Connect:Express

Java application interface

Version 1.3.6

Sterling Commerce
An IBM Company

Connect:Express
Java Application Interface

Version 1.3.6
First Edition

Contents

Preface

This document describes the use of the Connect:Express JAI, which enables Java client applications to access remotely to the different functions of a Connect:Express monitor.

Chapter 1 provides a general description of the JAI.
Chapter 2 shows examples of programs using the JAI.
Appendix 1 describes the data of the Connect:Express monitors.

The current version of the JAI only supports Connect:Express Unix and Windows monitors. Next versions will also give the possibility to access to Connect:Express OS/390. However, this document refers to the behavior of the JAI for each type of monitor.

The Connect:Express JAI requires a version of the JDK greater or equal to JDK 1.4.2.

Introduction

The Connect:Express JAI allows Java applications to control remote Connect:Express Unix, Windows and OS/390. Each remote monitor (server) is represented by a CxServer class. This class provides methods that enable the application to display and update the various components of the monitor. The Application can perforrm the following operations:

- Display, create, update and delete the symbolic partners and the symbolic files
- Display and update the session and presentation tables
- Display the elements of the static configuration of the monitor
- Display active transfers, the journal records and the statistics records
- Submit transfer requests
- Interrupt, restart and purge transfers

The connections to the remote Connect:Express monitors use TCP/IP.

Next chapter describes these functions after a general presentation of how to use the JAI. The detailed description of the JAI can be found in the java documentation CXJAIDOC.jar.

The objects provided for the JAI are:

- CXJAI.jar Class files of the JAI
- CXJAIDOC.jar JAI documentation
- CXJAI_guide.pdf This document
- CXJAI_examples.jar Examples of programs using the JAI

This chapter gives a general description how to use the JAI.

Presentation

In order to access a monitor, the application performs the following operations:

- Connection to the monitor
- Function calls
- Disconnection

The application connects to a monitor by instantiating a CxServer class. It provides the necessary elements for the network and logging connections in the constructor's parameters.

The different calls to monitor use the appropriate methods of the CxServer class. Each component of the monitor is defined by a class that enables access to its different elements. These classes take place in the values provided (requests) or returned (responses) when using the methods of CxServer.

When the operations are completed, the application disconnects from the monitor by calling the method disconnect() of CxServer.

Example

The following example shows the different JAI calls mentioned above:

```
...
import com.sterlingcommerce.cx.sdk.*;
...


    ...
    //Connect to server
    Char[] pwd = {'A','D','M','I','N'};
      String c_host = "192.168.0.18";
      int c_port = 9000;
      CxServer = new CxServer(c_host,c_port,"ADMIN",pwd,"TCPIP",0,true);


      // Get symbolic partner BOUCLE
      CxPartner partner = srv.getPartner("BOUCLE");
      System.out.println("Protocol = "+partner.getProtocol());
```

```
// Get symbolic file FILE01
CxFile file = srv.getFile("FILE01");
System.out.println("LocalPhysicalName = "+file.getLocalPhysicalName());

// Disconnect from server
srv.disconnect();
...
```

Classes of the JAI

The following table shows the list of the different classes and interfaces of the JAI:

Classe / interface	**Description**
CxActiveTransfer	Describes an active transfer of Connect:Express
CxActivityFilter	Defines the selection criteria used for requesting a list of active transfers
CxActivityListElement	Describes an element of the list of active transfers returned by the server
CxAPKey	Describes the asset protection key of the Connect:Express server
CxConnectionInformation	Provides information on the current connection of the JAI with the server
CxEtebac3Presentation	Describes an Etebac3 presentation table of a Connect:Express Windows server
CxFile	Describes a symbolic file
CxJournal	Describes a journal record
CxJournalFilter	Defines the selection criteria used for requesting a list of transfers from the journal
CxJournalListElement	Describes an element of a list of journal records returned by the server
CxPartner	Describes a symbolic partner
CxPresentation	Describes a presentation table
CxPurgeFilter	Defines the selection criteria used for requesting a purge of transfers
CxServer	Main class that identifies the connection to the remote monitor
CxServerConfig	Describes the elements of the monitor's configuration
CxSession	Describes a session table
CxStatistics	Describes statistics record
CxStatisticsFilter	Defines the selection criteria used for retreiving a set of statistics records
CxTransferRequest	Defines the parameters used for submitting a transfer request
CxEnumeration	Enumeration of a set of statistics records returned by the server
CxException	Root class of the exceptions thrown by the JAI
CxConnectionException	Connection error exception
CxLogonException	Logon error exception

Classe / interface	Description
CxServerException	Exception thrown when the JAI has received an indication of error from the server (except logon)

OS Type

Although Connect:Express monitors data is very similar on all three platforms, some fields are specific to each type of operating system (Unix, Windows or OS/390). For example, the maximum length of the file physical name (PhysicalName) is 44 characters for Connect:Express OS/390 and Unix, and 127 characters for Connect:Express Windows.

Moreover, the monitors do not support all the same functionnalities. For example, the FTP protocol is available with Connect:Express OS/390 and Unix but not available with Connect:Express Windows. The network protocol LU6.2 is available with Connect:Express OS/390 and Windows and not with Connect:Express Unix.

The JAI manages these differences.

When the JAI connects to the server, it receives a string that indicates the type of OS of the server: “WINDOWS”, “UNIX” or “OS390”.

```
...
        String ostype = srv.getOSType();
...
```

The java documentation of the JAI (provided in CXJAIDOC.jar) indicates, for each method, the type of OS for which it is valid.

Besides, the documentation of the « setter » methods indicates the maximum length of the parameter depending on the type of OS.

For example, the method setEndTransmitCommand of the CxFile class is documented as shown below:

setEndTransmitCommand

public void **setEndTransmitCommand**(java.lang.String endTransmitCommand)
throws CxInvalidArgumentException

Sets the user command called at end of transmission.

OS: Windows, Unix, OS390.

Windows: Maximum length: 127 characters.

Unix: Maximum length: 12 characters.

OS390: Maximum length: 32 characters.

Parameters:

endTransmitCommand - the name of the command.

Throws:

CxInvalidArgumentException - if the length of the parameter is invalid.

The JAI controls that the call to the “setter” method as well as the length of the parameter matches the OS type. In case of error, it throws a CxInvalidArgumentException exception.

For the “getter” methods, the JAI does not throw any exception if the corresponding data is not defined for the platform, and it returns a null value (String “”, int 0, char ‘ ‘). This allows you to write a display application, partly transparent with regard to the OS type, since, for example, a data string which is not defined for an OS will display as an empty string.

The tables in Appendix 1 show the availability and the length of each data of Connect:Express depending on the operating system.

Exceptions

The exceptions of the JAI derive from the root class CxException, which derives itself from java.lang.Exception. These exceptions are:

CxConnectionException

The JAI throws this exception when it is unable to connect to the server, when there is a network failure during the connection or when there is a protocol error during the dialog between the JAI and the server.
If this exception occurs, the application must retry the connection later, by instantiating a new CxServer class.

CxLogonException

The JAI throws this exception when the user identification at the server fails. If this exception occurs, the application must connect again to the server with a new instance of CxServer, and provide a correct user name and a correct password.

Note: in the current connectExpress versions, the control of identification is only effective when logging to Connect:Express Windows.

CxInvalidArgumentException

The JAI throws this exception when there is an error in the parameters of a method of the JAI. This exception results of a local control by the JAI of the parameters provided by the calling program. If this exception occurs, the calling program must be fixed.

CxServerException

The JAI throws this exception when the monitor returns an error on a valid connection. This error corresponds to a "logical" error detected by the monitor. For example:

- Attempt to create an existing symbolic file
- Error detected in the submission parameters of a transfer request

In this case, the exception contains the error code in its field "mid" (message id). This value is issued by the monitor (in fact this code is a Connect:Express TRC), as well as an explanation message in the default local language.
The message id can be obtained by the method CxServerException.getMid().
The message text can be obtained by the method CxServerException.getMessageText().

The Connect:Express user guides give the meaning of the TRCs which can be returned by the monitor for the various platforms.

Note: Normally the methods of the CxServer class catch the other java exceptions, in order to throw only the exceptions mentioned above. Nevertheless, it is recommended that the calling application uses a last block catching java.lang.Exception (for example for tracing the stack).

The calling program can catch the exceptions in the following way:

```
CxServer srv = null;
try {
     ...
     //Connect to server
     Char[] pwd = {'A','D','M','I','N'};
       String c_host = "192.168.0.18";
       int c_port = 9000;
       srv = new CxServer(c_host,c_port,"ADMIN",pwd,"TCPIP",0,true);

       // Get symbolic partner BOUCLE
       CxPartner partner = srv.getPartner("BOUCLE");
       System.out.println("Protocol = "+partner.getProtocol());
```

```
      // Get symbolic file FILE01
      CxFile file = srv.getFile("FILE01");
      System.out.println("LocalPhysicalName = "+file.getLocalPhysicalName());

      // Disconnect from server
      srv.disconnect();
    srv = null;
    ...
} catch (CxServerException e) {
    // Depending on the server message id, processing can continue or not
    // See documentation on the Trc codes that can be returned by Connect:Express
    System.out.println("Message id = "+e.getMid());
    System.out.println("Message text = "++e.getMessageText());
    ...
} catch (CxLogonException e) {
    // Instantiate a new CxServer with different user/password ?
    ...
} catch (CxConnectionException e) {
    // Reconnect ?
      ...
} catch (CxInvaliArgumentException e) {
      // Fatal error. Correct the program
      ...
} catch (Exception e) {
      // Fatal error
      e.printStackTrace();
} finally {
      if (srv!=null} {
         Try {
              Srv.disconnect();
         } catch (Exception) {
         }
      }
}
...
```

Lists

The CxServer class enables the application to retrieve lists for the following elements:

- List of the symbolic partners (listPartners)
- List of the symbolic files (listFiles)
- List of the session tables (listSession)
- List of the presentation tables (listPresentations)
- List of the Etebac3 presentation tables of Connect:Express Windows (listEtebac3Presentations)
- List of the active transfers (selectActivity)
- List of the journal (selectJournal)

These methods return objects of type array, which can contain either a simple list of identifiers (for example the list of the partner names), or a set of more complete elements (list of active transfers or list of the journal).
Once the application has retrieved a list, it can retrieve the detail of a specific element from the server.

An example of getting a list is shown below:

```
...
// PARTNERS
// Get the list of the symbolic partners names
String[] = srv.listPartners();

// For each partner name, get the detail of the partner
for (i=0;i<partnerlist.length;i++) {
       // Get the detail of each partner
   CxPartner partner = srv.getPartner(partnerlist[i]);
   ...
}


// JOURNAL
// Get the list of all elements of the journal
// Selection criteria
CxJournalFilter filter = new CxJournalFilter(CxServer.UNIX);
filter.setFileName("*");
filter.setPartnerName("*");
filter.setDirection("*");
filter.setMinimumDate("*");
filter.setMaximumDate("*");
filter.setStatus("*");

CxJournalListElement[] journallist = srv.selectJournal(filter);
// Each journal element already contains basic information
// such as requestNumber,...

for (i=0;i<journallist.length;i++) {
   CxJournalListElement elem = journallist[i];
       // Get the full detail of the journal for this element
       // The key is the request number
   CxJournal jnl = srv.getJournal(elem.getRequestNumber());
   ...
}
...
```

Statistics

The application can retrieve statistics by providing, either selection criteria (CxStatisticsFilter), or a number of seconds, to the method selectStatistics of the CxServer class.
Using criteria allow to make the selection according to a range of dates or from a transfer request number. Using a number of seconds allows to select only the statistics in the range of the last elapsed seconds.
The JAI returns results in an enumeration. Each element of this enumeration is a "statistics record" object CxStatistics.

A statistics record is made up of a set of elements presented in the form keyword=value, separated by commas. Appropriate methods of the CxStatistics class give the possibility to extract in different array forms the elements of each line.

The following example shows how to get statistics from the server:

```
...
// Define minimum and maximum date
Date minimumDate = ...;
Date maximumDate = ...;

// Define a statistics filter for the selection
CxStatisticsFilter filter = new CxStatisticsFilter(CxServer.UNIX);
filter.setMinimumDate(minimumDate);
filter.setMaximumDate(maximumDate);

CxStatistics stat;
CxEnumeration senum = srv.selectStatistics(filter);
while (senum.hasMoreElements()==true) {
   Stat = (CxStatistics)senum.nextElement();
   System.out.println("statline = "+stat.toString();
   String[][] lvp = stat.getLabelValuePairs();
   for (int ii=0;ii<keys.length;ii++) {
        System.out.println(lvp[ii][0]+" = "+lvp[ii][1]);
   }
}
...
```

Note:
Inside the *while* loop above, the TCP/IP link with the server is busy. Indeed, as long as hasMoreElements() returns true, the JAI is likely to receive statistics records from the server.
Therefore, it is not possible to use inside this loop other JAI methods than nextElement().

Limitation on the Number of Records Returned

Getting too many elements from the journal or the statistics can consume too many resources and increase response time for the client application and the remote server. It is possible to fix a limit on the number of returned elements by calling the method setLimit of CxServer.
The behavior of both methods is changed by setLimit:

- CxServer.selectJournal: Maximum number of elements which can be returned in the list.
- CxServer.selectStatistics: Maximum number of statistics records which can be returned in the enumeration.

Traces

The application can activate or deactivate JAI traces by setting a trace flag:

- In the constructor of the CxServer class
- Dynamically, by calling CxServer.setTrace

The JAI directs the trace to a file cxjai.<server-address>.<server-port>.trc in the current execution directory of the application or to the user output depending on the presence of a flag file cxjaitrc.flag in this directory. The JAI traces the network data exchanged with the server.

Replacement of Unix environment variables.

Local physical name :
When a Unix server retrieves an element of the journal (class CXJournal), the physical name of the transferred file is set in two equivalent properties (LocalPhysicalName and XlocalPhysicalName). The first property contains the physical name without replacement of Unix environment variables. In the second, environment variables are replaced.
The maximum length of the string XlocalPhysicalName is 512. In case of overflow, the property XlocalPhysiclNameOvf is set to true.

getEnvironmentValue et replaceEnvironment :
The class CXServer contains 2 methods enabling to get the environment variables of the server.
getEnvironmentValue returns the value of an environment variable given as parameter.
replaceEnvironment takes as parameter a string containing environment variables and returns an equivalent string where variables have been replaced.

Transfer Request

The application can submit a transfer request by defining the parameters of the transfer in an object of the CxTransferRequest class, then by calling the method submitTransfer with this object as parameter.
The transfer requests of Connect:Express have specifics sets of parameters for each type of operating system of the server. The JAI controls the validity of each parameter with respect to the server OS type.
To get more details on the parameters of the transfer requests, refer to the Connect:Express user guide for the operating system involved .

The following example shows the submission of a transfer request for Connect:Express Unix:

```
...
// Transfer request
...
// Set the parameters
CxTransferRequest treq = new CxTransferRequest(CxServer.UNIX);
treq.setFileName("FILE01");
treq.setTransferDirection('T');
treq.setPartnerName("BOUCLE");
treq.setLocalPassword("PSW");
treq.setPhysicalName("$TOM_DIR/out/file.txt");

// Submit the transfer
String[] requestNumber = srv.submitTransfer(treq);

// Resulting request number
System.out.println("request mumber="+requestNumber[0]);

...
```

This example submits the transfer of the file “file.txt”, to the partner “BOUCLE”, in accordance with the symbolic file “FILE01”. The resulting request number is received in requestNumber[0].

Notes:

- With Connect:Express Windows, more than one transfer can result from a single submission when transmitting a group of files. This is why the result of submitTransfer() is an array and not a simple string. In the case of Connect:Express Unix, only one request number is obtained.
- The « logical » errors in a request submission generate CxServer exceptions.

X509 certificates display

Access to the certificates is read only, by using one of the 2 following methods of the CxServer class:

- selectCertificates(CxCertificateFilter filter)
- getCertificate(CxCertificateFilter filter)

The CxCertficateFilter parameter defines the selection criteria of the certificate(s). This parameter takes different forms depending on the OS type of the Connect:Express server (Unix or Windows).

Connect:Express Unix

The certificates are imported into a private database with the $sterm tool. Each certificate is then referenced by a unique identifier (CertificateId) of at most 8 characters. The certificates can be personnal certificates or root certification authority certificates.
The selection parameters in CxCertificateFilter are then:

For selectCertificates (selection of a set of certificates):

- CertificateId
- CertificateType
- Subject
- Issuer

These elements can be simple patterns.

For getCertificate (detail of a certificate):

- CertificateId

Examples:

```
...
//Get a list of certificates
CxCertificateFilter filter = new CxCertificateFilter(CxServer.UNIX);
```

```
filter.setCertificateId("*");
filter.setSubject("*www.caexmpl.com*");
CxCertificatesListElement[] elm = srv.selectCertificates(filter);
...
```

```
...
//Get a list of certificates
CxCertificateFilter filter = new CxCertificateFilter(CxServer.UNIX);
filter.setCertificateId("CAEXMP*");
CxCertificatesListElement[] elm = srv.selectCertificates(filter);
...
```

```
...
//Get a certificate
CxCertificateFilter filter = new CxCertificateFilter(CxServer.UNIX);
filter.setCertificateId("CAEXMP1");
CxCertificates cert = srv.getCertificate(filter);

//Display the certificate as a PEM certificate (Base 64 encoded)
System.out.println(cert.getPEMCertificate()) ;

//Display the certificate characteristics
System.out.println(cert.getCertificateCharacteristics()) ;
...
```

Connect:Express Windows

The certificates are imported into the certificate stores of the Windows system by using the mmc (Microsoft Management Console).
The selection parametrs in CxCertificateFilter are in this case:

For selectCertificates (selection of a set of certificates):

- StoreLocation
- StoreName
- Subject (character string to search in the distinguished name of the subject)
- Issuer (character string to search in the distinguished name of the issuer)

StoreName, StoreLocation are required.

For getCertificate (detail of a certificate):

- StoreLocation
- StoreName
- Subject (Complete distinguished name of the subject or value of the common name of this DN)
- Issuer (An optionnal character string that must be found in the distinguished name of the issuer)

StoreName and StoreLocation et Subject are required.

StoreLocation can take one of the following values: « SYSTEM_STORE_LOCAL_MACHINE », « SYSTEM-STORE_CURRENT_USER » ou « SYSTEM_STORE_SERVICES ».
StoreName can take one of the following values: « My » (Personal), « Root » (Trusted root authorities), « CA » (Intermediate certification authorities) ou Trust (Enterprise trust).

Examples:

```
...
//Get a list of certificates
CxCertificateFilter filter = new CxCertificateFilter(CxServer.WINDOWS);
filter.setStoreLocation("SYSTEM_STORE_LOCAL_MACHINE");
filter.setStoreName("Root");
filter.setSubject("Microsoft");
filter.setIssuer("O=Microsoft Trust Network");
CxCertificatesListElement[] elm = srv.selectCertificates(filter);
...
```

```
...
//Get a certificate
CxCertificateFilter filter = new CxCertificateFilter(CxServer.WINDOWS);
filter.setStoreLocation("SYSTEM_STORE_LOCAL_MACHINE");
filter.setStoreName("Root");

//Value of the common name of the subject dn
filter.setSubject("Microsoft Root Certificate Authority");

CxCertificates cert = srv.getCertificates(filter);

//Display the certificate as a PEM certificate (Base 64 encoded)
System.out.println(cert.getPEMCertificate()) ;

//Display the certificate characteristics
System.out.println(cert.getCertificateCharacteristics()) ;
...
```

```
...
//Get a certificate
CxCertificateFilter filter = new CxCertificateFilter(CxServer.WINDOWS);
filter.setStoreLocation("SYSTEM_STORE_LOCAL_MACHINE");
filter.setStoreName("Root");
```

```
//Complete distinguished name of the subject in reverse order
filter.setSubject(
           "CN=Microsoft Root Certificate Authority, DC=micosoft, DC=com");

CxCertificates cert = srv.getCertificate(filter);

//Display the certificate as a PEM certificate (Base 64 encoded)
System.out.println(cert.getPEMCertificate()) ;

//Display the certificate characteristics
System.out.println(cert.getCertificateCharacteristics()) ;
...
```

This chapter gives complete examples of programs using the JAI with a Connect:Express Unix server. Each example receives as its two first parameters the address and port of the server.

Examples

Examples 1 to 4 and 6 to 8 can be used with any Connect:Express Unix, with no special configuration.
Example 5 needs that a partner BOUCLE and a symbolic file FILE01 be previously defined.
The source code examples below are provided in the file CXJAI_examples.jar.

To execute the examples, extract CXJAI_examples.jar and copy CXJAI.jar into the resulting bin directory. Then, start the examples with the scripts examplex.sh (bat) and the appropriate parameters.

For example:

```
./example1.sh 10.87.15.106 9000 (Unix)
example1.bat  10.87.15.106 9000 (Windows)
```

Where 10.87.15.105 and 9000 correspond to the JAI address and port of the Connect:Express server.

Example 1: Partners, Symbolic Files, Tables.

This program displays the partners, symbolic files and the session and presentation tables.

```
                                                                /*Example1 (1)*/

/*
 * Example1.java
 * This program displays
 * - the symbolic partners
 * - the symbolic files
 * - the session tables
 * - the presentation tables
 * of a Connect:Express Unix or Windows server
 *
 * The parameters are the IP address/host name and port of the server
 *
 */

import java.util.Date;
```

```
                                                                  /*Example1 (2)*/

import com.sterlingcommerce.cx.sdk.*;


public class Example1 {

  public Example1() {
  }

  /**
   * @param args
   * Lists partners,files,tables ...
   * param1 = server address
   * param2 = server port
   *
   */
  public static void main(String[] args) {
    CxServer srv = null;
    String OSType = null;

    Date startDate = new Date();
    if (args.length!=2) {
      usage();
      return;
    }
    int uu = 0;
    try {
      uu = Integer.parseInt(args[1].trim());
    } catch (NumberFormatException e) {
      System.out.println("Invalid port");
      return;
    }
    try {
      srv = connectToServer(args[0],uu);
      displayConnectionInfo(srv);
      displayPartners(srv);
      displayFiles(srv);
      displaySessions(srv);
      displayPresentations(srv);
      Date endDate = new Date();
      System.out.println("Elapsed time = "
                  +(endDate.getTime()-startDate.getTime())+" ms");

    } catch (CxServerException e) {
      //Depending on the server message id, processing can continue or not
      //See documentation on the MIDs (TRC) that can be returned by Connect:Express
      System.out.println("Message id = "+e.getMid());
      System.out.println("Message Text = "+e.getMessageText());
      displayErrors(e);
    } catch (CxConnectionException e) {
      //Reconnect ?
      displayErrors(e);
    } catch (CxLogonException e) {
      //Enter userid/password again ?
      displayErrors(e);
    } catch (CxInvalidArgumentException e) {
      //Fatal error
      displayErrors(e);
    } catch (Exception e) {
      //Fatal error
      e.printStackTrace();
    } finally {
```

```
                                                              /*Example1 (3)*/
    try {
      srv.disconnect();
    } catch (Exception e) {
    }

  }
}//End main

public static void displayErrors(CxException e) {
  System.out.println("Errors:");
  System.out.println("--------");
  Throwable t = e;
  while (t!=null) {
    System.out.println(t.getMessage());
    t = t.getCause();
  }
}//End displayErrors

public static CxServer connectToServer(String c_host,int c_port)
                                        throws CxException {
  char[] pwd = {'A','D','M','I','N'};

  CxServer srv = new CxServer(c_host,c_port,"ADMIN",pwd,"TCPIP",0,false);
  return srv;
}//End connectToServer

public static void displayConnectionInfo(CxServer srv)
                                  throws CxException {
  CxConnectionInformation cInfo = srv.getConnectionInfo();
  System.out.println("Address               = "+cInfo.getAddress());
  System.out.println("Port                  = "+cInfo.getPort());
  System.out.println("Userid                = "+cInfo.getUserid());
  System.out.println("NodeName              = "+cInfo.getNodeName());
  System.out.println("OSType                = "+cInfo.getOSType());
  System.out.println("NodeVersion           = "+cInfo.getNodeVersion());
  System.out.println("APKeySupport          = "+cInfo.getAPKeySupport());
}//End displayConnectionInfo

public static void displayPartners(CxServer srv)  throws CxException {
  String[] list = null;

  list = srv.listPartners();
  System.out.println("-----------------------------------------------------");
  System.out.println("SYMBOLIC PARTNERS");
  System.out.println("-----------------------------------------------------");
  for (int i=0;i<list.length;i++) {
    CxPartner partner = srv.getPartner((String)list[i]);
    displayPartnerDetail(srv,partner);
    System.out.println("-----------------------------------------");
  }
}//End displayPartners

public static void displayPartnerDetail(CxServer srv,CxPartner partner)
                                                    throws CxException {
  System.out.println("PartnerName        = "+partner.getPartnerName());
  System.out.println("PartnerPassword    = "+partner.getPartnerPassword());
  System.out.println("LocalName          = "+partner.getLocalName());
  System.out.println("LocalPassword      = "+partner.getLocalPassword());
  System.out.println("PartnerState       = "+partner.getPartnerState());
  System.out.println("TypeOfPartner      = "+partner.getTypeOfPartner());
```

```
                                                        /*Example1 (4)*/

  System.out.println("Protocol           = "+partner.getProtocol());
  System.out.println("MaxSession         = "+partner.getMaxSession());
  System.out.println("MaxSessionIn       = "+partner.getMaxSessionIn());
  System.out.println("MaxSessionOut      = "+partner.getMaxSessionOut());
  System.out.println("SessionTableId     = "+partner.getSessionTableId());
  System.out.println("TypeOfLink         = "+partner.getTypeOfLink());
  System.out.println("TcpipAddress       = "+partner.getTcpipAddress());
  System.out.println("TcpipPort          = "+partner.getTcpipPort());
  System.out.println("TcpipHostName      = "+partner.getTcpipHostName());
  System.out.println("X25LocalAddress    = "+partner.getX25LocalAddress());
  System.out.println("X25RemoteAddress   = "+partner.getX25RemoteAddress());
  System.out.println("X25LocalPort       = "+partner.getX25LocalPort());
  System.out.println("X25UserDataField   = "+partner.getX25UserDataField());
  System.out.println("X25Facilities      = "+partner.getX25Facilities());
  String OSType = srv.getConnectionInfo().getOSType();
  if (OSType.equals(CxServer.UNIX)==true) {
    System.out.println("FtpDefaultFile     = "+partner.getFtpDefaultFile());
    System.out.println("RetryNumber        = "+partner.getRetryNumber());
    System.out.println("SessionTimer       = "+partner.getSessionTimer());
    System.out.println("TransferTimer      = "+partner.getTransferTimer());
  } else if (OSType.equals(CxServer.WINDOWS)==true) {
    System.out.println("LocalNameType      = "+partner.getLocalNameType());
    System.out.println("PartnerComment     = "+partner.getPartnerComment());
    System.out.println("RestartUsed        = "+partner.isRestartUsed());
    System.out.println("SnaLuName          = "+partner.getSnaLuName());
    System.out.println("AppcModeName       = "+partner.getAppcModeName());
    System.out.println("AppcTpName         = "+partner.getAppcTpName());
  }
}//End displayPartnerDetail

public static void displayFiles(CxServer srv)  throws CxException {
  String[] list = null;

  list = srv.listFiles();
  System.out.println("-----------------------------------------------------");
  System.out.println("SYMBOLIC FILES");
  System.out.println("-----------------------------------------------------");
  for (int i=0;i<list.length;i++) {
    CxFile file = srv.getFile((String)list[i]);
    displayFileDetail(srv,file);
    System.out.println("-----------------------------------------");
  }
}//End displayFiles

public static void displayFileDetail(CxServer srv,CxFile file)
                                                    throws CxException {
  System.out.println("FileName           = "+file.getFileName());
  System.out.println("FileState          = "+file.getFileState());
  System.out.println("TypeOfAllocation   = "+file.getTypeOfAllocation());
  System.out.println("FileDirection      = "+file.getFileDirection());
  System.out.println("TypeOfFile         = "+file.getTypeOfFile());
  System.out.println("FileOpenOption     = "+file.getFileOpenOption());
  System.out.println("FileSender         = "+file.getFileSender());
  System.out.println("FileReceiver       = "+file.getFileReceiver());
  System.out.println("PresentationTableId= "
                                      +file.getPresentationTableId());
  System.out.println("LocalPhysicalName  = "+file.getLocalPhysicalName());
  System.out.println("FileRecordLength   = "+file.getFileRecordLength());
  System.out.println("StartTransmitExit  = "+file.getStartTransmitExit());
  System.out.println("EndTransmitExit    = "+file.getEndTransmitExit());
```

```
                                                    /*Example1 (5)*/

 System.out.println("StartReceiveExit   = "+file.getStartReceiveExit());
  System.out.println("EndReceiveExit     = "+file.getEndReceiveExit());
  System.out.println("StartTransmitCommand= "
                                          +file.getStartTransmitCommand());
  System.out.println("EndTransmitCommand = "
                                          +file.getEndTransmitCommand());
  System.out.println("StartReceiveCommand= "+file.getStartReceiveCommand());
  System.out.println("EndReceiveCommand      = "+file.getEndReceiveCommand());
  String OSType = srv.getConnectionInfo().getOSType();
  if (OSType.equals(CxServer.UNIX)==true) {
    System.out.println("Priority           = "+file.getPriority());
    System.out.println("RemotePhysicalName = "+file.getRemotePhysicalName());
    System.out.println("FtpOptions         = "+file.getFtpOptions());
    System.out.println("ParamFileUsed      = "+file.isParamFileUsed());
    System.out.println("SpaceAllocationUsed= "+file.isSpaceAllocationUsed());
    System.out.println("FtpStoreUniqueUsed = "+file.isFtpStoreUniqueUsed());
    System.out.println("FileAgentUsed      = "+file.isFileAgentUsed());
    System.out.println("TypeOfNotification = "+file.getTypeOfNotification());

  } else if (OSType.equals(CxServer.WINDOWS)==true) {
    System.out.println("FileComment        = "+file.getFileComment());
    System.out.println("ErrorCommand       = "+file.getErrorCommand());
    System.out.println("NotifyUsed         = "+file.isNotifyUsed());
    System.out.println("ClientToNotify     = "+file.getClientToNotify());
    System.out.println("Pi99OffsetT        = "+file.getPi99OffsetT());
    System.out.println("Pi99LengthT        = "+file.getPi99LengthT());
    System.out.println("Pi99ValueT         = "+file.getPi99ValueT());
    System.out.println("Pi99OffsetR        = "+file.getPi99OffsetR());
    System.out.println("Pi99LengthR        = "+file.getPi99LengthR());
    System.out.println("Pi99ValueR         = "+file.getPi99ValueR());
    System.out.println("FileLabel          = "+file.getFileLabel());
  }
}//End displayFileDetail

public static void displaySessions(CxServer srv)  throws CxException {
  String[] list = null;

  list = srv.listSessions();
  System.out.println("-----------------------------------------------------");
  System.out.println("SESSION TABLES");
  System.out.println("-----------------------------------------------------");
  for (int i=0;i<list.length;i++) {
    CxSession session = srv.getSession((String)list[i]);
    displaySessionDetail(srv,session);
    System.out.println("-------------------------------------------");
  }
}//End displaySessions

public static void displaySessionDetail(CxServer srv,CxSession ses)
                                                    throws CxException {
  System.out.println("SessionTableId          = "+ses.getSessionTableId());
  System.out.println("BaseMessageSize         = "+ses.getBaseMessageSize());
  System.out.println("BaseSynchronizationSize = "
                                  +sess.getBaseSynchronizationSize());
  System.out.println("CrcUsed                 = "+ses.isCrcUsed());
  String OSType = srv.getConnectionInfo().getOSType();
  if (OSType.equals(CxServer.UNIX)==true) {
    System.out.println("BaseWindowSize          = "+ses.getBaseWindowSize());
    System.out.println("ProtocolVersion         = "+ses.getProtocolVersion());
```

```
                                                        /*Example1 (6)*/

      System.out.println("RetryNumber             = "+ses.getRetryNumber());
    } else if (OSType.equals(CxServer.WINDOWS)==true) {
      System.out.println("SessionDirection        = "+ses.getSessionDirection());
      System.out.println("ResynchronizationNumber = "
                                              +ses.getResynchronizationNumber());
    }
  }//End displaySessionDetail

  public static void displayPresentations(CxServer srv) throws CxException {
    String[] list = null;

    list = srv.listPresentations();
    System.out.println("-----------------------------------------------------");
    System.out.println("PRESENTATION TABLES");
    System.out.println("-----------------------------------------------------");
    for (int i=0;i<list.length;i++) {
      CxPresentation presentation = srv.getPresentation((String)list[i]);
      displayPresentationDetail(srv,presentation);
      System.out.println("-----------------------------------------");
    }
  }//End displayPresentations

  public static void displayPresentationDetail(CxServer srv,CxPresentation pres)
                                                          throws CxException {
    System.out.println("PresentationTableId     = "+pres.getPresentationTableId());

    System.out.println("TypeOfCompression       = "+pres.getTypeOfCompression());
    System.out.println("MultiArticleUsed        = "+pres.isMultiArticleUsed());
    System.out.println("TranslationToEbcdic     = "+pres.getTranslationToEbcdic());
    String OSType = srv.getConnectionInfo().getOSType();
    if (OSType.equals(CxServer.WINDOWS)==true) {
      System.out.println("ConcatenationUsed       = "+pres.isConcatenationUsed());
      System.out.println("SegmentationUsed        = "+pres.isSegmentationUsed());
      System.out.println("TranslationUsed         = "+pres.isTranslationUsed());
      System.out.println("TranslationToAscii      = "+pres.getTranslationToAscii());
    }
  }//End displayPresentationDetail

  static void usage() {
    System.out.println("Usage: java Example1 <ip-address/host-name> <port>\n");
  }//End usage

  private CxServer srv = null;
}//End class Example1
```

Example 2: Statistics.

This program displays the statistics for the last x seconds. x is passed as the 3rd execution parameter of the program.

```
                                                        /*Example2 (1)*/


/*
 * Example2.java
 * This program displays the statistics of a Connect:Express Unix or Windows server
 * for the last seconds indicated as 3rd parameter
 *
 * The parameters are
 * - the IP address/host name of the server
 * - the port of the server
 * - a number of seconds
 *
 */



import java.util.Locale;

import com.sterlingcommerce.cx.sdk.*;


public class Example2 {


  public Example2() {
  }


  /**
   * @param args
   * Displays statistics
   * param1 = address

   * param2 = port
   * param3 = number of seconds
   */
  public static void main(String[] args) {
    CxServer srv = null;

    if (args.length!=3) {
      System.out.println("Invalid number of parameters");
      usage();
    }

    int uu = 0;
    try {
      uu = Integer.parseInt(args[1].trim());
    } catch (NumberFormatException e) {
      System.out.println("Invalid port");
      return;
    }
```

```
                                                              /*Example2 (2)*/
   int vv = 0;
    try {
    vv = Integer.parseInt(args[2]);
    } catch (NumberFormatException e) {
    System.out.println("The 3rd argument must be a number of seconds");
    return;
    }

    try {
      srv = connectToServer(args[0],uu);
      //srv.setLimit(1000);
      displayStatistics(srv,vv);
    } catch (CxServerException e) {
      //Depending on the server message id, processing can continue or not
            //See  documentation  on  the  MIDs  (TRC)  that  can  be  returned  by
Connect:Express
      System.out.println("Message id = "+e.getMid());
      System.out.println("Message Text = "+e.getMessageText());
      displayErrors(e);
    } catch (CxConnectionException e) {
      //Reconnect ?
      displayErrors(e);
    } catch (CxLogonException e) {
      //Enter userid/password again ?
      displayErrors(e);
    } catch (CxInvalidArgumentException e) {
      //Fatal error
      displayErrors(e);
    } catch (Exception e) {
      //Fatal error
      e.printStackTrace();
    } finally {
      try {
        //Disconnect
        srv.disconnect();
        } catch (Exception e) {
      }
    }

  }//End main

  public static void displayErrors(CxException e) {
    System.out.println("Errors:");
    System.out.println("--------");
    Throwable t = e;
    while (t!=null) {
      System.out.println(t.getMessage());
      t = t.getCause();
    }

  }//End displayErrors

  public static CxServer connectToServer(String c_host,int c_port)
                                                                          throws
CxException {
    char[] pwd = {'A','D','M','I','N'};

    CxServer srv = new CxServer(c_host,c_port,"ADMIN",pwd,"TCPIP",0,false);
    return srv;
  }//End connectToServer
```

```
    public  static  void  displayStatistics(CxServer  srv,int  seconds)  throws
CxException {
   CxStatistics stat;
```

```
                                                        /*Example2 (3)*/


   CxEnumeration senum = srv.selectStatistics(seconds);
   System.out.println("-----------------------------------------------------");
   System.out.println("STATISTICS");
   System.out.println("-----------------------------------------------------");
   while (senum.hasMoreElements()==true) {
     stat = (CxStatistics)senum.nextElement();
     String[][] lvp = stat.getLabelValuePairs(Locale.FRANCE);
     System.out.println("-----------------------------------------");
     for (int i=0;i<lvp.length;i++) {
       System.out.println(lvp[i][0]+" = "+lvp[i][1]);
     }
     String trc = stat.getTrc();
     if (trc.equals("")==false) {
       if (trc.equals("0000")==false) {
         System.out.println("TRC explanation = "+srv.getTrcMessage(trc));
       }
     }
     String prc = stat.getPrc();
     if (prc.equals("")==false) {
       if (prc.substring(1).equals("0000")==false) {
         System.out.println("PRC explanation = "+srv.getPrcMessage(prc));
     }
   }//End while
   System.out.println("Returning from displayStatistics");

 }//End displayStatistics

 static void usage() {
   System.out.println(
   "Display statistics\n"+
   "Usage: java Example2 <ip-address> <port> <number-of-seconds>\n");
   return;
 }//End usage

 private CxServer srv = null;
}//End class Example2
```

Example 3: Static Configuration of the Server.

This program displays the static configuration of the server as well as the elements of the asset protection key (components, expiration dates, …).
The source code file Example3.java is provided in the file CXJAI_examples.jar.

Example 4: Journal.

This program displays the content of the journal for the transfers that took place in the last x seconds.
x is passed in the 3rd execution parameter.

The source code file Example4.java is provided in the file CXJAI_examples.jar.

Example 5: Transfer Request Submission.

This program submits a loop transfer request to a Connect:Express Unix. The use of this program assumes that a partner BOUCLE and a symbolic file FILE01 have been previously created in the monitor.
The source code file Example5.java is provided in the file CXJAI_examples.jar.

Example 6: Unix server environment variables.

This program gets the value of the $TOM_DIR and $PATH variables from the server.
It, also, shows environment variables replacement in a string.
The source code file Example6.java is provided in the file CXJAI_examples.jar.

Example 7: SSL client or server parameters.

This program displays the client and server SSL parameters.
Each set of SSL parameters is identified when created in Connect :Express by an identifier of at most 8 characters.
Contrary to the SSL client parameters, the SSL server parameters cannot be created, updated or deleted with the API.
These operations require a stop/restart of the moniteur which cannot be done remotely.
The source code file Example7.java is provided in the file CXJAI_examples.jar.

Exemple 8: X509 Certificates.

This program displays the X509 certificates which can be used by Connect:Express for doing authentication during SSL transfers.
The source code file Example8.java is provided in the file CXJAI_examples.jar.

Connect:Express Server Configuration

The following paragraph describes the configuration parameters of the Connect:Express monitors that are related to the JAI.

Connect:Express Unix.

To use the JAI, add a line APPORT=<listening-port> or APPORT=<local-ip-address>:<listening-port > in the file $TOM_DIR/config/sysin.

For example:

```
APPORT=9000
```

In order to use the statistics functionnality use ISSTAT parameter, in the file $TOM_DIR/config/sysin.

```
ISSTAT=1
```

Connect:Express Windows.

The listening port number of the JAI is fixed to the default value 7000 during the installation of the monitor. Every client application, for example the local graphical interface, must use this port.

To modify this port number, start the local graphical interface and open the dialog box Configuration / Network / TCPIP.
To enable the statistics, use the dialog box Configuration / Files.

Connect:Express OS/390.

Appendix. Connect:Express Data

The tables below indicate, depending on the operating system, the availability and the size of the different data fields of the Connect:Express servers.
The column “Key” contains the keywords used to identify the data in the statistics records.

Symbolic Partner Data

Key	Field	Lg max	Description	Win	Unix	OS / 390
APPD	AppcDisconnectUsed	1	Disconnect Option for LU6.2 is used, Y or N			1
APPM	AppcModeName	8	Remote LU6.2 mode name	8		8
APPT	AppcTpName	64	Remote LU6.2 transaction program	64		8
FTAC	FtpAccessRight	8	Access rights definition name			8
FTDF	FtpDefaultFile	8	Default file name for FTP		8	8
FTPV	FtpPasvUsed	1	Pasv is active, Y or N			1
LNKL	LinkList	3	If link type = M, list of available links			3
LNAM	LocalName	8	Alias name of the local Connect:Express	8	8	8
LNTP	LocalNameType	1	Dynamic local ID is used, Y or N	1		
LPSW	LocalPassword	8	Alias password of the local Connect:Express	8	8	8
MSES	MaxSession	3	Maximum simultaneous sessions	3	2	3
MSIN	MaxSessionIn	2	Maximum simultaneous sessions Inbound	3	2	3
MSOU	MaxSessionOut	2	Maximum simultaneous sessions Outbound	3	2	3
ODNM	OdetteName	25	Odette identification			25
PCLA	PartnerClass	1	Class to use for inbound transfers			1
PTXT	PartnerComment	80	Description of the symbolic partner definition	80		70
PNAM	PartnerName	8	Symbolic Partner name	8	8	8
PPSW	PartnerPassword	8	Symbolic Partner password	8	8	8
PSTA	PartnerState	1	Symbolic Partner status, E=Enable, H=Disabled	1	1	1
PROT	Protocol	1	Transfer protocol Windows: D=PesitD, E=PesitE, O=Oftp, 3=Etebac3, F=ftp Unix: 0: PeSIT, 1: FTP, 2: Etebac3 (See ProtocolVersion in session table to get the version level of PeSIT, D ou E)	1	1	1
RACG	RacfGroup	8	Security racf group			8
RACU	RacfUser	8	Racf user			8
RCSD	RemoteClientSubjectDn	256	Criteria for remote client subject DN control	256		
RCRD	RemoteClientRootDn	256	Criteria for remote client root DN control	256		
RETO	RestartUsed	1	Automatic restart is used, Y or N	1		1
RETN	RetryNumber	2	Maximum number of retries for this partner		2	
RSSD	RemoteServerSubjectDn	256	Criteria for remote server subject DN control	256		
RSRD	RemoteServerRootDn	256	Criteria for remote server root DN control	256		
STAB	SessionTableId	50	Name of the session table or identification number	50	1	1
STMR	SessionTimer	2	Session timer		2	
SLID	SldEntryId	1	SLD entry identification			1
SNAL	SnaLuName	8	Remote SNA address	8		8
TCPA	TcpIpAddress	15	Remote TCP/IP address	15	15	15
TCPH	TcpIpHostName	127	Remote TCP/IP host name	127	32	32
TCPP	TcpIpPort	5	Remote listening TCP/IP port	5	5	5

TTMR	TransferTimer	2	Transfer timer		2	
TYPL	TypeOfLink	1	Type of link, 0 = LU 6.2, 1 = X25, 2 = TCP/IP, M=mixed	1	1	1
TYPP	TypeOfPartner	1	Type of Partner, Other or Tom	1	1	1
X25F	X25Facilities	32	Remote X25 address, facilities	32	16	12
X25L	X25LocalAddress	15	Local X25 address	15	15	15
X25P	X25Localport	2	Local device or MCH identification	2	1	1
X25A	X25RemoteAddress	15	Remote X25 address	15	15	15
X25T	X25Taxation	1	Remote X25 address, Tax rule			1
X25U	X25UserDataField	16	Remote X25 address, user data field	8	8	16
X25G	X25UserGroup	2	X25 Remote X25 address, user Group			2

Symbolic File Data

Key	Field	Lg max	Description	Win	Unix	OS / 390
CLIN	ClientToNotify	8	Name of the client to notify	8		
DIRB	DirectoryBlock	3	Number of directory blocks (file = P or PU) (MVS dcb)			3
DISP	Disposition	3	Allocation disposition (SHR, NEW, OLD) (MVS dcb)			3
ERCD	EndReceiveCommand	127	User command called at end of reception	127	12	32
EREX	EndReceiveExit	127	User exit called at end of reception	127	12	8
ETCD	EndTransmitCommand	127	User command called at end of transmission	127	12	32
ETEX	EndTransmitExit	127	User exit called at end of transmission	127	12	8
ERRC	ErrorCommand	127	User command called when an error occurs	127		
FLAO	FileAgentUsed	1	Interconnected File Agent is active, Y or N		1	
FBLK	FileBlockSize	5	File Physical block size (MVS dcb)			5
FTXT	FileComment	80	Description of the symbolic file definition	80		79
FDIR	FileDirection	1	Transfer direction authorized, T = transmit, R = receive, * = both	1	1	1
FLAB	FileLabel	80	File user identification	80		
FNAM	FileName	8	Symbolic file name	8	8	8
FOPO	FileOpenOption	1	Allocation rule, N = New file, R = Replace, O = Append	1	1	2
FRCV	FileReceiver	8	Partner, or list of partners, authorized to receive the file	8	8	8
FRFM	FileRecordFormat	3	Local record format (MVS DCB)			3
FRLG	FileRecordLength	5	Local record length	5	5	5
FRET	FileRetention	8	Local expiration or retention date (MVS dcb)			8
FSND	FileSender	8	Partner, or list of partners, authorized to send the file	8	8	8
FSTA	FileState	1	Symbolic file status, E=Enable, H=Disabled	1	1	1
FUNM	FileUnitName	8	Local unit name for allocation (MVS dcb)			8
FTOP	FtpOptions	4	FTP file transfer options (type/structure/mode)		4	4
FTSU	FtpStoreUniqueUsed	1	FTP store Unique is used, Y or N		1	1
GDGN	GdgNumber	3	Gdg file generation number (+xx or -xx)			3
MEMB	JobMember	8	Unload/Reload selection member (file = PU, SU, UU)			8
LPHN	LocalPhysicalName	127	Local file physical name	127	44	44
NFYO	NotifyUsed	1	Notification is used, Y or N	1		
OPHN	OriginPhysicalName	44	File name proposed to remote as their remote data set name (or Pi99)			44
PARM	ParamFileUsed	1	Parameter card file is used, Y or N		1	
RP99	Pi99ValueR	254	Value for Pi99 (reception)	254		
R99O	Pi99OffsetR	3	Offset of the preceding value in the Pi99 (reception)	3		
R99L	Pi99LengthR	3	Length of the preceding value in the Pi99 (reception)	3		
SP99	Pi99ValueT	254	Value for Pi99 (transmission)	254		
S99O	Pi99OffsetT	3	Offset of the preceding value in the Pi99 (transmission)	3		
S99L	Pi99LengthT	3	Length of the preceding value in the Pi99 (transmission)	3		
PTAB	PresentationTableId	50	Name or identification number of the presentation table used	50	1	2

PRIO	Priority	1	Transfer priority, 0 = Urgent, 1 = Normal , 2 = slow		1	1
RPHN	RemotePhysicalName	44	Remote file physical name		44	
SECU	Security	2	Security table identification number			2
SPAO	SpaceAllocationUsed	1	Space reservation Y/N		1	
SPA1	SpacePrimary	4	Allocation primary space (MVS dcb)			4
SPA2	SpaceSecondary	4	Allocation secondary space (MVS dcb)			4
SPAT	SpaceType	3	Allocation space type (CYL, TRK, …) (MVS dcb)			3
SRCD	StartReceiveCommand	127	User command called at beginning of reception	127	12	32
SREX	StartReceiveExit	127	User exit called at beginning of reception	127	12	8
STCD	StartTransmitCommand	127	User command called at beginning of transmission	127	12	32
STEX	StartTransmitExit	127	User exit called at beginning of transmission	127	12	8
TYPA	TypeOfAllocation	1	Type of allocation, F = Fixed, D=Dynamic	1	1	1
TYPF	TypeOfFile	2	Type of file, TF = Text fixed, TV = text variable, XF = Unix text fixed on Windows, XV = Unix text variable on Windows, UF = Unix fixed, UV = Unix Variable, BF = binary fixed, BU = Binary undefined, S = Sequential, V = VSAM, P = PDS, PE = PDSE, PU = PDS unload, VU = VSAM unload, SU = SYSOUT unload, UU = User unload	2	2	2
TYPN	TypeOfNotification	1	Type of Notification: 1 character (‘0’ to ‘7’). ‘0’: No notification. ‘1’: Notification at the beginning of the transfer. ‘2’: Notification at the end of the transfer. ‘4’: Notification if transfer error. Other possibilities are combinations with inclusive « OR » of these values. For example: ‘6’ = ‘2’ OR ‘4’ for a notification at the end of transfer or in case of transfer error. **Windows**: This flag is used for HTTP notification only. **Unix:** This flag is used for HTTP notification or standard notification depending on the values of the keywords HTTPNF and NOTIFY in the sysin configuration file.	1	1	
VOLN	VolumeName	30	List of 1 to 5 (6 characters) Volume(s) name(s)			30

Session Table Data

Key	Field	Lg max	Description	Win	Unix	OS / 390
BMSG	BaseMessageSize	5	Network message size (negociation)	4	5	5
BSNC	BaseSynchronizationSize	5	Synchronization Kbytes size (negociation)	2	2	5
BWIN	BaseWindowSize	2	Synchronization window size (negociation)		2	2
CMPO	CompressionUsed	1	Odette Compression used, Y or N			1
CRCO	CrcUsed	1	CRC used , Y or N	1	1	1
PVER	ProtocolVersion	2	Protocol version		1	2
RSYN	ResynchronizationNumber	2	Number of resynchronization for the request	2		
RSYO	ResynchronizationUsed	1	Resynchronization is used, Y or N	1		1
RETN	RetryNumber	2	Number of retries for the request		2	
SDIR	SessionDirection	1	Transfer direction authorized, T = transmit, R = receive, * = both	1		1
STAB	SessionTableId	50	Name of the session table or identification number	50	1	1
SMSG	SnaMessageSize	5	Specific SNA message size (negociation)			5
SSNC	SnaSynchronizationSize	5	Specific SNA Synchronization Kbytes size (negociation)			5
SWIN	SnaWindowSize	2	Specific SNA Synchronization window size (negociation)			2
TMSG	TcpMessageSize	5	Specific TCP/IP message size (negociation)			5
TSNC	TcpSynchronizationSize	5	Specific TCP/IP Synchronization Kbytes size (negociation)			5
TWIN	TcpWindowSize	2	Specific TCP/IP Synchronization window size (negociation)			2

XMSG	X25MessageSize	5	Specific X25 message size (negociation)			5
XSNC	X25SynchronizationSize	5	Specific X25 Synchronization Kbytes size (negociation)			5
XWIN	X25WindowSize	2	Specific X25 Synchronization window size (negociation)			2

Presentation Table Data

Key	Field	Lg max	Description	Win	Unix	Mvs
CMPO	CompressionUsed	1	Compression used, Y or N			1
CONC	ConcatenationUsed	1	PeSIT Fpdu Data option	1		
HPFO	HighPerformanceUsed	1	Bulk transfer flag			3
IBUF	InternalBuffer	3	Size of internal buffer			3
IOEX	IoUserExit	8	Name of the I/O user exit			8
MULT	MultiArticleUsed	1	Multiarticle is used in PeSIT Fpdu Data, Y or N	1	1	
PTAB	PresentationTableId	50	Name or identification number of the presentation table used	50	1	3
SEGM	SegmentationUsed	1	Segmentation is used in PeSIT Fpdu Data, Y or N	1		1
TREA	TranslationToAscii	127	Ebcdic to Ascii Translation table identification number or name	127		
TRAE	TranslationToEbcdic	127	Ascii to Ebcdic translation table identification number or name	127	1	
TRAO	TranslationUsed	1	Translation is used, Y or N	1		
TYPC	TypeOfCompression	2	Compression, Horizontal,Vertical,Mixed or pres.table identification number	1	1	1
TYPD	TypeOfData	1	Type of data, A = Ascii, E = Ebcdic, B = Binary			1
UEX1	UserExitOne	8	Name of the first user exit			8
UEX3	UserExitThree	8	Name of the third user exit			8
UEX2	UserExitTwo	8	Name of the second user exit			8

Request Submission Parameters

Key	Field	Lg max	Description	Win	Unix	Mvs
AHGP	AdHocGroup	8	AdHoc remote user racf group			8
AHN2	AdHocNewConfirm	8	AdHoc remote user New Password confirmation			8
AHN1	AdHocNewPassword	8	AdHoc remote user New Password			8
AHPW	AdHocPassword	8	AdHoc remote user Password	8		8
AHUS	AdHocUser	8	AdHoc remote user ID	8		8
CLIN	ClientToNotify	256	Name of the client to notify	256		
DATE	DateOfExecution	18	Date Time when the request must be scheduled	18	18	18
FLAO	FileAgentUsed	1	Interconnected File Agent is active, Y or N		1	
FJAI	FileJAI	88	User description of the transfer – Etebac3 card – Odette transfer	80	88	82
FLAB	FileLabel	80	File user identification	80	80	
FNAM	FileName	8	Symbolic file name	8	8	8
FRFM	FileRecordFormat	3	AdHocLocal record format (MVS DCB)			3

FRLG	FileRecordLength	5	Local record length		5	5
FTOP	FtpOptions	4	FTP file transfer options (type/structure/mode)		3	4
FTSU	FtpStoreUniqueUsed	1	FTP store Unique is used, Y or N		1	1
MEMB	JobMember	8	Unload/Reload selection member (file = PU, SU, UU)			8
JOBN	JobName	8	Job name of the sysout to transfer (SYSOUT)			8
LSP1	LoaclSpacePrimary	4	AdHoc local allocation primary space type (MVS dcb)			4
LBLK	LocalBlockSize	5	AdHoc local physical block size (MVS dcb)			5
LDIR	LocalDirectoryBlock	3	AdHoc local number of directory blocks (file = P or PU)			3
LDS1	LocalDisposition1	3	AdHoc local allocation disposition (SHR, NEW, OLD)			3
LDS2	LocalDisposition2	3	AdHoc local allocation disposition (KEEP,CTLG)			3
LDS3	LocalDisposition3	3	AdHoc local Allocation disposition (KEEP,CTLG)			3
LNAM	LocalName	8	Alias name of the local Connect:Express	8	8	8
LPSW	LocalPassword	8	Alias password of the local Connect:Express	8	8	8
LPHN	LocalPhysicalName	127	Local file physical name	127	44	44
LRET	LocalRetentionDate	8	AdHoc local expiration or retention date (MVS dcb)			8
LSP2	LocalSpaceSecondary	4	AdHoc local allocation secondary space type (MVS dcb)			4
LSPT	LocalSpaceType	3	AdHoc local allocation space type (CYL, TRK, …) (MVS dcb)			3
LTAP	LocalTapeDefinition	7	AdHoc local sequence number (4) and Tape label (3)			7
LUNT	LocalUnitName	8	AdHoc local unit name for allocation (MVS dcb)			8
LVOL	LocalVolumeName	30	AdHoc list of 1 to 5 (6 characters) local volume(s) name(s)			30
MEML	MemberList	256	List of 1 to 32 (8 characters) members (SELECTION)			256
NFYO	NotifyUsed	1	Notification is used, Y or N	1		
OPHN	OriginPhysicalName	44	File name proposed to remote as their remote data set name (or Pi99)	44		44
PNAM	PartnerName	8	Symbolic Partner name	8	8	8
PI99	Pi99Value	254	Pi99 to send	254	254	
P99O	Pi99Offset	3	Offset in the Pi99		3	
P99L	Pi99Length	3	Length in the Pi99		3	
PRIO	Priority	1	Transfer priority, 0 = Urgent, 1 = Normal , 2 = slow	1	1	1
RACG	RacfGroup	8	Security racf group			8
RSP2	RemoteSpaceSecondary	4	AdHoc remote allocation secondary space (MVS dcb)			4
RBLK	RemoteBlockSize	5	AdHoc remote physical block size (MVS dcb)			5
RDIR	RemoteDirectoryBlock	2	AdHoc remote number of directory blocks (file = P or PU) (MVS dcb)			2
RDS3	RemoteDisposition3	3	AdHoc remote allocation disposition (KEEP,CTLG) (MVS dcb)			3
RDS1	RemotelDisposition1	3	AdHoc remote allocation disposition (SHR, NEW, OLD) (MVS dcb)			3
RDS2	RemotelDisposition2	3	AdHoc remote allocation disposition (KEEP,CTLG) (MVS dcb)			3
RPHN	RemotePhysicalName	44	Adhoc remote file physical name	44	44	44
RRFM	RemoteRecordFormat	3	AdHoc remote Record format (MVS dcb)			3
RREC	RemoteRecordLength	5	AdHoc remote record length (MVS dcb)			5
RRET	RemoteRetentionDate	8	AdHoc remote expiration or retention date (MVS dcb)			8
RSP1	RemoteSpacePrimary	4	AdHoc remote allocation primary space (MVS dcb)			4
RSPT	RemoteSpaceType	3	AdHoc allocation space type (CYL, TRK, …) (MVS dcb)			3
RTAP	RemoteTapeDefinition	7	AdHoc remote sequence number (4) and tape label (3)			7
RUNT	RemoteUnitName	8	AdHoc remote unit name for remote allocation (MVS dcb)			8
RVOL	RemoteVolumeName	40	AdHoc list of 1 to 5 (6 characters) remote volume name(s) (MVS dcb)			40
RCLA	RequestClass	1	APM class where to execute the request			1
RMOD	RequestMode	1	Request scheduling mode, I = Immediat, D = Differed			1
REQR	Requestor	8	The name of the entity (user, job ..) that submitted the request	8		
SYSN	SysoutNumber	8	Sysout identification number to transfer (SYSOUT transfer)			8
PRMF	SysprmFile	44	Unload/Reload selection member directory (SELECTION)			44
TDST	TransferDestination	8	The entity that is processing the transfer request	8	8	8
TDIR	TransferDirection	1	Transmission or Reception	1	1	1
TORG	TransferOrigin	8	The entity that is requesting the transfer	8	8	8
TRCV	TransferReceiver	24	The entity that is processing the file after receiving it	8	24	
TSND	TransferSender	24	The entity that is processing the file before sending it	8	24	
TYPF	TypeOfFile	2	Type of file, TF = Text fixed, TV = text variable, UF = Unix fixed, UV		2	

			= Unix Variable, BF = binary fixed, BU = Binary undefined, S = Sequential, V = VSAM, P = PDS, PE = PDSE, PU = PDS unload, VU = VSAM unload, SU = SYSOUT unload, UU = User unload			
TYPL	TypeOfLink	1	Type of link, 0 = LU 6.2, 1 = X25, 2 = TCP/IP, M=mixed	1	1	1
TYPN	TypeOfNotification	1	Type of Notification: 1 character (‘0’ to ‘7’). ‘0’: No notification. ‘1’: Notification at the beginning of the transfer. ‘2’: Notification at the end of the transfer. ‘4’: Notification if transfer error. Other possibilities are combinations with inclusive « OR » of these values. For example: ‘6’ = ‘2’ OR ‘4’ for a notification at the end of transfer or in case of transfer error. **Windows**: This flag is used for HTTP notification only. **Unix:** This flag is used for HTTP notification or standard notification depending on the values of the keywords HTTPNF and NOTIFY in the sysin configuration file.	1	1	
TYPR	TypeOfRequest	1	Type of request, N = Normal, I = Inquiry, H = Hold	1	1	1
REQU	UserRequestId	16	Identification of the request given by the user	16		

Journal Data

Key	Field	Lg max	Description	Win	Unix	OS / 390
APPM	AppcModeName	8	Remote LU6.2 mode name	8		8
APPT	AppcTpName	64	Remote LU6.2 transaction program	64		8
BDEB	BitDebit	8	Number of bits per second			8
CDEB	CharacterDebit	8	Number of characters per second			8
CLIN	ClientToNotify	8	Name of the client to notify			x
CRCO	CrcOption	1	CRC used , Y or N	1	1	
ERC	CtreeReturnCode	4	Return code from ctree access	4		
DATE	DateOfExecution	18	Date when the request is accepted by Connect:Express	18	18	18
REQX	ExternalRequestNumber	8	Request number on the remote side	8	8	8
FLAO	FileAgentUsed	1	Interconnected File Agent is active, Y or N		1	
FJAI	FileJAI	88	User description of the transfer – Etebac3 card – Odette transfer		88	82
FBYT	FileBytes	12	Number of bytes of the file	12		8
FLAB	FileLabel	80	File user identification	80	80	80
FNAM	FileName	8	Symbolic file name	8	8	8
FNRD	FileNumberOfRecords	12	Number of records sent/received	12	12	8
FOPO	FileOpenOption	1	Allocation rule, N = New file, R = Replace, O = Append	1	1	1
FORG	FileOrganization	1	File organization, S = Sequential, I = Indexed, R = Relative	1	1	1
FRLG	FileRecordLength	5	Local record length	5	5	5
FSIZ	FileSize	8	Size announced by the sender			8
FTOP	FtpOptions	3	FTP file transfer options (type/structure/mode)		3	
FTSU	FtpStoreUniqueUsed	1	FTP store Unique is used, Y or N		1	
JNDA	JournalRecordDate	18	Date when the journal record is written by Connect:Express	18		18
LNAM	LocalName	8	Alias name of the local Connect:Express	8	8	8
LPHN	LocalPhysicalName	127	Local file physical name	127	44	44
MRET	MaxRetries	2	Maximum number of retries for this partner		2	
MULT	MultiArticleUsed	1	Multiarticle is used in PeSIT Fpdu Data, Y or N		1	1
NBYT	NetworkBytes	12	Number of bytes transferred	12	12	8
NMGS	NetworkMessageSize	5	Network message size	4	5	5
NRC	Nrc	4	Network Return code	4	4	6
OPHN	OriginPhysicalName	44	File name proposed to remote as their remote data set name (or Pi99)		44	

PNAM	PartnerName	8	Symbolic Partner name	8	8	8
PPSW	PartnerPassword	8	Symbolic Partner password		8	
PRC	Prc	3	Protocol return code	4	4	4
PTAB	PresentationTableId	2	Identification number of the presentation table used			2
PRIO	Priority	1	Transfer priority, 0 = Urgent, 1 = Normal , 2 = slow	1	1	1
PRID	ProcessId	12	Identifiation of the process that executed the request (PID, APMEFF ..)		12	5
PROT	Protocol	1	Transfer protocol, D=PesitD, E=PesitE, O=Oftp, 3=Etebac3, F=ftp	1	1	1
REQP	PurgedByMonitor	1	Request is purged, Y or N	1		
COMP	RealCompression	4	Compression performed	2		4
RPHN	RemotePhysicalName	44	Remote file physical name	44	44	
RCLA	RequestClass	1	APM class where the request has been executed			1
RELA	RequestElapse	8	Transfer elapse			8
REQN	RequestNumber	12	Request number given by Connect:Express	12	8	8
REQR	Requestor	8	The name of the entity (user, job ..) that submitted the request	8	8	8
RSYN	ResynchronizationNumber	3	Number of resynchronization for the request	3		3
RETN	RetryNumber	2	Number of retries for the request	2	2	2
SRCT	SendReceiveCount	12	Number of network send receive	12		8
SERV	ServiceType	1	Type of transfer request, AdHoc or Normal			1
STAB	SessionTableId	2	Indentification number of the session table used			2
STMR	SessionTimer	2	Session timer		2	
SNAL	SnaLuName	8	Remote SNA address	8		8
SNA1	SnaRc1	4	Primary SNA return code	4		
SNA2	SnaRc2	8	Secondary SNA return code	8		
SPAO	SpaceAllocationUsed	1	Space reservation Y/N		1	
SRC	Src	8	System Return code	8	8	4
SRC2	Src2	4	Complementary System Return code			
RSTA	Status	1	Transfer status, E = Ended / S = Started / I = Interrupted / W = Waiting	1	1	1
SYNC	SynchronizationInterval	2	Size of synchronization interval Kbytes	2		2
WIND	SynchronizationWindow	2	Number of synchronization intervals	2		2
TCPA	TcpIpAddress	15	Remote TCP/IP address	15	15	15
TCPH	TcpIpHostName	127	Remote TCP/IP host name	127	32	15
TCPP	TcpIpPort	5	Remote listening TCP/IP port	5	5	5
TCPC	TcpipRc	4	TCP/IP return code	4	4	
TBDA	TransferBeginningDate	18	Beginning of transfer date and time	18	18	18
TDST	TransferDestination	8	The entity that is processing the transfer request	8	8	8
TDIR	TransferDirection	1	Transmission or Reception	1	1	1
TEDA	TransferEndDate	18	End of transfer date and time	18	18	18
TIDT	TransferIdent	8	Transfer ident exchanged with the partner	6	6	8
TORG	TransferOrigin	8	The entity that is requesting the transfer	8	8	8
TRCV	TransferReceiver	24	The entity that is processing the file after receiving it	8	24	24
TSND	TransferSender	24	The entity that is processing the file before sending it	8	24	24
TTMR	TransferTimer	2	Transfer timer		2	
TRTN	TranslationTableNumber	1	Translation table identification number		1	2
TRC	Trc	4	Connect:Express Return code	4	4	4
TYPA	TypeOfAllocation	1	Type of allocation, F = Fixed, D=Dynamic	1		
TYPC	TypeOfCompression	1	Compression, Horizontal,Vertical,Mixed or presentation table identification	1	1	1
TYPD	TypeOfData	1	Type of data, A = Ascii, E = Ebcdic, B = Binary	1	1	1
TYPF	TypeOfFile	2	Type of file, TF = Text fixed, TV = text variable, UF = Unix fixed, UV = Unix Variable, BF = binary fixed, BU = Binary undefined, S = Sequential, V = VSAM, P = PDS, PE = PDSE, PU = PDS unload, VU = VSAM unload, SU = SYSOUT unload, UU = User unload	2	2	2
TYPL	TypeOfLink	1	Type of link, 0 = LU 6.2, 1 = X25, 2 = TCP/IP, M=mixed	1	1	1
TYPN	TypeOfNotification	1	Type of Notification: 1 character (‘0’ to ‘7’). ‘0’: No notification. ‘1’: Notification at the beginning of the transfer. ‘2’: Notification at the end of the transfer. ‘4’: Notification if transfer error. Other possibilities are combinations with inclusive « OR » of these values. For example: ‘6’ =	1	1	

			'2' OR '4' for a notification at the end of transfer or in case of transfer error. **Windows**: This flag is used for HTTP notification only. **Unix:** This flag is used for HTTP notification or standard notification depending on the values of the keywords HTTPNF and NOTIFY in the sysin configuration file.			
TYPP	TypeOfPartner	1	Type of Partner, Other or Tom	1	1	1
TYPR	TypeOfRequest	1	Type of request, N = Normal, I = Inquiry, H = Hold	1	1	1
TYPU	TypeOfUser	1	Type of user, I = Internal, E = External	1	1	1
USDR	UserDataReceived	254	User information received with the file	254	254	44
USDS	UserDataSent	254	User information sent with the file	254	254	44
REQU	UserRequestID	16	Identification of the request given by the user	16		
X25C	X25Cause	2	X25 Cause	2	2	2
X25D	X25Diagnostic	2	X25 Diagnostic	2	2	
X25F	X25Facilities	32	Remote X25 address, facilities	32	32	12
X25L	X25LocalAddress	15	Local X25 address	15	15	15
X25P	X25Localport	1	Local device or MCH identification	2	1	1
X25R	X25Rc	4	X25 Return code	4	4	
X25A	X25RemoteAddress	15	Remote X25 address	15	15	15
X25U	X25UserDataField	16	Remote X25 address, user data field	8	8	16
XLOV	XLocalPhysicalNameOvf	1	Set to true if overflow while setting XlocalPhysicalName value		8	
XLPH	XLocalPhysicalName	16	Equivalent to LocalPhysicalName with environment variables replaced.		512	

Active Transfer Data

Key	Field	Lg max	Description	Win	Unix	OS / 390
APPM	AppcModeName	8	Remote LU6.2 mode name			8
APPT	AppcTpName	64	Remote LU6.2 transaction program			8
BDEB	BitDebit	12	Number of bits per second			12
CDEB	CharacterDebit	12	Number of characters per second			12
CLIN	ClientToNotify	8	Name of the client to notify			
CRCO	CRCOption	1	CRC used , Y or N			1
ERC	CtreeReturnCode	4	Return code from ctree access	4		
REQX	ExternalRequestNumber	8	Request number on the remote side			8
FLAO	FileAgentUsed	1	Interconnected File Agent is active, Y or N			
FJAI	FileJAI	82	User description of the transfer – Etebac3 card – Odette transfer			82
FBYT	FileBytes	12	Number of bytes of the file	12		12
FLAB	FileLabel	80	File user identification			80
FNAM	FileName	8	Symbolic file name	8	8	8
FNRD	FileNumberOfRecords	12	Number of records sent/received			12
FOPO	FileOpenOption	1	Allocation rule, N = New file, R = Replace, O = Append			1
FORG	FileOrganization	1	File organization, S = Sequential, I = Indexed, R = Relative			1
FRLG	FileRecordLength	5	Local record length			5
FSIZ	FileSize	12	Size announced by the sender			12
JNDA	JournalRecordDate	18	Date when the journal record is written by Connect:Express			18
LNAM	LocalName	8	Alias name of the local Connect:Express			8
LPHN	LocalPhysicalName	127	Local file physical name	127	44	44
MULT	MultiArticleUsed	1	Multiarticle is used in PeSIT Fpdu Data, Y or N			1
NBYT	NetworkBytes	12	Number of bytes transferred	12	12	12

NMGS	NetworkMessageSize	4	Network message size			4
NRC	Nrc	4	Network Return code	4	4	6
PNAM	PartnerName	8	Symbolic Partner name	8	8	8
PRC	Prc	4	Protocol return code	4	4	4
PTAB	PresentationTableId	2	Identification number of the presentation table used			2
PRIO	Priority	1	Transfer priority, 0 = Urgent, 1 = Normal , 2 = slow			1
PRID	ProcessID	12	Identifiation of the process that executed the request (PID, APMEFF ..)			12
PROT	Protocol	1	Transfer protocol, D=PesitD, E=PesitE, O=Oftp, 3=Etebac3, F=ftp			1
REQP	PurgedByMonitor	1	Request is purged, Y or N			
COMP	RealCompression	4	Compression performed			
RCLA	RequestClass	1	APM class where the request has been executed			1
REQD	RequestDate	18	Date when the request is accepted by Connect:Express			18
RELA	RequestElapse	8	Transfer elapse			8
REQN	RequestNumber	12	Request number given by Connect:Express	12	8	8
REQR	Requestor	8	The name of the entity (user, job ..) that submitted the request	8	8	8
RSYN	ResynchronizationNumber	2	Number of resynchronization for the request			2
RETN	RetryNumber	2	Number of retries for the request			2
SRCT	SendReceiveCount	12	Number of network send receive			12
SERV	ServiceType	1	Type of transfer request, AdHoc or Normal			1
STAB	SessionTableId	2	Indentification number of the session table used			2
SNAL	SnaLuName	8	Remote SNA address			8
SNA1	SnaRc1	4	Primary SNA return code			
SNA2	SnaRc2	8	Secondary SNA return code			
SRC	Src	8	System Return code	8	8	4
SRC2	Src2	4	Complementary System Return code			
RSTA	Status	1	Transfer status, E = Ended / S = Started / I = Interrupted / W = Waiting	1	1	1
SYNC	SynchronizationInterval	4	Size of synchronization interval Kbytes			4
WIND	SynchronizationWindow	4	Number of synchronization intervals			4
TCPA	TcpIpAddress	15	Remote TCP/IP address			15
TCPH	TcpIpHostName	127	Remote TCP/IP host name			32
TCPP	TcpIpPort	5	Remote listening TCP/IP port			5
TCPC	TcpipPrc	4	TCP/IP return code			
TBDA	TransferBeginningDate	18	Beginning of transfer date and time	18	18	18
TDST	TransferDestination	8	The entity that is processing the transfer request		8	8
TDIR	TransferDirection	1	Transmission or Reception	1	1	1
TEDA	TransferEndDate	18	End of transfer date and time			18
TIDT	TransferIdent	8	Transfer ident exchanged with the partner			8
TORG	TransferOrigin	8	The entity that is requesting the transfer		8	8
TRCV	TransferReceiver	24	The entity that is processing the file after receiving it			24
TSND	TransferSender	24	The entity that is processing the file before sending it			24
TRTN	TranslationTableNumber	1	Translation table identification number			2
TRC	Trc	4	Connect:Express Return code	4	4	4
TYPA	TypeOfAllocation	1	Type of allocation, F = Fixed, D=Dynamic			1
TYPC	TypeOfCompression	2	Compression, Horizontal,Vertical,Mixed or pres.table identification number			1
TYPD	TypeOfData	1	Type of data, A = Ascii, E = Ebcdic, B = Binary			1
TYPF	TypeOfFile	2	Type of file, TF = Text fixed, TV = text variable, UF = Unix fixed, UV = Unix Variable, BF = binary fixed, BU = Binary undefined, S = Sequential, V = VSAM, P = PDS, PE = PDSE, PU = PDS unload, VU = VSAM unload, SU = SYSOUT unload, UU = User unload			2
TYPL	TypeOfLink	1	Type of link, 0 = LU 6.2, 1 = X25, 2 = TCP/IP, M=mixed			1
TYPP	TypeOfPartner	1	Type of Partner, Other or Tom	1	1	1
TYPR	TypeOfRequest	1	Type of request, N = Normal, I = Inquiry, H = Hold			1
TYPU	TypeOfUser	1	Type of user, I = Internal, E = External	1	1	1
USDR	UserDataReceived	254	User information received with the file		254	254
USDS	UserDataSent	254	User information sent with the file		254	254

REQU	UserRequestID	16	Identification of the request given by the user	16		8
X25C	X25Cause	2	X25 Cause			
X25D	X25Diagnostic	2	X25 Diagnostic			
X25F	X25Facilities	32	Remote X25 address, facilities			12
X25L	X25LocalAddress	15	Local X25 address			15
X25P	X25Localport	1	Local device or MCH identification			1
X25R	X25rc	4	X25 Return code			
X25A	X25RemoteAddress	15	Remote X25 address			15
X25U	X25UserDataField	16	Remote X25 address, user data field			16

Monitor's Configuration Data

Key	Field	Lg max	Description	Win	Unix	OS / 390
AHSO	AdHocSecurityUsed	1	AdHoc Racf option: Yes or Unsafe			1
ALIAS	Alias name	80	Alias name from asset protection key	80	80	
JAIA	JAIAddress	15	JAI listening address	x		15
JAIP	JAIPort	5	JAI listening port	X	5	5
JAIV	JAIVersion	3	JAI version Number	X		
AUXX	APLinexx	100	Asset protection line numer xx (n lines)			
APPV	AppcVersion	3	Appc JAI version number	3		
AUTF	AuthFile	44	AUTHDS data set name			44
BLDO	BuidDataBaseUsed	1	Automatic build Data base during termination		1	
CLDN	ClientDefaultToNotify	8	Name of the default client for notifications	8		
CLTM	ClientTimer	4	Timer for Client sessions	4		
CPUI	CpuId	20	JES2 Interface (ISF, SAM, SYSV)			20
CSVV	CsvVersion	3	Csv JAI version number	3		
DNOT	DefaultNotificationUsed	1	default client to notify is defined, Y or N	1		
SPSW	Dpcpsw	8	Partner password of the Connect:Express server		8	8
DPCI	Dpcsid	8	Partner identification of the Connect:Express server		8	8
FILF	FilesFile	44	Path and file name for the Files directory			44
FILT	FilesTotalDefinitions	5	Number of symbolic files defined			5
FTDF	FtpDefaultFile	8	Default file name for FTP		8	8
FTAL	FtpListAllUsed	1	ALL files are included into the FTP list, Y or N		1	1
FTLA	FtpListenAddress	15	FTP listening address		15	15
FTLP	FtpListenPort	5	FTP listening port		5	5
FTPN	FtpTransferNumber	3	Maximum number of FTP file transfers			3
IJNO	InitJournalUsed	1	Automatic journal initialization	1		
ILOG	InitLogfileUsed	1	Automatic logfile initialization	1		
ISTA	StatisticsUsed	1	Statistics active, Y or N		1	
JES2	Jes2Interface	4	JES2 Interface (ISF, SAM, SYSV)			4
JNLS	JournalSize	5	Number of records of the journal	5		5
LOAD	LoadLib	44	Loadlib data set name			44
LODB	LogDebugUsed	1	Syslog debugging activation		1	
LOGS	LogSize	5	Number of records of the log	5	4	5
MSGU	MessageUser	80	Message sent to operator at initialization			80
MSTA	MonitorStatus	1	Monitor status, Active or Inactive			1
NPPN	NamedPipeName	127	Name of the named pipe	127		
NPPO	NamePipeUsed	1	Named pipe resource status	1		

NRES	Network ressources	10	T/S/X/L… for TCP/IP, SNA, X25, APPC ….			10
NOTI	NotificationsUsed	1	Notifications active, Y or N			
NOTS	NotificationSize	5	Number of records of the notification file	5		5
X25C	NumberCVC	2	Number of X25 virtual circuit			2
ODPT	OdetteListenPort	5	OFTP (Odette) listening port			5
PARF	PartnersFile	44	Path and file name for the partners directory			44
PART	PartnersTotalDefinitions	5	Number of symbolic partners defined			5
PROD	ProductInfo	80	Connect:Express information (version ….)	x	14	80
RACF	RacfProfile	1	Transfers are under security control, Y or N			1
RQTS	RequestTableSize	5	Request table size			5
RETN	RetryNumber	4	Number of retries for the request	4		4
RUTY	RunType	1	Run type, Hot or Cold	1	1	1
SVCO	ServiceUsed	1	Service configuration	1		1
SMSO	SmsUsed	1	Sms is used, Y or N			1
SDL1	SnaAppcWindowsDll	15	Name of the APPC dll	15		
SNAA	SnaApplication	8	SNA application prefix or SNA LUName	8		6
SNAB	SnaControlUsed	1	SNA Incoming call control is bypassed, Y or N	1		
SDL2	SnaCsvWindowsDll	15	Name of the CSV dll	15		
SNAO	SnaUsed	1	SNA resource is active, Y or N	1		
SSNM	Subsystem name	4	Name of connect:Express subsystem interface			4
STEV	SessionTimer	4	Session timer in minutes		2	
EVTF	SysevtFile	44	SYSEVT data set name			44
SYCF	SysinConfigurationFile	44	SYSIN data set name			44
JCLF	SysjclFile	44	SYSJCL data set name			44
JNLF	SysjnlFile	44	SYSJNL data set name			44
LOGF	SyslogFile	44	SYSLOG data set name			44
PRMF	SysprmFile	44	SYSPRM data set name			44
SYSF	SyssnaFile	44	SYSSNA data set name			44
SYTF	SystcpFile	44	SYSTCP data set name			44
SYXF	Sysx25File	44	SYSX25 data set name			44
SINF	SytemInfo	255	Operating system information (version ….)	x	255	80
TCPB	TcpipBypassUsed	1	TCP/IP Incoming call control is bypassed, Y or N	1		
TDLL	TcpipDll	15	Name of the windows socket	15		
TCPO	TcpipUsed	1	TCP/IP resource is active, Y or N	1		
TPLA	TcpListenAddress	15	General listening address	15	15	
TPPT	TcpListenPort	5	General listening port	5	5	5
TWUN	TemporaryWorkUnit	6	Work Unit used for temporary files			6
TRAC	TraceUsed	1	Trace is active, Y or N	1	1	
TRFM	TransferMax	3	Maximum number of simultaneous file transfers (outbound for Unix)		4	
TTEV	TransferTimer	2	Timer for file transfer sessions in minutes		2	
TTMR	TransferTimer	4	Timer for file transfer sessions	4		
X25B	X25BypassUsed	1	X25 Incoming call control option	1		
X25N	X25LineNumber	2	Number of X25 lines			2
X25L	X25LocalAddress	15	Local X25 address ou DLL Windows	15		15
X25P	X25Localport	2	Local X25 address, port number	2		
X25O	X25Used	1	X25 resource is active, Y or N	1		
XDLL	X25WindowsDll	15	Name of the Eicon X25 dll	15		

SSL client parameters

Key	Field	Lg max	Description	Win	Unix	OS / 390
SNAM	ParameterName	8	Name of this parameter definition	8	8	
SSTA	Status	1	Status, E=Enabled, H=Disabled	1	1	
STOL	StoreLocation	64	Store location SYSTEM_STORE_LOCAL_MACHINE, SYSTEM_STORE_CURRENT_USER or SYSTEM_STORE_SERVICES	64		
STNM	StoreName	128	Store name (My)	128		
SUBJ	Subject	256	Certificate subject distinguished name	256		
ISSU	Issuer	256	Certificate issuer distinguished name	256		
PROT	Protocol	1	Protocol – ‘1’:TLSV1, ‘3’:SSLV3, ‘2’:SSLV2	1		
CISU	CipherSuites	45	List of cipher suites numbers 00: TLS_RSA_WITH_RC4_128_MD5 01: TLS_RSA_WITH_RC4_128_SHA 02: TLS_RSA_WITH_3DES_EDE_CBC_SHA 03: TLS_RSA_WITH_DES_CBC_SHA 04: TLS_RSA_WITH_NULL_MD5 05: TLS_RSA_WITH_NULL_SHA 06: SSL_CK_RC4_128_WITH_MD5 07: SSL_CK_DES_64_CBC_WITH_MD5 08: SSL_CK_RC2_128_CBC_WITH_MD5	45		
TRLV	TraceLevel	1	‘0’:None, ‘1’:Medium, ‘2’: Full	1		
SBHD	SslBytesHeaderUsed	1	A 2 bytes length header is inserted before each PeSIT data sent ‘1’: Yes, ‘0’: No	1	1	
VOPT	VerificationOption	1	Authentication ‘0’:VERIFY_NONE, ‘1’: VERIFY_PEER		1	
CERT	CertificateId	8	Connect:Express Identifier of a certificate associated to this definition. This identifier references a name chosen when importing the certificate into the Connect:Express certificates database		8	
CILI	CipherList	16	Name of a file located in $TOM_DIR/config/ssl/ciphlist and containing a list of ciphers that can be used		16	
TLS1	Tlsv1	1	‘1’:TLSV1 available, ‘0’:TLSV1 not available		1	
SSL3	Sslv3	1	‘1’:SSLV3 available, ‘0’:SSLV3 not available		1	
SSL2	Sslv2	1	‘1’:SSLV2 available, ‘0’:SSLV2 not available		1	

SSL server parameters

Key	Field	Lg max	Description	Win	Unix	OS / 390

SNAM	ParameterName	8	Name of this parameter definition	8	8	
SSTA	Status	1	Status, E=Enable, H=Disabled	1	1	
STOL	StoreLocation	64	Store location	64		
STNM	StoreName	128	Store name	128		
SUBJ	Subject	256	Certificate subject distinguished name	256		
ISSU	Issuer	256	Certificate issuer distinguished name	256		
PROT	Protocol	1	Protocol – ‘1’:TLSV1, ‘3’:SSLV3, ‘2’:SSLV2	1		
CISU	CipherSuites	45	List of cipher suites numbers 00: TLS_RSA_WITH_RC4_128_MD5 01: TLS_RSA_WITH_RC4_128_SHA 02: TLS_RSA_WITH_3DES_EDE_CBC_SHA 03: TLS_RSA_WITH_DES_CBC_SHA 04: TLS_RSA_WITH_NULL_MD5 05: TLS_RSA_WITH_NULL_SHA 06: SSL_CK_RC4_128_WITH_MD5 07: SSL_CK_DES_64_CBC_WITH_MD5 08: SSL_CK_RC2_128_CBC_WITH_MD5	45		
TRLV	TraceLevel	1	‘0’:None, ‘1’:Medium, ‘2’: Full	1		
SBHD	SslBytesHeaderUsed	1	A 2 bytes length header is inserted before each PeSIT data sent ‘1’: Yes, ‘0’: No	1	1	
VOPT	VerificationOption	1	Authentication ‘0’:VERIFY_NONE, ’1’:VERIFY_PEER, ’2’:VERIFY_FAIL_IF_NO_PEER_CERT		1	
CERT	CertificateId	8	Connect:Express Identifier of a certificate associated to this definition. This identifier references a name chosen when importing the certificate into the Connect:Express certificates database		8	
CILI	CipherList	16	Name of a file located in $TOM_DIR/config/ssl/ciphlist and containing a list of ciphers that can be used		16	
CLIA	ClientAuthenticationUsed	1	‘0’: None, ‘1’: Server requests clients to authenticate	1		
TLS1	Tlsv1	1	‘1’:TLSV1 available, ‘0’:TLSV1 not available		1	
SSL3	Sslv3	1	‘1’:SSLV3 available, ‘0’:SSLV3 not available		1	
SSL2	Sslv2	1	‘1’:SSLV2 available, ‘0’:SSLV2 not available		1	
TCPP	TcpipPort	5	SSL server listen port number	5	5	
TCPA	TcpipAddress	15	Server IP address		15	
CALI	CaList	8	Certification authority certificate identifier (as imported into Connect:Express) or name of a file containing a list of such identifiers (#LIST).		8	
DHPF	DHParamFile	16	Name of a file containing Diffie-Hellman parameters		16	