Statements für den Internet- und XML-Zugriff

Dieses Kapitel gibt einen Überblick über die Natural-Statements für den Zugriff auf Internet und XML, behandelt grundsätzliche Voraussetzungen für die Benutzung dieser Statements in einer Großrechner-Umgebung, informiert über generell geltende Einschränkungen und enthält ein Verzeichnis sonstiger Informationsquellen.

Um diese Statements vollständig nutzen zu können, ist eine gründliche Kenntnis der zugrunde liegenden Kommunikationsstandards erforderlich.

Dieses Kapitel behandelt folgende Themen:

- Verfügbare Statements
- Generelle Voraussetzungen
- HTTPS-Unterstützung für das REQUEST DOCUMENT-Statement unter z/OS
- Einschränkung bezüglich IMS/TM
- Voraussetzung für die Unterstützung von Statements für den Internet- und XML-Zugriff unter openUTM
- Programmbeispiel
- Häufig gestellte Fragen
- Weitere Informationsquellen

Verfügbare Statements

Die folgenden Natural-Statements stehen zum Zugriff auf das Internet und auf XML-Dokumente zur Verfügung:

- REQUEST DOCUMENT
- PARSE XML

REQUEST DOCUMENT

Leistungsumfang

Dieses Statement ermöglicht die Verwendung des Hypertext-Übertragungsprotokolls (Hypertext Transfer Protocol, HTTP) sowie - nur unter z/OS - des sicheren Hypertext-Übertragungsprotokolls (Hypertext Transfer Protocol Secure, HTTPS), um mit einem gegebenen Uniform Resource Identifier (URI) oder Uniform Resource Locator (URL), oder anders ausgedrückt, mittels einer Internet- oder Intranet-Adresse einer Web-Seite, auf Dokumente im Web zuzugreifen.

REQUEST DOCUMENT implementiert einen HTTP-Client auf Natural-Statement-Ebene. Damit können Anwendungen auf jeden beliebigen HTTP-Server entweder im Intranet oder im Internet zugreifen. Das Statement bietet verschiedene Operanden, mit denen HTTP-Anfragen entsprechend den Erfordernissen der Benutzeranwendung formuliert werden können. Zum Beispiel kann man mit den nach Außen weisenden Operanden benutzerdefinierte HTTP-Nachrichtenköpfe (Header), Form-Daten oder ganze Dokumente an einen HTTP-Server senden. Die nach Innen weisenden Operanden können verwendet werden, um ein Dokument vom Server abzurufen, den gesamten vom Server zurückgesandten HTTP Nachrichtenkopfblock (Header Block) zu sichten oder um die Werte spezieller Nachrichtenköpfe zurückzusenden usw. Über Binär-Format-Operanden können auch binäre Objekte wie zum Beispiel GIF-Dateien mit dem HTTP-Server ausgetauscht werden. Außerdem können Operanden zur Basis-Authentifizierung mit Benutzerkennung und Paßwort angeben werden, deren Inhalt gemäß den HTTP-Standards mittels Base64-Kodierung verschlüsselt übertragen wird.

Natural unterstützt die folgenden Anforderungsmethoden:

- GET - Dokumente und HTTP-Header abrufen,
- HEAD - Nur HTTP-Header abrufen,
- POST - Form-Daten zu einem HTTP-Server übertragen,
- PUT - Eine Datei auf einen HTTP-Server hochladen.

Die Auswertung der Anforderungsmethode (Request Method) erfolgt normalerweise automatisch anhand der beim REQUEST DOCUMENT-Statement kodierten Operanden. Die dadurch vorgegebene Anforderungsmethode kann jedoch durch eine explizite Benutzerangabe in einem Request Method Header überschrieben werden.

Bei der Übertragung von Daten mit Hilfe des REQUEST DOCUMENT-Statements erfolgt normalerweise keine Codepage-Umsetzung. Wenn Sie möchten, dass die ausgehenden und/oder eingehenden Daten in einer spezifischen Codepage kodiert werden, können Sie die DATA ALL-Klausel und/oder die RETURN PAGE-Klausel des REQUEST DOCUMENT-Statements benutzen, um dies anzugeben.

Um den Datenaustausch von einem EBCDIC-basierten Großrechner mit HTTP-Servern, die meistens mit UTF-8- or ISO-8859-1-kodierten Daten arbeiten, zu erleichtern, bietet das Statements ENCODED-Klauseln, die eine implizite oder automatische Konvertierung von eingehenden und ausgehenden Dokumentendaten gestatten.

Technische Umsetzung

Die Implementierung des REQUEST DOCUMENT-Statement erfolgt im Wesentlichen in zwei Schichten:

- eine unabhängige Laufzeitschicht, in der die gesamte HTTP-Verarbeitung, URL-Analyse, Datenkonvertierung usw. stattfindet, und
- eine Schicht, in der eine umgebungsabhängige Routine die TCP/IP-Kommunikation zwischen Natural und dem HTTP-Server verarbeitet. Die Implementierung dieser Schicht erfolgt auf der Basis von LE (Language Environment) Sockets für z/OS, z/VSE und VM/CMS, SMARTS Sockets für Com-plete und den Natural Development Server und CRTE Sockets für BS2000/OSD. Für CICS ist die entsprechende Socket Library im Build-Prozess enthalten.

Natural für Großrechner unterstützt nur die HTTP-Protokoll-Version 1.0. Das heißt, dass keine persistente Verbindung zum Server aufrechterhalten wird. Da in fast allen Firmennetzen der Zugang zum Internet vom Client aus über einen Proxy-Server abgewickelt wird, ist es möglich, Natural mit entsprechenden Einstellungen für den Proxy-Server und für den Port zu konfigurieren, auf dem der Proxy-Server läuft. Darüber hinaus besteht die Möglichkeit, Namenssuffixe lokaler Domains (Intranet Sites) anzugeben, auf die anstelle des Proxy-Servers direkt zugegriffen werden soll. Siehe auch *Übersicht über Relevante Natural-Parameter* weiter unten.

Der Proxy-Server, der zwischen dem Client (Benutzer) und dem Internet angeordnet ist, hat folgende Aufgaben: Er empfängt die Anfrage vom Client, leitet sie an den Ziel-Server weiter, speichert das gelieferte Dokument zwischen und leitet es dann an den Client weiter. Die Nutzung eines Proxy-Servers ist von Vorteil, weil er dank Zwischenspeicherung eine verbesserte Performance bewirkt und weil er Sicherheitsprobleme vermeiden hilft (die meisten Proxy-Server fungieren auch als Firewall).

Das folgende Beispiel zeigt, wie dieses Statement benutzt werden kann, um auf ein extern vorliegendes Dokument zuzugreifen:

```
REQUEST DOCUMENT FROM
"http://bolsap1:5555/invoke/sap.demo/handle_RFC_XML_POST"
WITH
USER #User PASSWORD #Password
DATA
NAME 'XMLData' VALUE #Queryxml
NAME 'repServerName' VALUE 'NT2'
RETURN
PAGE #Resultxml
RESPONSE #rc
```

Syntax

Die Syntax des `REQUEST DOCUMENT`-Statements und konkrete Anwendungshinweise finden Sie in der *Statements*-Dokumentation.

Plattform-Unterstützung für REQUEST DOCUMENT

Das `REQUEST DOCUMENT`-Statement wird auf folgenden Großrechner-Plattformen unterstützt:

- **z/OS:** Batch, TSO, CICS, Com-plete und IMS/TM
- **z/VSE:** Batch, Com-plete und CICS
- **VM/CMS**
- **BS2000/OSD:** Batch, TIAM und *open*UTM *

* Siehe auch *Voraussetzung für die Unterstützung von Statements für den Internet- und XML-Zugriff unter openUTM* weiter unten.

Darüber hinaus steht dieses Statement auch auf allen von Natural unterstützten Open-Systems-Plattformen zur Verfügung.

PARSE XML

Leistungsumfang

Das PARSE XML-Statement ermöglicht es, XML-Dokumente von einem Natural- Programm aus zu parsen, das heißt, zu zerlegen und in ein für die Weiterverarbeitung brauchbares Format umzuwandeln.

Mit dem PARSE XML-Statement wird ein vollständiger XML-Parser in Natural integriert. Damit können Natural-Anwendungen XML-Dokumente parsen und ihren Inhalt auf einfache Weise verarbeiten. Das Statement öffnet eine Verarbeitungsschleife und liefert, wenn eines der Ereignisse aus einer Ereignisliste während des Parse-Prozesses auftritt, den entsprechenden Pfad durch das Dokument, Name und Wert von geparsten Elementen sowie einige Parser-Statusvariablen.

Technische Umsetzung

Für das Parsen von XML-Dokumenten sind die folgenden Parse-Strategien am geläufigsten:

- DOM (Document Object Model), ein objektorientierter Ansatz
- SAX (Simple Access to XML), ein datenstromorientiertes Parse-Modell

Die Implementierung des PARSE XML-Statements in Natural für Großrechner basiert auf der SAX-Methode. Verwendet wird ein Großrechner-Port der Version 2.0.0 des (Open Source) SAX Parsers EXPAT.

Das Parsen erfolgt intern mit einem UTF-16-kodierten Image des zu parsenden Dokuments. Wird das Dokument nicht in dieser Kodierung angeliefert, erfolgt eine interne Konvertierung nach UTF-16, bevor der Parse-Vorgang beginnt. Dies muss bei der Installation von Natural berücksichtigt werden, wenn die Thread-Größe für die TP-Umgebung bestimmt wird.

Die Kodierung des herein kommenden Dokuments wird automatisch geprüft:

1. Zunächst wird auf das Vorhandensein einer BOM (Byte Order Mark), die die Kodierung des Dokuments kennzeichnet, geprüft.
2. Wird keine BOM gefunden, erfolgt eine Prüfung auf ASCII, EBCDIC oder UTF-16 (BE oder LE: Big Endian oder Little Endian).
3. Wird eine EBCDIC- oder ASCII-Kodierung festgestellt, dann wird nach einer Kodierungsverarbeitungsanweisung (PI, Processing Instruction) gesucht.

   Kann überhaupt keine Kodierung festgestellt werden, dann wird eine entsprechende Fehlermeldung ausgegeben und der Parse-Vorgang beendet. Intern arbeitet der Parser mit UTF-16BE, deshalb wird das zu parsende Dokument immer in diese Kodierung konvertiert, bevor es an den EXPAT-Parser weitergeleitet wird.
4. Falls eine Kodier-PI gefunden wird, gelten die folgenden Standardvorgaben:
   - Für ASCII wird UTF-8 als Kodierung angenommen.
   - Für EBCDIC wird die Default-Codepage (siehe Systemvariable ,*CODEPAGE) als Kodierung angenommen.

Der Parse-Vorgang selbst läuft in zwei Phasen ab:

- In der ersten Phase wird der Parser wiederholt gerufen, um einen genau festgelegten Satz an Callback-Einträgen anzukündigen. Diese Einträge werden vom Parser jedes Mal vorgenommen, wenn in dem zurzeit geparsten Dokument ein entsprechendes Element vorgefunden wird. Ein solches Ereignis, das einen Callback auf den entsprechenden Eintrag auslöst, ist zum Beispiel das Auftreten einer Start-Markierung (Tag). Die Callback-Einträge bilden die Natural-Laufzeitlogik für die Ausführung des Parse-Vorgangs.

- In der zweiten Phase erfolgt der eigentliche Parse-Vorgang. Der Parser wird mit dem zu parsenden Dokument als Eingabe-Operand aufgerufen. Jetzt wird jedes Element geparst, und für jeden Elementtyp wird die entsprechende Callback-Routine aufgerufen. Dann verarbeitet die Natural-Laufzeitumgebung das zurückgegebene Element, aktualisiert die `RETURN`-Operanden und beginnt mit der Ausführung der Parse-Schleife zum Verarbeiten dieser Operanden. Danach wird der Parser erneut gestartet, um den Parse-Vorgang fortzusetzen. Der Parse-Vorgang wird beendet, wenn das Dokument vollständig geparst ist oder wenn im aktuellen Dokument ein XML-Syntaxfehler auftritt, was bedeutet, dass das Dokument formell nicht in Ordnung ist.

Anmerkung:
Aus technischen Gründen unterstützt Natural für Großrechner keine verschachtelten Parse-Schleifen.

Verarbeitung von XML-Leerzeichen und vordefinierten Entities

Ab Natural-Version 4.2.5 tritt beim Parsen von Zeichendaten keine Unterbrechung bzw. keine Schleife mehr auf, wenn die geparste Zeichenkette Leerzeichen (Whitespace Characters) oder vordefinierte XML Entities enthält. Dieses mit Natural-Versionen vor Version 4.2.5 auftretende Problem wurde gelöst. Bei Natural-Version 4.2.5 ist das Parsen von Zeichendaten kompatibel mit Natural für Windows, UNIX und Linux.

Die Ausgaben des folgenden Beispielprogramms verdeutlichen den Unterschied zwischen der Version 4.2.5 und den Vorgänderversionen.

```
DEFINE DATA
LOCAL
1 PAGE    (A)   DYNAMIC
1 #PATH   (A200)
1 #NAME   (A) DYNAMIC
1 #VALUE  (A40)
1 #CMX    (A) DYNAMIC
1 #CMP    (A) DYNAMIC
END-DEFINE
FORMAT PS=60 LS=80
COMPRESS ' A&lt;B ' H'0D0D' ' B&lt;C' INTO #CMX LEAVING NO
MOVE ALL  #CMX TO #CMP UNTIL 16
COMPRESS
'<?xml version="1.0" ?>'
'<character-data-sample>'
'<string_with_whitespace_and_predefined_entity>' #CMX
'</string_with_whitespace_and_predefined_entity>'
'</character-data-sample>'
 INTO PAGE LEAVING NO
PARSE XML PAGE INTO PATH #PATH NAME #NAME VALUE #VALUE
PRINT #PATH /   'NA='  #NAME / 'VA=' #VALUE
LOOP
END
```

Ausgabe bei Ausführung des Programms mit Natural-Versionen kleiner als Version 4.2.5:

```
Page      1                                                  08-11-04  14:39:51

character-data-sample
NA= character-data-sample
VA=
character-data-sample/string_with_whitespace_and_predefined_entity
NA= string_with_whitespace_and_predefined_entity
VA=
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
VA=  A
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
VA= <
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
VA= B
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
VA= ?
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
VA= ?
NA=
VA= B
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
VA= ?
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
VA= ?
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
VA=  B
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
VA= <
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
VA= C
character-data-sample/string_with_whitespace_and_predefined_entity//
NA= string_with_whitespace_and_predefined_entity
VA=
character-data-sample//
NA= character-data-sample
VA=
MORE
```

Ausgabe bei Ausführung des Programms mit Natural-Version 4.2.5 (oder höher):

```
Page      1                                                  08-11-04  13:41:34

character-data-sample
NA= character-data-sample
VA=
character-data-sample/string_with_whitespace_and_predefined_entity
NA= string_with_whitespace_and_predefined_entity
VA=
character-data-sample/string_with_whitespace_and_predefined_entity/$
NA=
```

```
VA=  A<B?? B<C
character-data-sample/string_with_whitespace_and_predefined_entity//
NA= string_with_whitespace_and_predefined_entity
VA=
character-data-sample//
NA= character-data-sample
VA=


MORE
```

Syntax

Die Syntax des PARSE XML-Statements und konkrete Anwendungshinweise finden Sie in der *Statements*-Dokumentation.

Plattform-Unterstüzung für PARSE XML

Das PARSE XML-Statement wird auf folgenden Großrechner-Plattformen unterstützt:

- **z/OS:** Batch, TSO, CICS, Com-plete, IMS/TM *
- **z/VSE:** Batch, Com-plete and CICS
- **VM/CMS**
- **BS2000/OSD:** Batch, TIAM und *open*UTM **

* Siehe *Einschränkung bezüglich IMS/TM* weiter unten.

* * Siehe auch *Voraussetzung für die Unterstützung von Statements für den Internet- und XML-Zugriff unter openUTM* weiter unten.

Darüber hinaus steht Ihnen dieses Statement auch auf allen von Natural unterstützten Open-Systems-Plattformen zur Verfügung.

Generelle Voraussetzungen

Dieser Abschnitt beschreibt die generellen Voraussetzungen, die erfüllt sein müssen, um die Natural-Statements REQUEST DOCUMENT und PARSE XML benutzen können.

- Installationserfordernisse erfüllen
- Grundsätzliche Profileinstellungen vornehmen
- REQUEST DOCUMENT und PARSE XML aktivieren/deaktivieren
- Unicode-Unterstützung einschalten

Installationserfordernisse erfüllen

Damit die Natural-Statements REQUEST DOCUMENT und PARSE XML überhaupt benutzt werden können, müssen zunächst die in der *Installation*-Dokumentation beschrieben Installationsschritte ausgeführt werden, siehe *Installation Steps for REQUEST DOCUMENT and PARSE XML*.

Da die Statements REQUEST DOCUMENT und PARSE XML zumindest intern immer Daten von einer Kodierung in eine andere zu konvertieren haben, muss Natural mit ICU-Support betrieben werden. Deshalb muss die ICU Library installiert sein.

Damit REQUEST DOCUMENT oder PARSE XML ausgeführt werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Ein TCP/IP-Stack muss verfügbar und für die Ausführungsumgebung freigegeben sein.
- Ein Domain Name System Server oder DNS-Dienste müssen in der Ausführungsumgebung vorhanden sein, um die Internet-Adressauflösungsanforderungen (Funktion gehthostbyname) aufzulösen.
- Ein Natural-Treiber muss installiert und für LE (Language Environment, in IBM-Umgebungen) bzw. CRTE (in BS2000/OSD-Umgebungen) freigegeben sein,
- Support von HTTPS unter Com-plete erfordert APS Version 2.7.2 Patch Level 16.

Grundsätzliche Profileinstellungen vornehmen

Nachfolgend erhalten Sie eine Übersicht über die Natural-Profil- und/oder Sessionparameter, die die Unterstützung der Statements REQUEST DOCUMENT and/or PARSE XML ein- oder ausschalten bzw. anderweitig beeinfussen:

Übersicht über die relevanten Natural-Parameter

Parameter	Zweck
XML	Dieser Natural-Profilparameter bzw. das entsprechende Parameter-Makro NTXML mit zugehörigen Schlüsselwort-Subparametern dient zum gemeinsamen oder separaten Aktivieren/Deaktivieren der Unterstützung der Statements REQUEST DOCUMENT and PARSE XML. Mit den Schlüsselwort-Subparametern können außerdem verschiedene Optionen eingestellt werden, zum Beispiel getrenntes Aktivieren/Deaktivieren der beiden Statements, Name der Default-Codepage, URL des (Intranet-) Proxy-Servers, Port-Nummer des Proxy, URL und Port-Nummer des (Intranet-) SSL-Proxy-Servers, Namen der lokalen Domains, die direkt angesprochen werden sollen, usw. Als Veraussetzung für die Benutzung des Profilparameters XML bzw. des Parameter-Makros NTXML muss der Profilparameter CFICUauf ONgesetzt sein.
CFICU	Dieser Natural-Profilparameter bzw. das entsprechende Parameter-Makro NTCFICU mit zugehörigen Schlüsselwort-Subparametern dient zum Einschalten der Unicode- und Codepage-Unterstützung.
CP	Dieser Natural-Profilparameter dient zum Festlegen der Default-Codepage für Natural-Daten und Natural-Sourcecode.
CPCVERR	Dieser Natural-Profil- und Sessionparameter legt fest, ob ein während des Konvertierens auftretender Konvertierungsfehler zu einer Natural-Fehlermeldung führt oder nicht.

REQUEST DOCUMENT und PARSE XML aktivieren/deaktivieren

Um die Unterstützung der Statements REQUEST DOCUMENT und PARSE XML einzuschalten

1. Um beide Statements *gemeinsam* einzuschalten, setzen Sie den Natural-Profilparameter XML (oder das entsprechende Parameter-Makro NTXML) und außerdem die Schlüsselwort-Subparameter RDOC und PARSE auf ON.

   Oder:
   Um die Unterstützung der Statements REQUEST DOCUMENT und PARSE XML *einzeln* einzuschalten, setzen Sie nur den betreffenden Schlüsselwort-Subparameter auf ON:

   RDOC zur Unterstützung des REQUEST DOCUMENT-Statements

   PARSE zur Unterstützung des PARSE XML-Statements

2. Falls die Installationsplattform hinter einer Internet Firewall betrieben wird oder falls der Internet-Datenverkehr über einen Proxy-Server geroutet wird, müssen Sie bei den XML/NTXML Keyword-Subparametern für Proxy und Proxyport entsprechende Angaben machen.

Um die Unterstützung der Statements REQUEST DOCUMENT und PARSE XML auszuschalten

1. Um beide Statements *gemeinsam* auszuschalten, setzen Sie den Natural-Profilparameter XML oder das Parameter-Makro NTXML auf OFF.

   Oder:
   Um die Unterstützung der Statements REQUEST DOCUMENT und PARSE XML *einzeln* auszuschalten, setzen Sie nur den betreffenden Schlüsselwort-Subparameter auf OFF:

   RDOC zum Abschalten der Unterstützung des REQUEST DOCUMENT-Statement

   PARSE zum Abschalten der Unterstützung des PARSE XML-Statements

Ausführliche Informationen hierzu finden Sie unter *XML - Activate PARSE XML and REQUEST DOCUMENT Statements* in der *Parameter Reference*-Dokumentation

Unicode-Unterstützung einschalten

Um die Unicode-Unterstützung einzuschalten, muss der Profilparameter CFICU auf ON gesetzt werden.

Informationen zu den verschiedenen Optionen, die mit den Schlüsselwort-Subparametern des Profilparameters CFICU gesetzt werden können, finden Sie unter *CFICU - Unicode Support* in der *Parameter Reference*-Dokumentation.

Siehe auch die Abschnitte bezüglich der Statements PARSE XML und REQUEST DOCUMENT im Abschnitt *Statements*, der Teil des Abschnitts *Unicode and Code Page Support in the Natural Programming Language* in der *Unicode and Code Page Support*-Dokumentation ist.

HTTPS-Unterstützung für das REQUEST DOCUMENT-Statement unter z/OS

- Kurze Einführung in HTTPS
- HTTPS über AT-TLS
- Verwaltung von Zertifikaten unter z/OS
- Verwendung von RACF Key Rings
- Verwendung von Key-Datenbanken

Kurze Einführung in HTTPS

HTTPS (Hypertext Transfer Protocol Secure), das sichere HTTP-Übertragungsprotokoll, ist eine zusätzliche Sicherheitsschicht zwischen dem HTTP- und dem TCP/IP-Protokoll-Stack:

Schicht	Protokoll
Anwendungsschicht	HTTP(S)
Sicherheitsschicht	TLS/SSL
Transportschicht	TCP
Netzwerkschicht	IP

HTTPS wurde eingeführt, um eine Verschlüsselung und eine Authentifizierung von Kommunikationspartnern für eine sichere Datenkommunikation über das Internet zu ermöglichen.

Das HTTPS-URI-Schema wird verwendet um anzeigen, dass die HTTP-Kommunikation gesichert ist. Für die Verschlüsselung der Daten wird das SSL-Protokoll (SSL = Secure Socket Layer) oder sein Nachfolger das TSL-Protokoll (TSL = Transport Layer Security) verwendet. Die Authentifizierung erfolgt hierbei durch den Austausch von Zertifikaten (Certificates), die die Identität der Kommunikationspartner garantieren.

In den meisten Fällen von HTTPS-Kommunikation identifiziert sich jedoch nur der Server gegenüber dem Client. Eine Identifizierung des Clients mittels eines Client-Zertifikats erfolgt relativ selten.

Eine SSL-Kommunikation wird in mehreren Schritten aufgebaut:

- Sie beginnt mit der Identifizierung und Authentifizierung der Kommunikationspartner über das so genannte SSL-Handshake-Protokoll (Client Hello, Server Hello).
- Auf diesen "Handshake" folgt der Austausch eines symetrischen Sitzungsschlüssels über eine asymetrische Verschlüsselung (Private – Public Key Proceeding). Der Public Key, der dabei vom Client verwendet wird, ist ein wesentlicher Bestandteil des Server-Zertifikats.
- Nach erfolgtem "Handshake" und Austausch der Schlüssel, werden die verschlüsselten Payload Request Messages übermittelt. Der in den vorangegangenen Schritten ausgehandelte symmetrische Sitzungsschlüssel wird zur Ver- bzw Entschlüsselung dieser Mitteilungen verwendet.

Beim HTTPS-Protokoll werden andere Port-Nummern als beim Standard-HTTP-Prokoll verwendet. Während HTTP normalerweise Port 80 verwendet, benutzt HTTPS als Default die Port-Nummer 443.

Der HTTP-Zugang zum Internet von einem Client, der an ein LAN (Local Area Network) angeschlossen ist, wird normalerweise über spezielle HTTP-Server, so genannte Proxy-Server, abgewickelt. Proxy-Server sind Gateways vom LAN. Sie dienen zur Durchführung von Sicherheitsmaßnahmen, stellen Platz zur Zwischenspeicherung (Cache) zu Verfügung, führen Validierungsroutinen oder Filterfunktionen aus und wirken als Firewall. zum Internet. Durch HTTPS gesicherter Internet-Zugang erfolgt meistens über einen eigenen Proxy-Server, der die Verbindungen zu den fernen Servern aufrecht erhält. Dieser Proxy ist als "SSL Proxy" bekannt.

Zertifikate sind binäre Dokumente, die unter anderem auch Informationen über den Besitzer und die Ausgabestelle des Zertifikats, den Public Key für die Verschlüsselung der Schlüsseldaten der Sitzung, das Verfallsdatum und eine digitale Signatur enthalten. Die von HTTPS-Servern vorgelegten Zertifikate sind normalerweise das letzte Glied in einer kompletten Kette von Zertifikaten. Eine solche Kette von Zertifikaten bezeichnet man als Public Key Infrastructure (PKI). Das Zertifikat am oberen Ende der Kette bezeichnet man als Wurzel-Zertifikat (Root Certificate). Diese Zertifikate werden grundsätzlich von speziellen Organisationen ausgegeben, die man als Certificate Authorities (CA) bezeichnet. Root-Zertifikate, die von einer CA ausgegeben und unterzeichnet werden, nennt man auch CA-(Root-)Zertifikate. Weitere Informationen siehe *HTTP Developers Manual* und andere Informationsquellen im Internet.

HTTPS über AT-TLS

Die HTTPS-Unterstützung für das Natural-Statement `REQUEST DOCUMENT` basiert auf der z/OS Communication Server Component AT-TLS (Application Transparent-Transport Layer Security).

AT-TLS bietet eine TLS/SSL-Verschlüsselung als konfigurierbaren Dienst für Socket-Anwendungen. Realisiert ist es als zusätzliche Schicht über dem TCP/IP Protocol Stack, die die SSL-Funktionalität in nahezu transparentem oder sogar voll transparentem Modus für Socket-Anwendungen nutzt. Siehe *z/OS Communications Server, IP Programmer's Guide and Reference. Version 1, Release 9*, Chapter 15, IBM manual SC31-8787-09).

Diese Modi sind:

- **Basic**

  Die Socket-Anwendung läuft unverändert im transparenten Modus, ohne zu "wissen", dass eine verschlüsselte Kommunkation über AT-TLS durchgeführt wird. Auf diese Weise können Altanwendungen ohne Änderungen am Sourcecode im gesicherten Modus laufen.

- **Aware**

  Die Anwendung "weiß", dass sie im gesicherten Modus läuft, und kann TLS-Statusinformationen abfragen.

- **Controlling**

  Die Socket-Anwendung "weiß" von der Verwendung von AT-TLS und kann zwischen gesichertem und ungesichertem Modus hin- und herschalten.

Natural für Großrechner verwendet den *Controlling*-Modus, um den gesicherten Modus für HTTPS-Anfragen einzuschalten, wohingegen HTTP-Anfragen unverschlüsselt bleiben.

Verwaltung von Zertifikaten unter z/OS

Zertifikate, die mit AT-TLS verwendet werden sollen, können unter z/OS auf zwei Arten verwaltet werden. Sie werden in RACF Key Rings oder in Key-Datenbanken gespeichert, die auf dem z/OS UNIX Services-Dateisystem liegen. Welche Vorgehensweise tatsächlich gilt, wird in der Konfigurationsdatei des AT-TLS Policy Agent für den vom Natural HTTPS Client verwendeten z/OS TCP/IP Stack festgelegt.

IBM liefert bei jeder z/OS-Systemauslieferung einen Satz normalerweise verwendeter CA-Root-Zertifikate mit. Wenn Key Rings zum Vorhalten von Server-Zertifikaten benutzt werden sollen, müssen diese Root-Zertifikate durch den Systemadministrator manuell in die Key Rings importiert werden. Wenn IBM neuere Ersatzzertifikate für abgelaufene Root-Zertifikate liefert, müssen alle betroffenen Key Rings entsprechend aktualisiert werden.

Im Gegensatz zu Key Rings enthalten Key-Datenbanken automatisch den aktuellen Satz Root-Zertifikate, nachdem diese neu erstellt worden sind. Jedoch besteht auch bei der Key-Datenbankalternative die Notwendigkeit, immer den neuesten Satz Root-Zertifikate zu pflegen.

Vom Natural HTTPS Client zu verwendende Zertifikate müssen per Flag als "Trusted" gekennzeichnet werden. Wenn sie Teil der Public Key Infrastructure sind, dann muss das entsprechende CA-Root-Zertifikat als "Trusted" gekennzeichnet sein.

Verwendung von RACF Key Rings

In RACF werden digitale Zertifikate in so genannten Key Rings gespeichert. Das RACF-Kommando `RACDCERT` wird verwendet, um Key Rings und in die in diesen Key Rings enthaltenen Zertifikate zu verwalten.

Siehe *z/OS Security Server RACF Security Administrator's Guide*, IBM manual SA22-7683-11, und *z/OS Security Server RACF Command Language Reference*, IBM manual SA22-7687-11.

Verwendung von Key-Datenbanken

Alternativ zu RACF können Zertifikate in Key-Datenbanken vorgehalten werden, die auf dem z/OS UNIX Services-Dateisystem liegen. Zum Erstellen und Verwalten von Key-Datenbanken muss die `GSKKYMAN`-Utility benutzt werden.

Siehe *z/OS Cryptographic Services PKI Services Guide and Reference*, IBM manual SA22-7693-10.

Einschränkung bezüglich IMS/TM

The following restriction applies if you wish to use the Natural statements `REQUEST DOCUMENT` and `PARSE XML` in an IMS/TM environment:

- The `PARSE XML` statement can be executed under the TP monitor IMS/TM with the restriction that no I/O statement is allowed within an active `PARSE` loop. If an I/O occurs within a `PARSE` loop, error NAT0967 is issued.

For further restrictions, see the corresponding notes in the statement descriptions.

Voraussetzung für die Unterstützung von Statements für den Internet- und XML-Zugriff unter *open*UTM

During an active parse loop with I/Os, the UTM function call PGWT must be used. This means:

1. The UTM application must be started with not less than 2 tasks, otherwise a UTM error K319 with subsequent dump will occur.
2. PGWT conditions must be defined for the KDCDEF.
    1. Define the maximum wait time (in seconds) for input messages during a PGWT call.

       Example:

       ```
       MAX PGWTTIME=60
       ```

    2. Define the maximum number of UTM tasks for PGWT calls.

       Example:

       ```
       MAX TASKS-IN-PGWT=1
       ```

    3. PGWT can be controlled using either the TAC-PRIORITIES instruction or the TACCLASS concept:

       - Control of PGWT using the TAC-PRIORITIES instruction:

         Example:

         ```
         DEFAULT TAC TYPE=D,PROGRAM=NATUTM,. . . . . .
         TAC NAT,ADMIN=NO,TIME=(0,0),PGWT=YES,TACCLASS=1
         TAC-PRIORITIES DIAL-PRIO=EQ
         ```

       - Control of PGWT using the TACCLASS concept:

         Example:

         ```
         DEFAULT TAC TYPE=D,PROGRAM=NATUTM,. . . . . .
         TAC NAT,ADMIN=NO,TIME=(0,0),TACCLASS=1
         TAC NAT1,ADMIN=NO,TIME=(0,0),TACCLASS=2
         TACCLASS 1,TASKS=2
         TACCLASS 2,TASKS=1,PGWT=YES
         ```

3. The keyword subparameter ILCS of parameter macro NURENT must set to ILCS=CRTE.

Programmbeispiel

Das folgende Programm ist ein Beispiel für die Anwendung der Statements REQUEST DOCUMENT und PARSE XML.

Weitere Programmbeispiele finden Sie jeweils am Ende einer Statement-Beschreibung in der *Statements*-Dokumentation und in der Natural Library SYSEXV.

```
DEFINE DATA
LOCAL
1 #FROM   (A) DYNAMIC
1 #HEADER (A) DYNAMIC
1 #PAGE   (A) DYNAMIC
1 #RC     (I4)
1 #COL    (N8)
1 #COL1   (I4)
1 #COL2   (I4)
1 #COL3   (I4)
1 #LOC    (A30)
1 #CP     (A) DYNAMIC
1 #PATH   (A) DYNAMIC
1 #NAME   (A) DYNAMIC
1 #VALUE  (A) DYNAMIC
1 #RTERR  (I4)
END-DEFINE
*
ASSIGN #FROM = 'HTTP://SI15.HQ.SAG/autos6.xml'
**
REQUEST DOCUMENT FROM #FROM
RETURN
HEADER ALL #HEADER
PAGE #PAGE    ENCODED    FOR TYPES 'TEXT/XML'
 CODEPAGE ' '
RESPONSE #RC
GIVING #RTERR
**
IF #RC NE 200 /* TEST FOR HTTP RESPONSE 200 = 'OK'
WRITE 'HTTP RESPONSE' #RC 'RECEIVED'
ESCAPE ROUTINE
END-IF
EJECT
PRINT #HEADER
/ '_'(79)
PRINT #PAGE
/ '_'(79)
/ '_'(79)
ASSIGN #CP = *CODEPAGE
EXAMINE #PAGE FOR 'encoding' GIVING POSITION  #COL1
 IF #COL1 GT 0
   EXAMINE #PAGE FOR '?>' GIVING POSITION #COL3
   IF #COL3 GT #COL1
     EXAMINE #PAGE FOR 'ISO-8859-1' GIVING POSITION #COL2
   END-IF
   IF #COL2 GT #COL1 AND #COL2 LT  #COL3
     EXAMINE #PAGE FOR 'ISO-8859-1' REPLACE #CP
   END-IF
 END-IF
PRINT #PAGE
/ '_'(79)
EJECT
PARSE XML #PAGE INTO PATH #PATH NAME #NAME VALUE #VALUE
 PRINT #PATH / 'NAME=' #NAME / 'VALUE=' #VALUE / '_'(79)
END-PARSE
END
```

Anmerkung:
Die URL, auf die im obigen Programm zugegriffen wird, adressiert eine Intranet Site und kann nicht aus dem Internet aufgerufen werden.

Ausgabe des Beispielprogramms:

```
HTTP/1.1 200 OK?Date: Thu, 10 Aug 2006 16:26:22 GMT?Server: Apache/1.3.19 (
BS2000)?Last-Modified: Thu, 27 Jul 2006 16:44:42 GMT?ETag: "2602c-111-44c8ed7a"
?Accept-Ranges: bytes?Content-Length: 273?Connection: close?Content-Type: text/
xml??
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?><autos>?<make></make>?<make>Ford</
make>?<model>Thunderbird</model>?<make>Merceds-Benz</make><model>S400</model><
make>BWM</make><model version="latest">330I</model>?<make><label><company>
Mercedes</company></label></make>?</autos>?
```

```
<?xml version="1.0" encoding="IBM01140" ?><autos>?<make></make>?<make>Ford</
make>?<model>Thunderbird</model>?<make>Merceds-Benz</make><model>S400</model><
make>BWM</make><model version="latest">330I</model>?<make><label><company>
Mercedes</company></label></make>?</autos>?
```

```
MORE

autos
Name= autos
Value=
```

```
autos/$
Name=
Value= ?
```

```
autos/make
Name= make
Value=
```

```
autos/make//
Name= make
Value=
```

```
autos/$
Name=
Value= ?
```

```
autos/make
Name= make
Value=
VVVV
Name= autos
Value=
```

```
autos/$
Name=
Value= ?
```

```
autos/make
Name= make
Value=
```

Häufig gestellte Fragen

- Warum muss Codepage-Unterstützung eingeschaltet sein?
- Wie sind die XML-Schlüsselwort-Parameter zu benutzen (z.B. RDP und RDNOP)
- Wo erhalte ich Angaben zu Proxy-Server, Port-Nummer und HTTP-Server einer Site?
- Woran erkenne ich, ob ein Problem mit TCP/IP, mit HTTP oder mit Natural vorliegt?
- Kann ich prüfen, ob ich eine Website von meinem Großrechner aus erreiche, ohne dazu Natural zu benutzen?
- Wird NAT2TCP korrekt geladen?
- Ich erhalte eine Meldung "unsupported coding"
- Wie kann ich bei REQUEST DOCUMENT die Ausgabe des Natural-Fehlers NAT3411 vermeiden?
- Kann ich selbstsignierte Zertifikate benutzen?
- Welches Methode ist für die Pflege von Zertifikaten zu bevorzugen?
- Wie konfiguriere ich TCP/IP für AT-TLS?
- Wie überprüfe ich die AT-TLS-Konfiguration?
- Wo finde ich weitere Informationen zur Problembestimmung?
- Wie schalte ich den Policy-Agent Trace ein?
- Fehler beim Verbindungsaufbau

Warum muss Codepage-Unterstützung eingeschaltet sein?

In der Installationsdokumentation für Natural für Großrechner steht, dass der ICU Handler zum Natural-Nucleus gelinkt sein muss.

PARSE XML-Statement

Die Codepage-Unterstützung wird benötigt, weil auf Großrechnerplattformen das zu parsende Dokument intern immer nach UTF-16 konvertiert wird (falls das Dokument nicht schon in UTF-16 kodiert ist). Meistens jedoch ist das Dokument nicht in UTF-16, und es findet eine Konvertierung statt. Ausführliche Informationen hierzu finden Sie in der Beschreibung des PARSE XML-Statements in der *Statements*-Dokumentation und im Abschnitt `PARSE XML` in der Dokumentation Unicode and Code Page Support.

REQUEST DOCUMENT-Statement

Die ICU Library wird benötigt, um von Außen eintreffende HTTP Headers zu interpretieren und nach Außen abgehende HTTP Headers zu konvertieren. Die herein kommenden Headers sind normalerweise in ISO 8859-1 kodiert und müssen auf dem Großrechner immer in die Natural-Default-Codepage (siehe auch Natural-Systemvariable `*CODEPAGE`) konvertiert werden - auf dem PC ist eine Konvertierung dagegen

nicht immer nötig.

Wie sind die XML-Schlüsselwort-Parameter zu benutzen (z.B. RDP und RDNOP)

Auf dem PC führt das `REQUEST DOCUMENT`-Statement den Browser (Internet Explorer) aus und verwendet dabei die dort vorhandenen Einstellungen.

Auf dem Großrechner muss die URL des (Intranet-)Proxy-Servers, über den alle Anfragen geleitet werden müssen, mit dem `NTXML/XML`-Schlüsselwort-Subparameter `RDP` angegeben werden. Mit dem Schlüsselwort-Subparameter `RDNOP` kann man eine oder mehrere lokale Domain(s) festlegen, die direkt und nicht über den Proxy-Server adressiert werden soll(en).

Wo erhalte ich Angaben zu Proxy-Server, Port-Nummer und HTTP-Server einer Site?

Informationen über Proxy-Server, Port-Nummer und HTTP-Server einer Site müssen Sie beim Netzwerkadministrator erfragen.

Sie können aber auch in Ihrem Browser nachsehen, welcher Proxy-Server dort für Ihre Site definiert ist.

Zum Beispiel im Internet Explorer unter: Tools > Internet Options > Lan Settings > Advanced

bzw. bei deutscher Oberfläche unter: Extras > Internetoptionen > Verbindungen > Einstellungen > Einstellungen für lokales Netzwerk (LAN)

Außerdem können Sie im Web nach Tools suchen, die Ihnen diese Informationen liefern. Zum Beispiel hier (ungeprüft): *http://www.sharewareconnection.com/titles/proxy-settings.htm*

Woran erkenne ich, ob ein Problem mit TCP/IP, mit HTTP oder mit Natural vorliegt?

HTTP Response Codes

Die HTTP-Rückmeldung erfolgt über die `RESPONSE`-Klausel in *operand16*. Eine *Übersicht über Response-Nummern für HTTP/HTTPs-Anfragen* finden Sie in der Beschreibung des `REQUEST DOCUMENT`-Statements in der *Statements*-Dokumentation.

TCP/IP-Fehler

Der Nummernkreis für diese Fehler beginnt bei NAT8300.

Insbesondere der Fehler NAT8304 liefert ausführlichere Angaben zu einer fehlgeschlagenen HTTP-Anfrage.

Da die TCP/IP-Fehlermeldungen je nach Installationsumgebung unterschiedlich sein können, ist der in NAT8304 zurück gegebene Text die beste Informationsquelle.

Weitere Informationen:

- Siehe Buffer RDOCWA bei Offset 480
- Sehr oft handelt es sich bei diesen Fehlern um ICU-Fehler: Daher wird empfohlen, den Natural-Sessionparameter CPCVERR auf OFF zu setzen..

Kann ich prüfen, ob ich eine Website von meinem Großrechner aus erreiche, ohne dazu Natural zu benutzen?

Um festzustellen, ob ein Problem an der Natural-Installation liegt oder ob es sich um ein generelleres Problem handelt, können Sie einen PING aus TSO heraus absetzen.

Geben Sie zum Beispiel in der TSO Command Shell folgendes Kommando ein:

```
TSO PING www.google.com
```

Die Antwort lautet:

```
CS V1R9: Pinging host WWW.GOOGLE.COM (66.249.91.99)
Ping #1 response took 0.018 seconds.
```

Aus der Natural-Session heraus können Sie den Zugang zu dieser Website mit dem folgenden kleinen Programm testen.

Starten Sie Natural zum Beispiel mit:

```
NATvr CFICU=ON
XML=(ON,RDOC=ON,PARSE=ON,RDP='HTTPPROX.HQ.SAG',RDPPORT=8080,RDNOP='*.EUR.AD.SAG;
*.HQ.SAG;*.SOFTWAREAG.COM')
```

dabei steht *vr* für die Natural-Release- und Versionsnummer.

Diese Werte einer internen Umgebung und ein Profil wurden benutzt, um es zu speichern. Für Ihre Zwecke müssen Sie die korrekten Werte für die Schlüsselwort-Subparameter RDP, RDPPORT and RDNOP bei Ihrem Netzwerk-Administrator erfragen oder probieren Sie es mit den in Ihrem Browser (Internet Explorer) definierten Werten.

Führen Sie das folgende Programm aus:

```
DEFINE DATA LOCAL
1 #RESULTXML (A) DYNAMIC
1 #RC (I4)
END-DEFINE
REQUEST DOCUMENT FROM "HTTP://WWW.GOOGLE.DE"
RETURN HEADER ALL #HEADER RESPONSE #RC
WRITE #RC
WRITE #HEADER (AL=79)
END
```

Wird NAT2TCP korrekt geladen?

Um zu überprüfen, ob das Modul NAT2TCP korrekt geladen wird, können Sie die Utility SYSPROD benutzen.

In SYSPROD geben Sie das Kommando SC (Display subcomponents) für das Produkt Natural ein. Wenn Sie die installierten Subkomponenten durchblättern, finden Sie einen Eintrag für Nat Request Document (Product ID .... TCP).

Ich erhalte eine Meldung "unsupported coding"

Der Grund für diese Meldung liegt in einem häufig gemachten Benutzerfehler: Ein XML-Dokument wird implizit oder explizit von einer Codepage in eine andere konvertiert, zum Beispiel von ISO-8859-1 in die Codepage, die in der Systemvariablen *CODEPAGE vorgefunden wurde. Die Kodier-PI des Dokuments PI encoding="ISO-8859-1" wurde jedoch nicht an die geänderte Kodierung angepasst. In disem Fall beendet der Parser den Vorgang und gibt bereit beim ersten Zeichen des zu parsenden Dokuments eine Fehlermeldung aus.

Wie kann ich bei REQUEST DOCUMENT die Ausgabe des Natural-Fehlers NAT3411 vermeiden?

Setzen Sie den Sessionparameter CPCVERR auf OFF.

Kann ich selbstsignierte Zertifikate benutzen?

Selbstsignierte Zertifikate können auf einem Intranet-Server unter Verwendung des open ssl sdk für Testzwecke benutzt werden. Nachdem sie in eine Key-Datenbank oder einen RACF Key Ring importiert wurden, müssen sie mit einem "Trusted" Flag versehen werden.

Welches Methode ist für die Pflege von Zertifikaten zu bevorzugen?

Der notwendige Pflegeaufwand für RACF Key Rings scheint viel höher zu sein als bei der Verwendung von Key-Datenbanken. Key Rings müssen für jeden Benutzer erstellt werden, der auf einen HTTPS-Server zugreifen möchte, während Key-Datenbanken von mehreren Benutzer gemeinsam genutzt werden können.

Wie konfiguriere ich TCP/IP für AT-TLS?

Dazu führen Sie folgende Schritte aus:

1. In der TCP/IP-Konfiurationsdatei: Setzen Sie die Option TTLS im TCPCONFIG-Statement.
2. Konfigurieren und starten Sie den AT-TLS Policy Agent. Dieser Agent wird von TCP/IP bei jeder neuen TCP-Verbindung aufgerufen um zu prüfen, ob es sich um eine SSL-Verbindung handelt.
3. Erstellen Sie die Policy Agent-Datei, die die AT-TLS-Regeln enthält. Diese Datei enthält Regeln zum Festlegen, welche Verbindung über SSL erfolgt. T

Siehe auch *z/OS Communications Server: IP Configuration Guide*, Chapter 18 *Application Transparent Transport Layer Security (AT-TLS) data protection*.

Die folgende Sample Policy Agent-Datei definiert alle nach Außen abgehenden Verbindungen als anwendungsgesteuerte TLS. Dies sollte außer der Natural-REQUEST DOCUMENT-Unterstützung keine andere TCP/IP-Anwendung betreffen, weill die Regel als anwendungsgesteuert definiert ist. Das bedeutet, dass die Anwendung den Verbindungsstatus auf SSL setzen kann. Solange wie die Anwendung diesen Status nicht setzt, ist sie nicht betroffen. Die Policy Agent-Datei gestattet es außerdem, die anwendungsgesteuerten SSL-Verbindungen auf bestimmte Ports, Benutzer oder Adressräume zu

beschränken. Bei diesem Beispiel wird vorausgesetzt, dass die Zertifikat-Datenbank in der HFS-Datei `/u/admin/CERT.kdb` liegt.

```
TTLSRule                          ConnRule01~1
{
  LocalAddrSetRef                 addr1
  RemoteAddrSetRef                addr1
  LocalPortRangeRef               portR1
  Direction                       Outbound
  Priority                        255
  TTLSGroupActionRef              gAct1~AllUsersAsClient
  TTLSEnvironmentActionRef        eAct1~AllUsersAsClient
  TTLSConnectionActionRef         cAct1~AllUsersAsClient
}
TTLSGroupAction                   gAct1~AllUsersAsClient
{
  TTLSEnabled                     On
  Trace                           6
}
TTLSEnvironmentAction             eAct1~AllUsersAsClient
{
  HandshakeRole                   Client
  EnvironmentUserInstance         0
  TTLSKeyringParmsRef             keyR1
}
TTLSConnectionAction              cAct1~AllUsersAsClient
{
  HandshakeRole                   Client
  TTLSCipherParmsRef              cipher1~AT-TLS__Silver
  TTLSConnectionAdvancedParmsRef  cAdv1~AllUsersAsClient
  Trace                           0
}
TTLSConnectionAdvancedParms       cAdv1~AllUsersAsClient
{
  ApplicationControlled           On
}
TTLSKeyringParms                  keyR1
{
  Keyring                         /u/admin/CERT.kdb
  KeyringStashFile                /u/admin/CERT.sth
}
TTLSCipherParms                   cipher1~AT-TLS__Silver
{
  V3CipherSuites                  TLS_RSA_WITH_DES_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_3DES_EDE_CBC_SHA
  V3CipherSuites                  TLS_RSA_WITH_AES_128_CBC_SHA
}
IpAddrSet                         addr1
{
  Prefix                          0.0.0.0/0
}
PortRange                         portR1
{
  Port                            1024-65535
}
```

Wie überprüfe ich die AT-TLS-Konfiguration?

Prüfen Sie den Policy-Agent Job Output `JESMSGLG` auf:

```
EZZ8771I PAGENT CONFIG POLICY PROCESSING COMPLETE FOR <your TCP/IP address space>: TTLS
```

Diese Meldung zeigt die erfolgreiche Initialisierung an.

Prüfen Sie den Policy-Agent Job Output `JESMSGLG` auf:

```
EZZ8438I PAGENT POLICY DEFINITIONS CONTAIN ERRORS FOR <your TCP/IP address space>: TTLS
```

Diese Meldung weist auf Fehler in der Konfigurationsdatei hin. Prüfen Sie bitte die Datei `syslog.log` auf weitere Hinweise.

Umfasst die Konfigurationsregel auch den Client?

Prüfen Sie `syslog.log` auf:

```
EZD1281I TTLS Map   CONNID: 00002909 LOCAL: 10.20.91.61..1751 REMOTE: 10.20.91.117..443
JOBNAME: KSP USERID: KSP TYPE: OutBound STATUS: Appl Control RULE: ConnRule01
ACTIONS: gAct1 eAct1 AllUsersAsClient
```

Der obige Eintrag zeigt an, dass die Verbindung zu Port 443 des Benutzers `KSP` von der Anwendung gesteuert wird.

Wo finde ich weitere Informationen zur Problembestimmung?

Siehe auch *z/OS V1R8.0 Comm Svr: IP Diagnosis Guide: 3.23*, Chapter 29 *Diagnosing Application Transparent Transport Layer Security (AT-TLS)*

Wie schalte ich den Policy-Agent Trace ein?

Informationen hierzu finden Sie im *Comm Svr: IP Configuration Reference*, Chapter 20 *Syslog deamon and Comm Svr: IP Configuration Guide*, Chapter 1.5.1 *Configuring the syslog daemon (syslogd)*

Fehler beim Verbindungsaufbau

Suchen Sie den Return Code `RC` und den entsprechenden `GSK_`-Funktionsnamen im Policy-Agent Trace.

Schlagen Sie im *System SSL Programming* im Chapter 12.1 *SSL Function Return Codes* den Return Code `RC` nach.

Beispiel-Trace zu `trace=255`:

```
EZD1281I TTLS Map   CONNID: 00002909 LOCAL: 10.20.91.61..1751 REMOTE: 10.20.91.117..443 JOBNAME: KSP USERID: KSP TYPE: OutBound STATUS: A
EZD1283I TTLS Event GRPID: 00000003 ENVID: 00000000 CONNID: 00002909  RC:    0 Connection Init
EZD1282I TTLS Start GRPID: 00000003 ENVID: 00000002 CONNID: 00002909 Initial Handshake ACTIONS: gAct1 eAct1 AllUsersAsClient HS-Client
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Call GSK_SECURE_SOCKET_OPEN - 7EE4F718
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Set GSK_SESSION_TYPE -  CLIENT
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Set GSK_V3_CIPHER_SPECS -  090A2F
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Set GSK_FD - 00002909
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Set GSK_USER_DATA - 7EEE9B50
EZD1284I TTLS Flow  GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:  435 Call GSK_SECURE_SOCKET_INIT - 7EE4F718
EZD1283I TTLS Event GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:  435 Initial Handshake 00000000 7EEE8118
EZD1286I TTLS Error GRPID: 00000003 ENVID: 00000002 CONNID: 00002909 JOBNAME: KSP USERID: KSP RULE: ConnRule01  RC:  435 Initial Handshake
EZD1283I TTLS Event GRPID: 00000003 ENVID: 00000002 CONNID: 00002909  RC:    0 Connection Close 00000000 7EEE8118
```

Weitere Informationsquellen

Die folgende Aufstellung enthält Hinweise auf weitere nützliche Informationsquellen:

- Schulungskurse
- Nützliche Links

Schulungskurse

Die Schulungsabteilung der Software AG bietet spezielle Schulungskurse zu diesem Thema an. Ausführlichere Informationen siehe Empower unter *https://empower.softwareag.com/*.

Oder wenden Sie sich an Ihre örtliche Software AG-Vertretung wegen spezieller Schulungen vor Ort.

Nützliche Links

Allgemeine Informationen finden Sie auf den folgenden Web-Seiten:

- World Wide Web Consortium (W3C): *http://www.w3.org/*
- Extensible Markup Language (XML): *http://www.w3.org/XML/*
- HyperText Markup Language (HTML) Home Page: *http://www.w3.org/MarkUp/*
- W3 Schools: *http://www.w3schools.com/*