - SSLCACHE={<u>CICS</u> | SYSPLEX }

Performance Enhancements - Notes

The sysplex session cache support makes SSL server session information available across the sysplex. An SSL session established with a server on one system in the sysplex can be resumed using a server on another system in the sysplex as long as the SSL client presents the session identifier obtained for the first session when initiating the second session. SSL V3 and TLS V1 server session information can be stored in the sysplex session cache while SSL V2 server session information and all client session information is stored only in the SSL cache for the application process.

The SSL started task (GSKSRVR) provides sysplex session cache support.

In order to use the sysplex session cache, each system in the sysplex must be using the same external security manager (for example, z/OS Security Server RACF) and a userid on one system in the sysplex must represent the same user on all other systems in the sysplex (that is, userid XYZ on System A has the same access rights as userid XYZ on System B). The external security manager must support the RACROUTE REQUEST=EXTRACT,TYPE=ENVRXTR and RACROUTE REQUEST=FASTAUTH functions.

SSLCACHE={CICS| SYSPLEX} Specifies whether SSL is to use the local or sysplex caching of session ids. Sysplex caching is only useful if multiple CICS socket-owning regions accept SSL connections at the same IP address.

SSL Open Transaction Environment Exploitation

- SSL implementation prior to TS 3.1
  - Used a separate S8 TCB for each SSL requests
    - Fixed pool size
      - Specified by SSLTCBS initialization parameter
    - Each TCB has its own LE enclave
      - Consumes below-the-line storage
    - S8 TCB is assigned for the duration of the requesting task
      - Use across CWXN and CWBA transactions

SSL Open Transaction Environment Exploitation - Notes

The current CICS implementation for SSL uses a fixed pool of S8 TCBs to handle requests. The pool size is specified by the system initialization parameter SSLTCBS. The maximum number of SSL TCBs is 255 per CICS address space.

Each S8 TCB has its own LE enclave and the TCB is assigned for the duration of the requesting task, even though it is only active for a very short period during that time.

SSL Open Transaction Environment Exploitation…

- OTE implementation
  - SP TCB
    - Created when KEYRING is specified in the SIT
    - Owns the LE enclave and SSL cache
  - S8 TCBs
    - Run as a UNIX pthreads within the single LE enclave
    - Variable pool size
      - Controlled by MAXSSLTCBS parameter
    - Assigned for the duration of the request only

SSL Open Transaction Environment Exploitation - Notes

CICS 3.1 uses the open transaction environment (OTE) to manage SSL connections.

There is a new open TCB mode called SP, that is used for socket pthread owning tasks. The SSL pool is managed by the one SP TCB that runs in the CICS region. The SP TCB and subsequent SSL pool will only be created if the KEYRING parameter is present in the SIT, which indicates that SSL processing is required.

Each SSL connection uses an S8 TCB, which is allocated from the SSL pool. The S8 TCBs run as UNIX pthreads. This allows many more simultaneous SSL connections in CICS than the limit of 256 in previous releases. MAXSSLTCBS has a limit of 1024 TCBs. All of the S8 TCBs run within a single enclave, which is owned by the SP TCB and contains the SSL cache.

S8 TCBs are now only locked to a transaction for the period that it needs to perform SSL functions. After the SSL negotiation is complete, the TCB is released back into the SSL pool to be reused.

Certificate Revocation Lists

- Digital Certificates
  - Are used in the process of validating signed data or securely transmitting encryption keys
  - Binds a subject's "distinguished name" to a public key
  - Have a limited lifetime
    - Start and end date specified in the certificate's contents
  - Every certificate has a unique serial number
- Certificate Revocation Lists
  - The issuing Certificate Authority can "revoke" a certificate by publishing its serial number in a Certificate Revocation List (CRL)
  - CRLs are periodically published on the Certificate Authorities' websites
  - Can be republished locally in Lightweight Directory Access Protocol (LDAP) directories

Certificate Revocation Lists - Notes

To make an environment secure, you must be sure that any communication is with "trusted" sites whose identity you can be sure of. SSL uses certificates for authentication -- these are digitally signed documents which bind the public key to the identity of the private key owner. Authentication happens at connection time, and is independent of the application or the application protocol. Authentication involves making sure that sites with which you communicate are who they claim to be. With SSL, authentication is performed by an exchange of certificates, which are blocks of data in a format described in ITU-T standard X.509. The X.509 certificates are issued, and digitally signed, by an external authority known as a certificate authority.

Certificates have a limited lifetime when issued. A certificate can also be revoked by the issuing authority. Certificate authorities are independent bodies who act as the trusted third parties, by issuing certificates for use by others. Before issuing a certificate, a certificate authority will examine the credentials of the person or organization that has requested the certificate. When the certificate has been issued, information about it is held on a publicly accessible repository. Users can consult the repository to check the status and validity of any certificates received.

Certificate authorities will periodically create and publish Certificate Revocation Lists (CRLs). The CRLs can be downloaded and used as part of the certificate validation process. The downloaded information can be stored locally in an LDAP server and is used by z/OS Public Key Infrastructure services.

Certificate Revocation Lists

- CICS provides a utility transaction to download Certificate Revocation Lists and store them in LDAP
  - CCRL transaction
    - Can be invoked from a terminal: CCRL *url-list*

```
CCRL http://crl.geotrust.com/crls/secureca.crl
     http://crl.verisign.com/ATTClass1Individual.crl
```

    - Can be invoked as a started task:

```
EXEC CICS START TRANSID(CCRL)
     FROM(‘http://crl.geotrust.com/crls/secureca.crl
      http://crl.verisign.com/ATTClass1Individual.crl’)
     LENGTH(89) INTERVAL(960000)
```

Certificate Revocation Lists - Notes

Certificate revocation lists are available from certificate authorities such as Verisign, Geotrust, and Equifax. They are kept in CRL repositories that are available on the world wide web and can be downloaded and stored in the LDAP server.

To populate the LDAP server and update certificate revocation lists, use the CICS-supplied transaction CCRL. You can run the CCRL transaction from a terminal or using a START command. Use the START command to schedule regular updates.

From a terminal, enter CCRL *url-list* where *url-list* is a space-delimited list of URLS that specify the locations of the certificate revocation lists that you want to download.

To use a START command, enter EXEC CICS START TRANSID(CCRL) FROM (*url-list*) LENGTH (*url-list-length*) [INTERVAL(*hhmmss*) | TIME(*hhmmss*)] where *url-list* is a space-delimited list of URLs from where certificate revocation lists can be downloaded, *url-list-length* is the length of the URL list, and *hhmmss* is the interval or expiration time at which the CCRL transaction is to be scheduled.

The location of the LDAP server is obtained from the LDAPHOST field of the CRLSERVER profile, but the bind authentication details are obtained from the person who wishes to perform the update.

Changes to Revocation Processing

- Previous releases of CICS (Version 2)
  - Do not check the revoked status of a USERID for:
    - EXEC CICS VERIFY
    - ATTACHSEC(VERIFY)
    - START with USERID
  - Do not check if a connection to a GROUP was revoked
- Revoked status of a user ID or a user’s group connection is now honored by CICS TS Version 3

Changes to Revocation Processing - Notes

The current implementations of ATTACHSEC(VERIFY), EXEC CICS VERIFY PASSWORD and EXEC CICS START TRANSID() USERID() do not check the revoked status of the USERID being verified. After the USERID and password have been verified by either of these techniques, CICS allows the user to reuse a security environment that had been built earlier and represented by a previously created ACEE. If the USERID has been revoked since that security environment had been built, the pre-built security environment still persists and CICS is therefore allowing users to continue to reuse it even though they are revoked.

CICS TS 3.1 now honors the revoked status of a USERID including the case where a user’s connection to their default GROUP is revoked.

CICS support for mixed case passwords

- Requires z/OS 1.7 or later
  - Mixed case support is enabled by RACF command:

    SETROPTS PASSWORD(MIXEDCASE)
- Password uppercasing in CICS is now conditional
  - Omitted if mixed case support is present in ESM
- CESN enhancements
  - Will not translate password fields to upper case
    - Independent of terminal UCTRAN setting
    - Similar to CEDA mixed-case support
  - Will issue an appropriate caution message:
    - **DFHCE3540 Ensure that passwords are entered in the correct case.**
- Retrofitted to earlier CICS releases in PK08496

CICS support for mixed case passwords - Notes

CICS TS 3.1 provides support for mixed case passwords.

Mixed case password support will require z/OS 1.7 which will provide Security Server support for mixed case passwords. (This support is mentioned on the z/OS 1.7 preview section of the z/OS 1.6 announcement letter.) CICS TS 3.1 will take advantage of this operating system enhancement if it is available. If the support is not available, CICS will always uppercase the password before presenting it to the External Security Manager.

CESN, the sign-on transaction, will be enhanced so as NOT to translate password fields into uppercase even if UCTRAN(YES) is specified for the terminal. This is similar to what CEDA currently does for mixed-case fields in CICS TS 2.3. If necessary, the uppercasing is deferred until just before the password is presented to the External Security Manager.

CESN will also be enhanced to issue a cautionary message if mixed case password support is available. The message "Ensure that passwords are entered in the correct case" will be issued.

Security for document templates

- Resource level security added for DOCTEMPLATEs
  - Documents delivered as a static response to a Web request
    - TEMPLATE name specified on the URIMAP definition
  - Document templates used by an application program
  - CREATE, INQUIRE and DISCARD DOCTEMPLATE commands
  - EXEC CICS DOCUMENT CREATE and INSERT commands

Notes

You can apply access controls to individual CICS document templates. Security checking for this resource is applied using the XRES system initialization parameter, which is set to YES by default. You can use this capability to secure individual Web pages delivered as static responses (using URIMAP definitions). You can also secure document templates that are used by application programs, either for Web delivery as part of an application-generated response, or for any other purpose.

The XRES system initialization parameter activates security checking for CICS document templates. The default setting for this system initialization parameter is YES, meaning that each time a document template is requested, CICS calls the external security manager to check that the user ID associated with the transaction is permitted to access the template. When YES is specified, the default resource class name RCICSRES and grouping class name WCICSRES are used. Alternatively, you can specify a different resource class name. If you set XRES to NO, no security checks are performed for document templates.

- Access to CICS document templates is controlled in the following cases:
- Document templates delivered as a static response to a Web client's request (specified on the TEMPLATENAME attribute of the URIMAP definition for the request).
- Document templates delivered as part of an application-generated response to a Web client's request (used by an application program that handles the request).
- All EXEC CICS CREATE, INQUIRE, and DISCARD DOCTEMPLATE commands.
- All EXEC CICS DOCUMENT INSERT and CREATE commands with the TEMPLATE option.

Security for document templates…

- Security name passed to the ESM is:
  - SECPRFX value if present + "." +
    - e.g. TEST versus PROD
  - "DOCTEMPLATE." +
    - Fixed name to identify the resource type
  - Resource definition name for the document template
    - e.g. HELLO
  - Profile name defined in the RCICSRES class
    - e.g. TEST.DOCTEMPLATE.HELLO

Notes

When calling the external security manager, CICS uses the name of the DOCTEMPLATE resource definition for the CICS document template, prefixed by its resource type, DOCTEMPLATE. For example, for a document template whose resource definition is named “WELCOME", the profile name passed to the security manager is DOCTEMPLATE.WELCOME. If you have used the system initialization parameter SECPRFX to add an additional prefix specific to the CICS region, this prefix is also used. For example, if the document template is in a development region where SECPRFX is set to TEST, then the profile name TEST.DOCTEMPLATE.WELCOME is passed to the security manager. You need to set up the profile names using this format.

The EXEC CICS DOCUMENT commands reference document templates using the 48-character name of the template (as specified in the TEMPLATENAME attribute of the DOCTEMPLATE resource definition). However, security checking for these commands uses the name of the DOCTEMPLATE resource definition that corresponds to the TEMPLATENAME attribute. This means that you only need to set up one profile name for each document template, using the name of the DOCTEMPLATE resource definition, and not the TEMPLATENAME attribute.

Note: Document templates can be retrieved from a variety of sources, including partitioned data sets, CICS programs, CICS files, z/OS UNIX System Services HFS files, temporary storage queues, transient data queues, and exit programs. When resource security checking is carried out for a document template, CICS does not perform any additional security checking on the resource that supplies the document template, even if resource security is specified for that type of resource in the CICS region.

Security for document templates…

- Setting up for DOCTEMPLATE security
  - Specify SEC=YES in the CICS systems initialization table
    - Optionally specify SECPRFX to distinguish between regions
  - Specify XRES=YES to activate the resource class names of RCICSRES and WCICSRES
    - Optionally specify XRES=name for a user defined resource class
  - Define the resource or the grouping profiles to your ESM
    - RDEFINE RCICSRES (DOCTEMPLATE.doc1, … DOCTEMPLATE.docn) UACC(NONE) NOTIFY(userid)
    - PERMIT DOCTEMPLATE.doc1 CLASS(RCICSRES) ID(group1, group2) ACCESS(READ)
  - Specify RESEC(YES) on the TRANSACTION definition
    - RESSEC(YES) is the default for the CWXN transaction
  - RCICSRES and WCICSRES classes defined in RACF APAR OA20162

Notes

To implement security for CICS document templates, follow these steps.

Ensure RACF APAR OA20162 is applied.

Define profiles to RACF in the default RCICSRES resource class or WCICSRES grouping class, or their equivalents if you have user-defined resource class names. For the profile names, use the name of the DOCTEMPLATE resource definition, prefixed by the resource type DOCTEMPLATE, and any additional prefix specified by the SECPRFX system initialization parameter for the CICS region. For example, use the following commands to define document templates in the RCICSRES class, and authorize users to use them to assemble documents:

```
  RDEFINE  RCICSRES (DOCTEMPLATE.doc1, DOCTEMPLATE.doc2, .., DOCTEMPLATE.docn) UACC(NONE)
  PERMIT   DOCTEMPLATE.doc1 CLASS(RCICSRES) ID(group1, group2) ACCESS(READ)
```

To define document templates as members of a profile in the WCICSRES resource grouping class, with an appropriate access list, use the following commands:

```
  RDEFINE  WCICSRES (doc_groupname) UACC(NONE)
                    ADDMEM(DOCTEMPLATE.doca, DOCTEMPLATE.docb, ..., DOCTEMPLATE.docz)
  PERMIT doc_groupname CLASS(WCICSRES) ID(group_userid) ACCESS(READ}
```

Specify SEC=YES as a CICS system initialization parameter (and SECPRFX if you define profiles with a prefix).
Specify XRES=YES for the default resource class names of RCICSRES and WCICSRES, or XRES=name for user-defined resource class names. XRES=YES is the default.
Specify RESSEC(YES) in the TRANSACTION resource definition of the transactions that access the CICS document templates. For CICS Web support, the transaction for all static responses is CWXN, or an alternate transaction that you have specified in place of CWXN using the TRANSACTION attribute on your TCPIPSERVICE definitions. The transaction for application-generated responses is an alias transaction, which can be specified in the URIMAP definition for the request or set by an analyzer program, and defaults to CWBA. **RESSEC(YES) is specified in CWXN as supplied by CICS**, but CWBA specifies RESSEC(NO), and for TRANSACTION resource definitions in general the default is RESSEC(NO).

Security for document templates…

- Static delivery of DOCTEMPLATEs is done in CWXN
  - CICS-supplied definition of CWXN has been changed to have RESSEC(YES)
  - CWXN definition is customizable if you copy it into your own RDO group

Notes

Resource Security behavior always depends on the RESSEC definition of the transaction accessing the resource.

CICS has changed the RESSEC specification for CWXN to RESSEC(YES), but this is unusual for a CICS system transaction (Category 1). One side effect of this is that, if ever CWXN accesses any other resources than doctemplates, resource security will apply to those accesses too.

Security for USS files

- Access to USS files can be controlled based on USERID
  - Only for CICS Web clients
  - Only for pages delivered as static content
    - Specified in a URIMAP definition
- HFS (and zFS) resource security is activated by:
  - XHFS parameter in the systems initialization table
  - Security check will be made on the Web client USERID
    - USERID from basic authentication or a client certificate

Notes

Files stored in UNIX System Services can be used to supply Web pages through CICS Web support, as static responses provided by URIMAP definitions. When access control for USS files is specified, you can control access to USS files on the basis of the user IDs for individual Web clients. Access control for USS files is enabled by default.

Access control for USS files is activated by the XHFS system initialization parameter. The default for this parameter is YES, meaning that resource security for USS files is active. If you do not want resource security for USS files, set this system initialization parameter to NO. The RESSEC option on the transaction resource definition does not affect this security checking.

The CICS region user ID must always have a minimum of read access to all USS files that it uses for CICS Web support, and the directories containing them. The user ID of the Web client is only used when accessing USS files as a static response, but the CICS region user ID applies to all other attempts to access the USS file. If the CICS region user ID does not have permission to access the file, even an authorized Web client will be unable to view it. This is the case even when the USS file is defined as a CICS document template.

Security for USS files…

- Security for USS files is based on:
  - A USS UID associated with an ESM USERID
    - You may use ESM grouping to allow a number of USERIDS to have the same permissions
  - Permissions to directories and files

Notes

All Web clients who use a connection with basic authentication or client certificate authentication and attempt to access any USS files, must have a user profile in the security manager which contains a valid z/OS UNIX UID, and connects to a RACF group with a valid z/OS UNIX GID.

To be able to view a web page derived from an USS file, Web clients who use a connection with basic authentication or client certificate authentication must have read permissions to the file and to the directories containing it, either individually or through the RACF groups to which they are connected.

Security for USS files…

- Setting up CICS USS security
  - Ensure that CICS:
    - Has READ access to all directories and files in the path
  - Ensure that all USERIDs
    - Have a USS UID and GID assigned
    - Have permission to READ the appropriate directories and files
  - Specify SEC=YES in the SIT
  - Specify XHFS=YES in the SIT

Notes

The CICS region user ID must always have a minimum of read access to all USS files that it uses for CICS Web support, and the directories containing them. The user ID of the Web client is only used when accessing USS files as a static response, but the CICS region user ID applies to all other attempts to access the USS file. If the CICS region user ID does not have permission to access the file, even an authorized Web client will be unable to view it. This is the case even when the USS file is defined as a CICS document template.

For each Web client user ID, choose and assign a suitable z/OS UNIX user identifier (UID). The UIDs are numbers that can be in the range 0 to 16,777,216. (0 is a superuser ID with system privileges, which should not be assigned to CICS users.)

Give permissions to Web clients to access the USS files and directories. You might choose to use the group permissions for the files and directories, or access control lists (ACLs). **If you are using ACLs, ensure that the FSSEC class is activated**. Use the RACF command SETROPTS CLASSACT(FSSEC) to do this. You can define ACLs prior to activating the FSSEC class, but you must activate the FSSEC class before ACLs can be used in access decisions.

Specify SEC=YES as a CICS system initialization parameter. (SECPRFX is not relevant for USS files, as they do not have RACF profiles).

Specify XHFS=YES as a CICS system initialization parameter. This step activates access control for all USS files in the CICS region.

When you have completed the setup procedure, from this point onwards:
- All Web clients who use a connection with basic authentication or client certificate authentication and attempt to access any USS files, must have a user profile in the security manager which contains a valid z/OS UNIX UID, and connects to a RACF group with a valid z/OS UNIX GID.
- To be able to view a web page derived from an USS file, Web clients who use a connection with basic authentication or client certificate authentication must have read permissions to the file and to the directories containing it, either individually or through the RACF groups to which they are connected.

If these conditions are not in place, Web clients will receive a 403 (Forbidden) status code, and CICS issues message DFHXS1116.

Overview of HTTP Basic Authentication

- Basic authentication flow
  - Client requests a page that requires authentication but does not provide credentials (username/password)
  - Server sends HTTP 401 response code and specifies an authentication realm
  - Client will display the authentication realm to the user and prompt for a username/password
  - Client resends request and includes an authentication header composed from the username and password
  - Server accepts the authentication and the requested page is returned

Notes

HTTP basic authentication is a simple challenge and response mechanism with which a server can request authentication information (a user ID and password) from a client. The client passes the authentication information to the server in an Authorization header. The authentication information is in base-64 encoding.

If a client makes a request for which the server expects authentication information, the server sends an HTTP response with a 401 status code, a reason phrase indicating an authentication error, and a WWW-Authenticate header. Most Web clients handle this response by requesting a username and password from the end user.

The format of a WWW-Authenticate header for HTTP basic authentication is:
WWW-Authenticate: Basic realm="*service-realm-name*"

The WWW-Authenticate header contains a realm attribute, which identifies the security system to which the authentication information requested (that is, the username and password) will apply. Web clients display this string to the end user when they request a username and password. Each realm may require different authentication information. Web clients may store the authentication information for each realm so that end users do not need to retype the information for every request.

When the Web client has obtained a username and password, it re-sends the original request with an Authorization header. Alternatively, the client may send the Authorization header when it makes its original request, and this might be accepted by the server, avoiding the challenge and response process.

The format of the Authorization header is:
Authorization: Basic *username:password*
The username:password string is encoded using the base-64 encoding scheme.

Specification of the Realm Name…

- In releases of CICS up to TS 3.1
  - The text contained in the realm keyword of the WWW-Authenticate header is fixed for each CICS region
    - WWW-Authenticate: Basic realm="CICS application ********"
      - Where ******** is the CICS region APPLID
  - This may force multiple requests for the userid/password when cloned listener regions are implemented
    - Each region is a different realm
- CICS TS 3.2 allows the text in the realm keyword to be provided by the installation
  - REALM keyword on the TCPIPSERVICE definition

Notes

TCPIPSERVICE definition

REALM(string) specifies the realm that is used for HTTP basic authentication. You can only specify this attribute for the HTTP protocol.

The realm is provided by CICS in the WWW-Authenticate header, and is seen by the end user during the process of basic authentication. It identifies the set of resources to which the authentication information requested (that is, the user ID and password) will apply. If you do not specify a realm, the default used by CICS is CICS application aaaaaaaa, where aaaaaaaa is the applid of the CICS region.

The realm can be up to 56 characters, and can include embedded blanks. It is specified in mixed case, and the case is preserved. Do not specify opening and closing double quotes, as CICS provides these when assembling the WWW-Authenticate header.

Acceptable characters: A-Z a-z 0-9 $ @ # . - _ % & ? ! : | ' = ¬ + * , ; < > ( ) Space characters are also permitted. If parentheses ("(" and ")") are used, you must use them as pairs of opening and closing parentheses.

Specification of the Realm Name…

- INQUIRE TCPIPSERVICE changes
  - REALM attribute added
- CEMT INQUIRE TCPIPSERVICE changes
  - REALM specification returned
- CICSPLEX SM
  - REALM attribute added to BAS panels
  - REALM specification returned on TCPIPS view

Notes

REALM(data-area) Returns the 56-character realm that is used during the process of HTTP basic authentication. This value is returned only when PROTOCOL has a value of HTTP. If no realm is specified for this service, the default realm used by CICS is returned, which is CICS application aaaaaaaa, where aaaaaaaa is the applid of the CICS region.

How to handle Web-client Basic Authentication

- CICS has handled HTTP Basic Authentication as a server for several releases:
  - Server sends HTTP 401 response with a WWW-Authenticate header
  - Client must respond with a Authorization header containing username:password encoded in a base-64 encoding
  - CICS decodes the base-64 encoding and validates the username and password
- What happens when CICS is the web client?
  - i.e. when using EXEC CICS WEB OPEN

How to handle Web-client Basic Authentication - Notes

- The HTTP Basic Authentication protocol is described in RFC2617 at http://www.ietf.org/rfc/rfc2617.txt
- The username and password have to be supplied in ASCII and then encoded in base-64.
- The web client then responds with the following header:

Authorization: Basic *xxxxxxxxxx*

- Where *xxxxxxxxxx* is the result after encoding the ASCII **username:password** in base-64

How to handle Web-client Basic Authentication

- In *product-level* CICS, up to TS 3.1, the application just receives the HTTP 401 response
- Application must handle this, and compose an Authorization header
- No easy tools to build base-64 response

How to handle Web-client Basic Authentication - Notes

- Product-level CICS provides no support for client-side Basic Authentication

SupportPac CA8J for Basic Authentication

- SupportPac CA8J provides assistance to:
  - Accept username and password from the application
  - Build `username:password` string in ASCII
  - Encode result in base-64
  - Give examples of how to use the result to create an `Authorization` header
- Supports CICS TS 3.1, TS 2.3, and SOAP for CICS feature
- **Available now** at the SupportPac website

SupportPac CA8J for Basic Authentication - Notes

- SupportPacs are available from http://www.ibm.com/cics/supportpacs/
- SupportPac CA8J contains only sample code. Product-level CICS (up to TS 3.1) is not modified to support client-side Basic Authentication

Basic authentication assistance for HTTP client applications

- In CICS TS 3.2, support for client-side Basic Authentication is included in the base product
- Parameters are added to Client EXEC CICS WEB SEND and CONVERSE commands
  - Allow the application to specify credentials
    - (username and password)
- AUTHENTICATE (<u>NONE</u>|BASICAUTH) option
  - If BASICAUTH is specified, credentials may be specified
    - USERNAME, USERNAMELEN, PASSWORD, PASSWORDLEN
  - If BASICAUTH is specified, but credentials are ***not*** supplied
    - Global user exit XWBAUTH is invoked

Notes

The WEB SEND and WEB CONVERSE commands have been expanded to allow you to provide basic authentication credentials (a username and password). CICS sends these details in an Authorization header to a server that is expecting it or in response to a HTTP 401 WWW-Authenticate message.

CICS converts the supplied username and password to the format that the HTTP basic authentication protocol is expecting. This allows you to supply your credentials in your usual EBCDIC character set through the WEB SEND or WEB CONVERSE command, or through the XWBAUTH user exit.

AUTHENTICATE (cvda)
This option allows you to specify user authentication details (credentials), to control access to restricted data. The CVDA values that apply for CICS as an HTTP client are:

NONE
specifies that there are no restrictions on accessing this data, therefore no credentials are required. This is the default value for AUTHENTICATE.

BASICAUTH
specifies that HTTP Basic Authentication credentials are required for this session. These details can be supplied within the command or by using the XWBAUTH global user exit.

Basic authentication assistance for HTTP client applications…

- Changes to WEB EXTRACT command
  - Discovers what REALM was sent in the HTTP 401 challenge
- REALM and REALMLEN parameters added

Notes

If a CICS HTTP client program sends a request to an HTTP server, that requires basic authentication, without supplying any credentials the server will send back a 401 challenge. The client program may find it necessary to determine the REALM that was sent back in the challenge message. CICS has added new options to the EXTRACT WEB command to allow the client to determine the REALM.

```
EXEC CICS WEB EXTRACT  SESSTOKEN(data-area)  REALM(data-area) REALMLEN(data-area)
```

REALM is the realm that was sent on the most recent HTTP 401 response, if any, from the server

REALMLEN is the length of the realm name in REALM.

On input, it is the length of the area provided in REALM to contain the realm name. On output, it is the length of the realm name returned in the HTTP 401 response. It may be zero if no HTTP 401 response has yet been received.

Error conditions

LENGERR

RESP2 values are:

141 REALMLEN is not positive, or is not large enough to contain the realm value returned in the HTTP 401 response.

Basic authentication assistance for HTTP client applications…

- Open a web session with the server
  - EXEC CICS WEB OPEN SESSTOKEN
- Send a message to the server
  - EXEC CICS WEB SEND SESSTOKEN
- Receive data from the server
  - EXEC CICS WEB RECEIVE STATUSCODE
    - Check for a 401 response
    - If your program needs to know the REALM returned on the 401 response
      - EXEC CICS WEB EXTRACT REALM REALMLEN
- If the status code is 401 (the server requires authentication)
  - Repeat the first Web send request with the authentication option
  - EXEC CICS WEB SEND SESSTOKEN AUTHENTICATE(BASICAUTH)
    - Supply the username and password on the request
      - EXEC CICS WEB SEND SESSTOKEN AUTHENTICATE(BASICAUTH) USERNAME () USERNAMELEN () PASSWORD () PASSWORDLEN ()
    - Or, allow user exit XWBAUTH to supply the credentials
      - EXEC CICS WEB SEND SESSTOKEN AUTHENTICATE(BASICAUTH)
- CICS will build and pass the credentials to the server in an Authentication header

Notes

When an HTTP 401 WWW-Authenticate message is received, your application must provide the username and password (credentials) required by the server for basic authentication. Your application can also provide these credentials without waiting for the 401 message. This explains the sequence of events for providing this information through your HTTP client application.

1) Open a web session with the server using the EXEC CICS WEB OPEN command, using the SESSTOKEN option. The SESSTOKEN will be returned to you when the session is opened successfully, and the session token must be used on all CICS WEB commands that relate to this connection.

2) Issue an EXEC CICS WEB SEND command, specifying the SESSTOKEN for this connection. This WEB SEND command retrieves the realm from the server.

3) Issue an EXEC CICS WEB RECEIVE command. The server returns a status code. Use the STATUSCODE option on the WEB RECEIVE command to check for a 401 response.

4) If the status code is 401 (the server requires authentication details), issue a repeat of your first EXEC CICS WEB SEND request, but this time add the AUTHENTICATE(BASICAUTH) option. The XWBAUTH global user exit is called by the client application. This second WEB SEND command uses the realm received from the first WEB SEND command and the XWBAUTH exit to determine the required username and password.

You might prefer to specify AUTHENTICATE(BASICAUTH) in your initial EXEC CICS WEB SEND command, instead of waiting for the 401 response. You can either:

4a) Supply your username and password in the WEB SEND command using the AUTHENTICATE(BASICAUTH) option.

4b) Invoke the XWBAUTH global user exit by specifying the AUTHENTICATE(BASICAUTH) option, but omitting your credentials. The user exit will be invoked, but the realm passed to the exit will be empty, as the realm has not yet been received from the server. The user exit must derive the required credentials from other parameters, for example, HOST and PATH.

If your application needs to know the realm that was sent in the 401 response, use the EXEC CICS WEB EXTRACT command.

5) CICS passes the username and password credentials to the server in an Authentication header.

XWBAUTH Global User Exit

- Used to supply credentials to a remote HTTP server
  - When invoked:
    - If the EXEC CICS WEB SEND or WEB CONVERSE command specifies AUTHENTICATE(BASICAUTH), but the USERNAME and PASSWORD are not specified
  - Inputs are the host and path components from the target URL
    - REALM name if returned on the 401 challenge
  - Output from the exit is a username and password
    - Specific to the platform on which the remote server is running
      - May be up to 256 characters in length
        - CICS will provide a predefined 64 byte area to store the username
        - Alternatively, you can place your username in your own area and replace the address in UEPUSNM with your username's address
        - If you create your own username area, the field can be up to 256 bytes in length
      - No validation of syntax performed by CICS

Notes

XWBAUTH enables you to specify basic authentication credentials (username and password) for a target server. XWBAUTH passes these to CICS on request, to create an Authorization header. XWBAUTH is called during processing of an EXEC CICS WEB SEND or EXEC CICS WEB CONVERSE command. The host name and path information are passed to the user exit, with an optional qualifying realm.

When AUTHENTICATE(BASICAUTH) is specified within the EXEC CICS WEB SEND or WEB CONVERSE command, a username and password can be provided directly by the application. If these are not supplied, XWBAUTH is invoked, providing an alternative way of specifying these credentials.

The username and password are usually specific to the remote server environment, and might be longer than the standard eight characters used by RACF systems. The username and password fields can be up to 256 characters in length. The syntax of these fields is not validated.

The host is passed to the user exit program as the UEPHOST parameter, and the path is passed as the UEPPATH parameter. The realm is passed optionally as the UEPREALM parameter. In response, the user exit program returns the username and password as the UEPUSNM and UEPPSWD parameters. A return code of UERCNORM indicates a successfully returned username and password. Return code UERCBYP indicates that the username and password cannot be identified, so the Authorization header will not be added to the request.

Directory Domain XPI Enhancements

- DFHDDAPX directory domain XPI functions for use with XWBAUTH
  - BIND_LDAP
  - END_BROWSE_RESULTS
  - FLUSH_LDAP_CACHE
  - FREE_SEARCH_RESULTS
  - GET_ATTRIBUTE_VALUE
  - GET_NEXT_ATTRIBUTE
  - GET_NEXT_ENTRY
  - SEARCH_LDAP
  - START_BROWSE_RESULTS
  - UNBIND_LDAP

Notes

BIND_LDAP: Establishes a session with an LDAP server

END_BROWSE_RESULTS: End browse session

FLUSH_LDAP_CACHE: Removes contents of all cached search responses

FREE_SEARCH_RESULTS: Release storage held by the search function

GET_ATTRIBUTE_VALUE: Retrieve value associated with an attribute returned by a search call

GET_NEXT_ATTRIBUTE: Get the next attribute in a series from an entry returned by a search call

GET_NEXT_ENTRY: Get the next entry from a series returned by a search call

SEARCH_LDAP: Send a search request, on a distinguished name, to the LDAP server

START_BROWSE_RESULTS: Start a browse on the results (entries or attributes) from a search call

UNBIND_LDAP: Terminates a session with an LDAP server

All the above XPI commands are threadsafe.

Typical use of the LDAP XPI functions by XWBAUTH

- BIND_LDAP
  - Establishes a session with an LDAP server
    - Used once on the first call to the global user exit XWBAUTH
    - The LDAP session token is stored in XWBAUTH's GWA
- UNBIND_LDAP
  - Releases the connection with the LDAP server
    - This function is only required during CICS shutdown processing
    - This function can be used during the XSTERM (system termination) global user exit
- SEARCH_LDAP
  - Searches for credentials based on an LDAP distinguished name obtained from the current user and the target URL and realm

Notes

If the XWBAUTH user exit uses the DFHDDAPX XPI, it should specify an LDAP distinguished name that identifies the URL and realm of the required user information.

For example, a distinguished name could be specified in the following format:

```
racfcid=uuuuuuuu, ibm-httprealm=rrrrrrrr,
labeledURI=xxxxxxxx, cn=BasicAuth
```

where:

`uuuuuuuu` is the current userid, obtained from the XWBAUTH parameter, UEPUSER

`rrrrrrrr` is the HTTP 401 realm, obtained from the XWBAUTH parameter, UEPREALM (if this exists)

`xxxxxxxx` is the target URL, obtained by concatenating http:// with the hostname from the XWBAUTH parameter, UEPHOST, and the path from the XWBAUTH parameter, UEPPATH

`cn=BasicAuth` is an arbitrary suffix that is configured into the LDAP server for storing Basic Authentication credentials

Authentication LDAP/TFIM sample programs

- DFH$WBPI
  - PLTPI program to set proxy server and LDAPBIND or Secure Token Server profile
- DFH$WBX1
  - XWBAUTH exit to retrieve userid & password from an LDAP server
- DFH$WBX2
  - XWBAUTH exit to retrieve userid & password from an STS
- DFH$WBEX
  - XWBOPEN exit to set proxy server
- DFH$WBLD
  - Sample LDIF file for populating the LDAP server with basic authentication credentials
  - Used by the `ldapmodify` command

Notes

These sample programs are for use at the Web domain exit, XWBOPEN and XWBAUTH. The XWBOPEN exit is invoked during processing of EXEC CICS WEB OPEN commands. XWBAUTH is called during processing of an EXEC CICS WEB SEND and EXEC CICS WEB CONVERSE commands.

DFH$WBPI: This sample PLT program is designed to be specified in the PLTPI and invoked during CICS initialization.

DFH$WBX1: This sample global user exit program has the following functions:

- Obtains an LDAP connection token
- Composes a distinguished name from XWBAUTH exit-specific parameters
- Searches the LDAP server
- Returns username and password to XWBAUTH

DFH$WBX2: This sample global user exit program has the following functions:

- Obtains an STS connection URL
- Send an ISSUE request to the Secure Token Server such as TFIM
- Returns username and password to XWBAUTH

DFH$WBEX: This sample global user exit program is designed to check the host name specified on the EXEC CICS WEB OPEN command, and make any host name starting with www use a proxy server if a proxy server name is specified in the global work area.

DFH$WBLD: Sample LDIF file for populating the LDAP server with basic authentication credentials
Used by ldapmodify command

Authentication LDAP/TFIM sample programs…

- DFH$WBPI
  - Invoked during PLTPI processing
  - Gets and analyzes INITPARM
  - Sets proxy server, if present
    - Stores proxy name in XWBOPEN's Global Work Area
    - Enables the XWBOPEN exit
  - Sets LDAP profile, if present
    - Stores LDAPBIND profile in XWBAUTH's Global Work Area
    - Enables the XWBAUTH exit
  - Sets the STS profile, if present
    - Stores STS URL in XWBAUTH's Global Work Area
    - Enables the XWBAUTH exit

Notes

DFH$WBPI: This sample PLT program is designed to be specified in the PLTPI and invoked during CICS initialization. DFH$WBPI provides the following functions:

- Initializes a global work area with a proxy server name supplied by the INITPARM system initialization parameter. This enables the DFH$WBEX program at exit XWBOPEN. The proxy is specified as:
  ```
  INITPARM=(DFH$WBPI='PROXY=proxyname')
  ```
  In CICS TS 3.1, the PROXY= keyword is not used: the proxyname is specified alone.
- Stores an LDAP bind profile name in the global work area. This profile is supplied by the INITPARM system initialization parameter and enables the DFH$WBX1 program at exit XWBAUTH.
- Stores an STS URL in the global work area. This profile is supplied by the INITPARM SIT parameter and enables the DFH$WBX2 program at exit XWBAUTH.
- From CICS TS 3.2 onwards: The SIT parameter is specified as:
  ```
  INITPARM=(DFH$WBPI,'LDAPBIND=profilename') or
  INITPARM=(DFH$WBPI,'STS=https://secure-token-service')
  ```

- The proxy server name and security profile name can be specified in any order in the INITPARM system initialization parameter, for example:
  ```
  INITPARM=(DFH$WBPI='PROXY=proxyname,LDAPBIND=profilename')
  ```

- Note that the total length of the INITPARM quoted text cannot exceed 60 characters.
  - The keyword PROXY= can be abbreviated to P=
  - The keyword LDAPBIND= can be abbreviated to L=
  - The keyword STS= can be abbreviated to S=

  LDAPBIND and STS are mutually exclusive. Neither is available in CICS TS 3.1.

Authentication sample programs…

- DFH$WBEX
  - Invoked at exit point XWBOPEN
  - Check host name on EXEC CICS WEB OPEN request
    - If host name starts with www
      - Use proxy server name from the GWA

Notes

DFH$WBEX: This sample global user exit program is designed to check the host name specified on the EXEC CICS WEB OPEN command, and make any host name starting with www use a proxy server if a proxy server name is specified in the global work area.

If all the requests from your CICS system should use a single proxy server, you can use the proxy server name from the INITPARM system initialization parameter, that DFH$WBPI used to initialize the global work area.

The proxy name must be specified as: INITPARM=(DFH$WBPI=’PROXY=proxyurl’) where proxyurl is the URL if a proxy server.

If you use a number of proxy servers or want to apply a security policy to different host names, you can load or build a table that matches host names to appropriate proxy servers or marks them as barred, which can then be used as a look-up table during processing of the EXEC CICS WEB OPEN command.

Authentication LDAP sample programs…

- DFH$WBX1
  - Invoked at exit point XWBAUTH
  - If LDAPBIND profile is in the GWA
    - Connect to LDAP Server
    - Construct a Distinguished name from:
      - UEPUSER, UEPREALM, UEPHOST, UEPPATH
    - Search the LDAP server
    - If successful, return username and password
    - If not try additional searches removing the UEPREALM parameter and then the UEPUSER parameter
    - Release storage
    - Terminate the LDAP connection at session termination

Notes

DFH$WBX1: This sample global user exit program has the following functions:

Obtains an LDAP connection token, by issuing the DFHDDAP BIND_LDAP XPI call, and stores it in the global work area.

Composes a distinguished name from XWBAUTH exit-specific parameters (UEPUSER, UEPREALM, UEPHOST and UEPPATH) to allow for LDAP searching.

Uses the LDAP connection token for search requests within a task.
Searches the LDAP server using the DFHDDAP SEARCH_LDAP XPI call.

Removes first the UEPREALM parameter, then the UEPUSER parameter and reissues the DFHDDAP SEARCH_LDAP XPI call if the search fails.

Browses successful search results by issuing the DFHDDAP START_BROWSE_RESULTS XPI call.

Returns username (from the UID attribute) and password (from the UserPassword attribute) to XWBAUTH by issuing two DFHDDAP GET_ATTRIBUTE_VALUE XPI calls.

Releases storage used by the browse function, by issuing the DFHDDAP END_BROWSE_RESULTS XPI call.

Terminates the LDAP connection (if the global user exit is called at session termination or if the LDAP bind profile was not stored in the global work area), by issuing the DFHDDAP UNBIND_LDAP XPI call.

Authentication LDAP sample programs…

- DFH$WBLD
  - LDAP Data Interchange Format file (LDIF)
  - Illustrates how to add Basic Authentication credentials to an LDAP server for use by the sample XWBAUTH exit DFH$WBX1
    - It must be copied to an HFS file before use
    - Use **`ldapadd`** or **`ldapmodify`** commands to update the LDAP directory
  - Refer to:
    - z/OS Integrated Security Services LDAP Server Administration and Use (SC24-5923)

Notes

DFH$WBLD: Sample data for populating the LDAP server with basic authentication credentials is provided in the CICS sample library, SDFHSAMP, as member DFH$WBLD. Entries can be added to the LDAP directory by using the ldapmodify command. The sample data file must be copied to an HFS file before use, as it uses the ldapmodify command, which does not support MVS datasets.

Example entry in DFH$WBLD:

```
This sample entry can be used when neither the CICS
signed-on user nor the realm of the remote host is known.

dn: labeledURI=http://www.example.com, cn=Basicauth
objectclass: labeledURIobject
objectclass: account
objectclass: simpleAuthObject
uid: guest
userPassword: UnauthenticatedUser
```

Authentication TFIM sample programs…

- DFH$WBX2
  - Invoked at exit point XWBAUTH
  - If an STS URL is in the GWA
    - Construct an XML request for the STS (such as TFIM)
    - Invoke a web service pipeline with an ISSUE request for the STS using the DFHWSTC-V1 channel
      - See session 4133 for details of the “Linkable Trust” interface
    - If successful, populate the UEPUSNM and UEPPSWD fields from the STS response
    - If unsuccessful, set UERCERR

Notes

DFH$WBX2: This sample global user exit program has the following functions:

Build the container structure for the web service invocation. This includes containers for the STS URL, STS action, requested token type and the user being validated.

Invoke a CICS Web Service request with a link to DFHPIRT using channel DFHWSTC-V1 (the “Linkable Trust” interface). This sends an “issue” request to the specified Secure Token Service. This is a Web Service specifically designed to convert credentials. As an example, you can use the Tivoli Federated Identity Manager as a secure Token Service. For more information about the Linkable Trust interface, see session 4133 *Web Services, Security and Transactions*.

Receive and analyze the response. If successful, populate UEPUSNM and UEPPSWD. If unsuccessful, set UERCERR.

Web Services Security

- CICS now provides support for the Web Services Security standard (WSSE) as part of CICS TS 3.1
  - Support was not present at General Availability for CICS TS 3.1
  - Delivered by Service stream as APAR PK22736 (PTF UK15261)
  - Incorporated into CICS TS 3.2 base
- CICS also supports the WS-Trust specification in CICS TS 3.2
- For more information
  - See full description in session 4133 *Web Services, Security and Transactions*

Web Services Security - Notes

Web Services Security is one of the Web Services specifications. The formal specification is entitled **Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)** and is found at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

Web Services Security overview

- WSSE is implemented by adding special-purpose sub-elements to the `Security` header in a SOAP envelope
  - **`UsernameToken`**
    - Containing a userid & optional password
  - **`BinarySecurityToken`**
    - Containing an X509 certificate or Kerberos ticket
  - **`Signature`**
    - Containing a digitally signed list of cryptographic digests of portions of the message
- …and the following element anywhere in the SOAP envelope
  - **`EncryptedData`**
    - Replaces any document fragment with an encrypted copy of it

Web Services Security - Notes

Most of the Web Services Security specification describes the subelements of the **Security** SOAP header:

UsernameToken contains a userid & optional password.
Contains subelements: **Username Password**

BinarySecurityToken contains an X509 certificate or Kerberos ticket.
Contains subelements: **EncodingType ValueType**

Signature contains a digitally signed list of cryptographic digests of portions of the message. Contains subelements: **SignedInfo CanonicalizationMethod SignatureMethod Transforms DigestMethod DigestValue SignatureValue KeyValue Object**

EncryptedData replaces any document fragment with an encrypted copy of itself.
Contains subelements: **EncryptionMethod CipherData CipherValue EncryptionProperties KeyInfo**

Web Services Security implementation

- WSSE is implemented in a Security “handler” within a CICS PIPELINE
- The Security handler must be the outermost handler in the pipeline
  - It has to decrypt inbound data that is contained within any **`EncryptedData`** document elements
    - It also validates any digital signatures
  - Other handlers cannot parse the XML until it is decrypted
  - It has to encrypt and digitally sign outbound data that is composed by “inner” handlers

Web Services Security implementation - Notes

As with most configuration for CICS Web Services support, the configuration details are provided in an XML configuration file that is associated with the RDO PIPELINE definition.

The XML configuration file contains specifications for a number of “handlers”. The SOAP message is processed by each handler in turn, and the result is passed on to the next handler, until a “terminal” hander is reached, which processes the message and produces a response. The response output by the terminal handler is then returned through the other handlers in reverse.

The Security handler has to be the first handler to process an incoming message, and the last handler to process an outgoing message. The Security handler has to decrypt the incoming message before any of the other handlers are able to process the message in clear text, and the outgoing clear text has to be complete before it can be encrypted or signed.

Summary

- Support for TLS 1.0
- Cipher suite selection
  - New AES 128 and 256 encryption
  - Specification of minimum and maximum encryption level
- Performance enhancements
  - SSL session id caching
  - OTE exploitation
- Revocation detection
  - For both userids and certificates
  - New Certificate Revocation List utility transaction
- Mixed case password support
- Client Basic Authentication *via* SupportPac or in CICS TS 3.2
- Web Services Security
  - including WS-Trust in CICS TS 3.2
- Resource Security for document templates
- Transaction user access control for UNIX files

Summary - Notes

There are a range of benefits that come from the improvements to security.

CICS now supports the Transport Layer Security (TLS) 1.0 protocol as well as SSL 3.0, allowing you to use the new AES cipher suites that offer 128-bit and 256-bit encryption.

There is more flexibility in controlling the encryption negotiation between client and server. You can specify a minimum as well as a maximum encryption level in CICS for negotiating with particular users.

CICS can now check all certificates against a certificate revocation list (CRL) when negotiating with clients. Any connections using revoked certificates are closed immediately.

You can specify whether you want to share session IDs across a sysplex by using the SSL cache. CICS will perform a partial SSL handshake if the client has negotiated with CICS previously. If the cache is shared across a number of CICS regions, this will improve the performance of SSL negotiation and connection

There are improvements to the performance of SSL to support new functions such as Web Services. The number of simultaneous SSL connections that can be used in the system at one time has increased to achieve better throughput.

At the time of writing, SupportPac CA8J is not yet available. Watch the SupportPac website for availability.

IBM 2007 IMPACT

IBM

Questions
and
Answers

impact▪venture*