

## Introduction



PC Soft and AURA Equipements are partnering for more than 20 years, to provide a complete set of drivers and tools to access AS/400 databases, programs, and objects, natively, from WinDev applications.

**Easycom For WinDev 28 and Easycom For WebDev 28** are the latest enhancement of this product range.

This documentation is common to WinDev and WebDev environments.

You can find some specific features for [WebDev](#).

EASYCOM For WinDev provides a global access to AS/400 resources:

- [Native Access](#) to DB2/400 files, using standard WLanguage and Hyper File functions. ([HReadFirst](#), etc...)
- [SQL Access](#) using standard WLanguage SQL functions, such as [HExecuteQuery](#).
- [Simple Commands](#) or [Commands with Result Values](#) calls.
- [Native Programs and Data Queues Access](#).

To access files or use SQL queries, the basic principle is the greater compatibility with HyperFile principles and functions and W-Language. You handle an AS400 file in the same way and with the same code as a Hyperfile file but with some restrictions due to the specificities of the AS / 400, some functions or optional parameters are not supported.

## Installation

### System requirement

EASYCOM is a Client/Server middleware. It is made of :

- A **Server engine** to be installed on IBM i – AS/400
- **Client connectors and drivers** to be installed on Windows, Linux or Unix workstations and servers.

### Required configuration

#### Server

- All AS/400 from series B
- All OS/400 version up to V7R5. Minimum of V5R3 is recommended.
- TCP/IP protocol

#### Client



- TCP/IP protocol.
- Operating system: all versions supported by Microsoft.

QSECOFR profile is required to install server on AS/400.

## Installing Easycom Server

EASYCOM Server installation procedure is launched from a Windows workstation, connected to the AS/400 via TCP/IP.

It uses FTP to upload objects on the system.

Server must be installed only once. If you run an Easycom connector installation again on a Windows station, you need to eventually uncheck "Install AS/400 server" option or leave the installation procedure when the Server installation wizard is shown.

EASYCOM server consists in a set of objects (programs, commands, and files) collected into one single library, named '**EASYCOM**' (default).

It is possible to change this default library name or to [install multiple EASYCOM servers](#). In the following, library name will be referred to as EASYCOM.

## Prerequisites - TCP/IP and FTP

TCP/IP must be installed, configured, and running on the AS/400 (see the CFGTCP and STRTCP AS/400 commands for more details).

**FTP is required for Easycom installation process.** Once installed, it is no longer need for the EASYCOM normal operation.

The AS/400 FTP service can be started if needed using STRTCPSVR SERVER (\*FTP) command.

## Profile to use

**QSECOFR profile is recommended: \*SECADM and \*ALLOBJ special authorities are needed for proper installation.**



### Server installation progress

- **You must confirm the destination Library Name.**

Default name is EASYCOM.

We suggest keeping the default name as it is, unless you have to [install multiple Easycom servers](#) on the same machine, or you want to test a new version without updating the existing one.

The library will be created if it doesn't already exist.

If the library already exists, a backup copy will be created in library EAC\_BACKUP.

In the future, you need to rename this server library, or copy it, you will need to run [CFGEACSB](#) command, using the new library name, to link the objects together in the new library.

- **Installation of the demonstration files**

For the first EASYCOM installation on AS/400, the demonstration files allow to run the test and demonstration programs installed on the client workstation within development environment.

- **Give the AS/400 name or IP address AS/400 on which the software will be installed.**

- **Enter a username and password to proceed with the installation**

It is not recommended to use any other user than QSECOFR.

Some EASYCOM library objects are configured to be owned by QSECOFR.

The EASYCOMD (\*PGM) object has to be run under QSECOFR permissions.

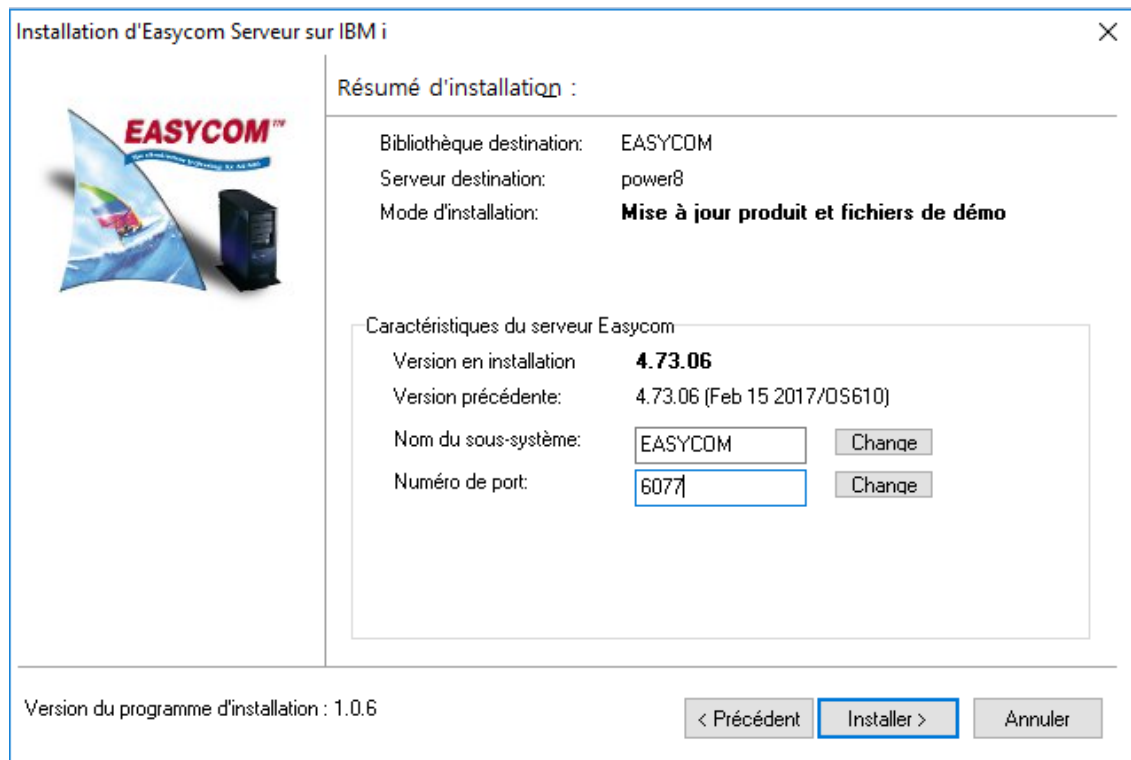
If QSECOFR is not used for installing the server, the auto-configuration may not be completed, and the first start-ups may be difficult.

### Testing the initial configuration and compatibility

The installation is first performing a routine test: if the OS/400 is compatible with the installation, if a previous version is present, ...

During the test, nothing is installed on the AS/400 (you even can cancel the process during the test)

Then it shows the following confirmation screen (here in case of a new installation):



It confirms the destination library, and if it is a new installation (for product and demo libraries), or an update. If it is an update, it shows the actual version number.

This step allows to change the current (or default) subsystem name and port values.

### Operations performed on AS/400

Creation of an EASYCOM library and restoration of some objects in this library. [EACINSTALL](#) command is automatically run by the installation process.

When installation is completed, **Easycom subsystem is started**.

This subsystem must remain active to accept client connections. See your system administrator to have the [subsystem started at IPL](#).

### Operations performed on PC

Creation of an \EASYCOM folder (by default: C:\Program Files (x86)\Easycom) and specific subfolders and copy of various files.

## Installing Easycom For WinDev on development PC

Easycom installation procedure runs on a Windows workstation.

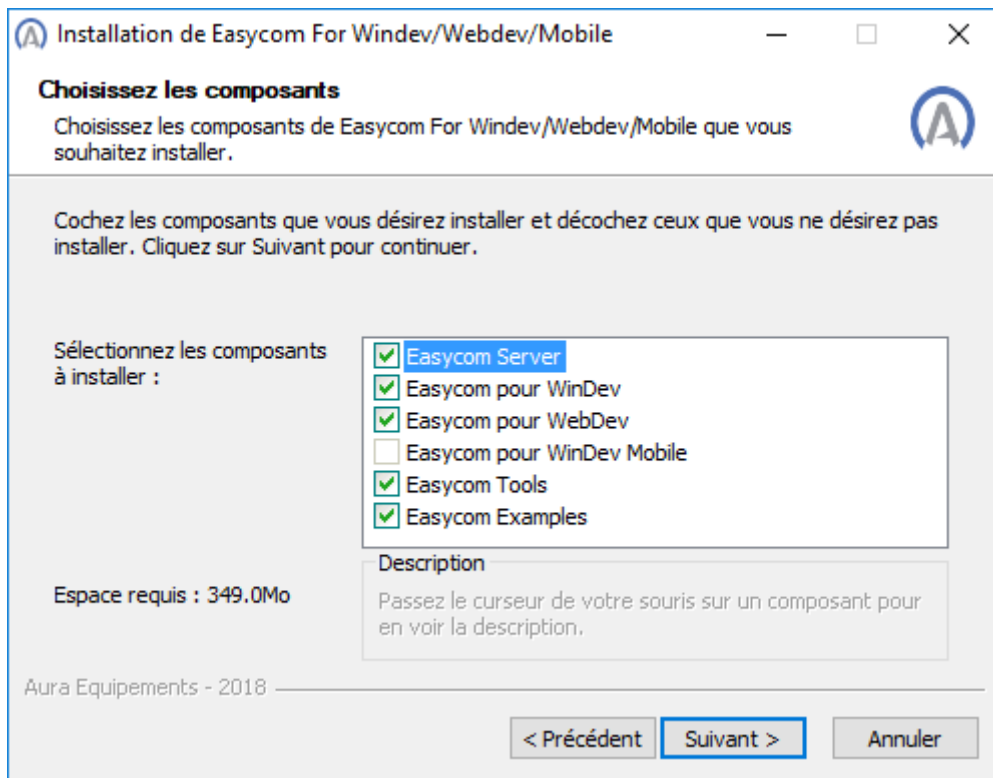
It is launched from the installation procedure, downloaded on our [website](#).

Installation package contains also the [installation procedure for the Server part](#) of EASYCOM.

If option is checked, Server installation will be launched by the client installation. Server needs to be installed only once. When you install Easycom on additional workstations, uncheck the option, or quit the installation process when it shows the server installation screen.

The same installer is used for Windev and Webdev. Select your choice according to the installed product (for Windev and Webdev) on your PC.

To proceed with the installation, at least one of the 2 products (Windev or Webdev) must already be installed on the PC.



## Installation progress

The different steps during installation:

- creation of a sub-directory in "Program Files" directory: "**Program Files\EASYCOM**".
- creation of a samples folder.
- copy connector and driver files to WinDev and WebDev directories.

Default location of the native access connectors is: C:\WinDev 28\Programmes\Framework.

## Compatibility with previous WinDev and WebDev versions.

**Installing Easycom For WinDev 28 has no effect on WinDev existing installations.**

Connector and driver files have different names.

Utility programs are in different sub-directories in "Program Files"

## Deploying WinDev applications

Installation of the client part of Easycom will be done by the WinDev deployment process.

When you create the deployment procedure for your application with WinDev IDE, the AS/400 native access DLL (**eac2800as.dll**) is always selected in the list of mandatory DLLs, as soon as an AS/400 file is used in the project, or an Easycom function is called.

We suggest using this standard WinDev mechanism to install Easycom client for your application deployments.

So, you don't have to run Easycom installation procedure on users' workstations.

Alternatively, you can also copy DLL file **eac2800as.dll** into Windows directory. If more than one application is using Easycom, all the applications will use the same DLL, and future updates of Easycom Client DLL will be easier.

Only file **eac2800as.dll** is mandatory.

**Easycom.ini : Local configuration file**

Easycom.ini file contains several settings (Cache size, Default system name, ...).

You can copy it from your development environment to the user workstations, or you can include it in your application deployment procedure.

## Shortcuts created in Windows Start menu

### Help on Easycom For WinDev/WebDev 28

Direct access to [online help](#).

### RPC – DTAQ Configuration

Tool to describe native programs to call from WinDev applications, and data queues.

See: Describing native AS/400 programs

See: [Calling Service Program procedures](#)

### Easycom Configuration

This configuration tool includes several features:

- License activation,
- Default connection configuration.
- Connection check.
- Trace file configuration
- Optimization
- Single Sign on configuration.
- Installation and Version check.

### Examples

Browses the Easycom For WinDev Samples directory.

This directory contains several WinDev sample projects using Easycom with WinDev.

### AS/400 File Export

Utility to export database files from a WinDev analysis, to AS/400 file system.

### Easycom Server installation

This shortcut is present only if you asked for Easycom Server Installation from your PC.

Using this shortcut, you can retry a server installation without launching again the full Install package.

### Easycom For WinDev Updates web page

Go to this [URL](#) to get information about the current Easycom For WinDev updates.

### Uninstalling Easycom

Before uninstalling Easycom For WinDev on PC, you need to end WinDev or WebDev development tool, and all WinDev applications running on your system.

To uninstall the client side of Easycom For WinDev, use Windows Configuration panel, Add and Suppress programs menu.



Before uninstalling Easycom server on your AS/400, disconnect all the users, and stop Easycom subsystem:

```
ENDSBS EASYCOM OPTION (*IMMED)
```

Then, you can delete the Easycom library.

## Installing an additional EASYCOM server

An Easycom Server has the following properties:

- A library with all the objects (Default name = EASYCOM)
- A Subsystem (Default name = EASYCOM)
- A TCP Port (Default = 6077)

To setup an additional server on a System, you need to install EASYCOM in a new library. The subsystem name must be unique, and a new unique port number must be assigned.

Proceed with the installation of Easycom server, from a Windows workstation.

Give a new unique name to the library in the installation wizard (Example EASYCOM2).

Once the library is installed, you need to create the new subsystem and assign a port number, by running command [CFGGEACSBS](#).

Example: To install an additional Easycom server, in library EASYCOM2, subsystem EASYCOM2, port 6078, run the following commands:

```
ADDLIBLE EASYCOM2  
CFGGEACSBS SBS(EASYCOM2) PORT(6078)
```

On the client workstations, you need to configure Easycom client, or applications, to connect to the right Easycom server.

Add the port number at the end of the name or address of the AS/400 to connect to, separated by a colon (:).

Example:

```
SYSTEMAS:6078  
192.168.0.10:6078
```

You need to change this value with "Easycom configuration" utility, if the system is the default one, or in the connection properties of your application.

## Easycom Licenses

To run Easycom For WinDev on your AS/400, you have to purchase a user license.

You must register the serial number of the Easycom for Windev product on our [website](#).

You need different activation keys for development and deployment.

User License is valid for one partition on one single AS/400 system, and for a limited number of simultaneous sessions, unless you have got a license for unlimited number of sessions.

**Warning:** One single application can open multiple connections to one AS/400 system. Each connection is counted as a session by the licensing system.

You need different licenses for **WinDev and WebDev development**.

**Deployment** license allows to run **WinDev and WebDev** applications on one single AS/400 system.

When you **change your AS/400 system**, Easycom licenses are moved to your new system for free if you have subscribed a maintenance contract.

## Development

### Connections

A connection is defined in the analysis, or dynamically defined in a program, using `HDescribeConnection` statement.

A connection defined in the analyse is automatically opened, while a file that belongs to the connection is open for the first time.

It can also be explicitly open with `HOpenConnection` statement.

Server name or address, User ID and Password are sets :

- In the connection description in the analysis,
- Or during `HDescribeConnection` call,
- Or on `HOpenConnection` call,
- Or in `easycom.ini`, file.

Sign in can be automated by using [Single Sign On](#).

Long passwords are supported.

Association between files and connections is made in the Analysis, when the connection is defined in the analysis, or using `HChangeConnection` statement.

Each connection is an active `JOB` in Easycom subsystem on AS/400.

If you have got a user license having limited concurrent sessions, took care when using `HChangeConnection`. Don't open too many connections in the same application.

An AS/400 file can be declared out of the analysis by using `HDeclareExternal` statement.

When a connection is closed with `HCloseConnection`, all files opened in the connection are closed.

### Connection parameters

All parameters are optional, we can use the following syntax:

```
HOpenConnection(NewConnexion, "", "", "", "", hAccèsNatifAS400)
```

and then complete the login window, provided that the dialogs are not disabled (see also automating the connection).

The provider is of course `hAccessNativeAS400`.

The access mode can be `hORedWrite` or `hORed`.

The three essential parameters are:

- the IP address of the AS/400 or its name (provided it is in the host file or in the DNS)
- the name of the user (profile)
- his password

Extended information can be useful for specifying settings specific to an AS/400 connection, such as the job name.

#### Example

```
MyConnection is a Connection
MyConnection..User = "me"
MyConnection..Password = "dontknow"
MyConnection..Server = "MY_SYSTEM"
MyConnection..Provider = hNativeAccessAS400
```





```
MyConnection..Access = hOReadWrite
MyConnection..ExtendedInfo = "<EASYCOM>"+CRLF+"JOBNAME="+CRLF+"</EASYCOM>"
```

```
HOpenConnection(MyConnection)
```

## ASProperty

This functions defines properties on a file or on a datasource :

- Activation of the alias name and path (only on a file),
- Selection of the member's name (only for a file).
- Cursor and cache options (for a file or a datasource),

This function allows backward compatibility with alias files (WinDev 5.5 backward compatibility)

### Syntax

```
Result = ASProperty(FileName, Property, Value [, Connection])
```

In French: [ASPropriété](#)

### Parameters

#### Result

Boolean. True if success, False if failed.

#### FileName

Character string

Name of the file in the analysis for what a property is to be changed. This is the WinDev name, not the AS/400 name!

If FileName is an empty string, the property will be changed for the connection, and so, for all the AS/400 files attached to the connection.

#### Property

Character string.

One value into the following table:

<i>Property</i>	<i>Value-Type</i>	<i>Value description</i>
MEMBER	String	Name of the file member to use. If the file has multiple members, the first i=one is open by default. File must be open after calling ASProperty. The member name can also be set by the extended info : MAIN=BIBLIO/FILE (MEMBER)
ALIASPATH	String	Path where the Alias file is located (*.AS) The alias files were used with WinDev 5.5. It is recommended to use extended info with WinDev 12.
ONLYALIAS	Boolean	Ignore extended info for the file, and use the Alias file, like i twas used with WinDev 5.5.
CACHESIZE	Integer	Maximum Number of records read.in one shot into the record cache.
FORWARDONLY	Boolean	Set the cursor non scrollable, for better performances. The file can be read forward only.
CACHEDINSERT	Boolean	When set to true, Records are kept in a cache before to be inserted into the file.



This property must be used carefully, while function `HInsert` will not insert the records in real time.

Records are physically inserted when Easycom cache is full, or when you set again this property to `False`.

You must set this property to `False` when the last record is inserted, in order to flush the cache, and write the data to the file.

## Value

New value for the property.

## Connection (optional)

Connection to AS/400

## Alias file

Alias files were used in WinDev 5.5 to identify AS/400 files. It is a text file located in the directory where the Hyperfile file is located.

```
[FILENAME]
$FILE=BIBLIO/FILE or FILE
CLE1=BIBLIO/LOGIC1
CLE2=BIBLIO/LOGIC2
...
$JNAL=TRUE // if the file is journalized
$READONLY=TRUE // readonly opening
```

`$FILE` is now the `MAIN` parameter in the extended information of the file.

The logical files associated with the keys are in the field extended information under the parameter `LFSYSNAME`.

`$JNAL` is now the `JOURNALED` parameter in the extended information of the file.

`$READONLY` is now applied by opening the WinDev connection with the read-only mode (`HOpenConnection`).

## Notes:

1. If an alias file is used, the modification of the `LIBL` of the job will not be taken into account (`ADDLIBLE`).
2. To use an alias file with `HExecuteSQLQuery`, use `hQueryDefault`

`HExecuteSQLQuery` (`MYQUERY`, `MaConnexionpower8`, `hQueryDefault`, `sString`)

Example of alias file: `SP_CUST._as`

```
[SP_CUST]
$FILE=EASYCOMXM2/SP_CUST
CUST_ID=EASYCOMXM2/SP_CUST_CU
FIRSTNAME=EASYCOMXM2/SP_CUST_NA
STATE=EASYCOMXM2/SP_CUST_ST
```

## Examples

### Use alias files

`Résultat` est un Boolean

```
// Indicate in which libray is located the alias file
```

```
Résultat=ASProperty("", "ALIAPATH", "C:\Program Files\Projet")

// ignore extended infos, before file opening
Résultat= ASProperty ("", "ONLYALIAS", "VRAI")
```

### Cache and cursor

Performances tuning for reading data from a [HExecuteSQLQuery](#).

```
// you need to execute the query first
HExecuteSQLQuery (MaRequete)
//Cache de 1000 enregistrements
ASProperty (MaRequete, "CACHESIZE", "1000")
ASProperty (MaRequete, "FORWARDONLY", "1")
```

Performance gain can theoretically be up to 50% on the reading of resulting data.

### Extended info

#### Connection Level Properties

Connection properties are:

- Set in the extended info of a connection description, in the analysis,
- Passed to function HOpenConnection , separated by CRLF,
- Set in a Connection type variable, separated by CRLF.

Easycom Properties start by literal "<EASYCOM>" and ends with "</EASYCOM>".

#### Example

```
HOpenConnection ("USER", "PASSWD", "", "MyServer",
hNativeAccessAS400, hReadWrite,
"<EASYCOM>"+CRLF+"JOBNAME="+CRLF+"INITLIBL="+CRLF+"</EASYCOM>")
```

Properties	Values & Description
AUTOJOURNAL	TRUE / FALSE During file description import, property "Journaled" is set to value "true" at file level. in the analysis. File will be journalized when open.
CODEPAGEFILE	Set the codepage file to use to convert to and from EBCDIC character set. Example: CODEPAGEFILE=
CONNECTION TIMEOUT	Delay in seconds before returning an error, if connection fail.
COMMAND TIMEOUT	Maximum Delay in seconds when waiting for an answer from the AS/400 system on a read operation for example.
DATETYPE	Set the AS/400 default data type for Date fields. See <a href="#">AS/400 Native Data Types</a> for the possible values.
TIMETYPE	Set the AS/400 default data type for Time fields. See <a href="#">AS/400 Native Data Types</a> for the possible values.
DRVOPTIMISTIC	When set to TRUE, every record update ( <a href="#">HModify</a> ) <ul style="list-style-type: none"> <li>• Locks the record,</li> <li>• Reads data,</li> <li>• Write new data if no change were made by another job since record was read.</li> </ul>

DUPPATH	Set this property to TRUE, to authorize Easycom to duplicate open paths when a filter with conditions is applied on a key, while the file is already filtered on the same key in another HyperFile Context.
EIM_LOOKUP_INFO	This is the lookup information to use when performing an EIM connection. That EIM connection can come from a certificate authentication (SSL) or from a Kerberos authentication.  This Lookup information is used to resolve ambiguous mappings: from one authentication it can go to different OS/400 users depending on the lookup information value. See EIM configuration for more details.
EXTRAIDX	Allows Easycom to use composite key as index on the first field of the composite key.  The column which is the first field of the composite key will become a sort of criteria in a Table control, and the magnifier icon will be displayed.
FIELDNAMES_MUST_MATCH	0 / 1  When = 1 fieldnames must be identical both on iseries and in the analysis (if not the HOpen fails with an error message)  When = 0 the fieldnames may be different but must be in the same physical order.
FORCELIBL	0 / 1  When set to 1, library names present in extended info, at file or field level, will be ignored. All files will be searched in the LIBL, even if it is qualified.
HFMXKEY	When the highest character in the AS/400 is not FF(hexa), you need to give it by property HFMXKEY, in order to handle correctly <a href="#">HValMax</a> .
IGNORE_EMPTY_EXTINFO	If no extended info is given for a file, Easycom will assume that the AS/400 physical file name is equal to the the WinDev file name, logical file name is equal to the key field names in the analysis.
IMPORT_DFT	0 / 1  When=1 the iseries default values are imported into the analysis.
IMPORT_SELOMIT	By default, Easycom does not use logical files with omissions or selections, as indexes.  By setting this property to 1, you allow Easycom to consider all logical files as indexes during file description import.  Warning! Logical files with selections or omissions do not contain all the data from the physical files!
INITLIBL	List the library names you want to add on top of the library list. Library names are separated by a semicolon (;). Example: INITLIBL=
LITERALCASE	Controls the way file and field names are built. See <a href="#">Controlling Character Case for file and field names</a> .
JOBNAME	Name to set to the client job on AS/400. Default job name is the workstation name.
LEADINGSPACES	WinDev 5.5 compatibility. When this option is set to True, all character field values are padded with blanks, up to the size of the database field on AS/400, on read operations.
ONLYSHORTFILEDNAMES	0 / 1 When a file description is imported, only short field

	names are used when this property is set.
PGMNAME	This option defines what is the EASYCOM program name. Default is EASYCOM. Example of value: EASYCOMT/EASYCOM.
SQLNAMING	<u>S</u> YS / SQL With value SYS, character slash (/) is used to separate library and file names. With value SQL, a point (.) is used.
SHOWDIALOGS	0 / 1 Disable / Enable Easycom dialog boxes. It is important to disable dialog boxes for web applications, or for applications running on servers. When dialog boxes are disabled, in case of error, error message is sent to the application without any intermediate message box.
SSL	True/False/Mandatory/ <u>U</u> ndefined Forces SSL mode. If true, the SSL connection will be attempted. If mandatory SSL connection will be attempted, and if the connection could not use SSL, it is aborted.
SSL_INTF	<u>W</u> indows/OpenSSL SSL interface to use.
SSL_CAFILE	In OpenSSL, the CA file to verify server certificate.
SSL_CAPATH	In OpenSSL, the CA path in which searching for authority certificates. This is to verify the server certificate.
TCP_VERSION	This option is used to force the TCP/IP version. By default, the tcp/ip version is automatically selected from the DNS. Possible values are 4 for IPv4 or 6 for IPv6.
USER_CERT_FILE	Client certificate file. This is for client certificate authentication, when OpenSSL is used. The certificate file must be in PEM format. When using Windows interface, the option is not needed: the certificate is automatically selected.
USER_CERT_PKEY_FILE	Client certificate private key file. This is the file path for the OpenSSL private key file. When using Windows interface, the option is not needed: the certificate and private key is automatically selected.
USER_CERT_PASSPHRASE	Passphrase to be able to read the private key file. This is used in OpenSSL only.
STRICTIDENTICAL	<u>Q</u> /1 When a hidentical search is performed, use strict behaviour. This uses really full key equal search (not greater equal), and the current position is unchanged in case of record not found condition
SQLFULLPRECISION	<u>Q</u> / 1 When this property is set to 1, long decimal values resulting from SQL functions are converted to character string, when the precision exceed the precision of WinDev "currency" data type.
SQLHPOS	<u>Q</u> / 1 When true, the table with direct link to data (file or query) generates more actions on the iseries to reflect intuitive behaviour (mostly scrollbar position)
SQLIDX	When set to True, this option allows to use any column of an SQL result set like an index. The user will be able to change the order of the records displayed in a table. This option can affect the performances; Use it carefully.
SQLLIVE	Possible values are: 0 (default) : It is not possible to update a record from an

	<p>SQL result set.</p> <p>1: Only files having "SQLUPDATABLE=1" extended info, can be updated out of an SQL result set.</p> <p>3: All files can be updates out of an SQL result set, except files having "SQLUPDATABLE=0" extended info. Field level extended info "SQLUPDATABLE=0" can also be set to deny modification of field values.</p> <p>Note: All the field names must appear in the "SELECT statement". You cannot use *.</p>
TCPIP_FATAL	<p>0 / 1</p> <p>By default, TCP/IP error is fatal, end stop the application. By setting this property to 0, you can manage TCP/IP errors, and for example, restart your application in case of error.</p>
UNLOCK	<p>This option is connected <a href="#">Easycom "Program Level Security"</a> Exit program. It sets the password value to pass to EACP003 Exit program to unlock Easycom server, and authorize databases and program access.</p>
USER_CERT_FILE	<p>Client certificate file. This is for client certificate authentication, when OpenSSL is used. The certificate file must be in PEM format. When using Windows interface the option is not needed: the certificate is automatically selected.</p>
USER_CERT_PKEY_FILE	<p>Client certificate private key file. This is the file path for the OpenSSL private key file. When using Windows interface the option is not needed: the certificate and private key is automatically selected.</p>
USER_CERT_PASSPHRASE	<p>Passphrase to be able to read the private key file. This is used in OpenSSL only.</p>

### File Level Properties

Properties	Values & Description
MAIN	<p>Set the main AS/400 file name associated with the WinDev file. File name can be qualified with a library name:</p> <ul style="list-style-type: none"> <li>LIBRARY/FILE</li> </ul> <p>Or without library name, to use *LIBL (Preferred):</p> <ul style="list-style-type: none"> <li>FILE</li> </ul> <p>A member name can be specified after the file name:</p> <ul style="list-style-type: none"> <li>MYFILE(MEMBER_A)</li> </ul>
JOURNALED	<p>Tell Easycom that the filed is journalized and must be opened under commitment control.</p> <p>This property is automatically set when file description is imported.</p> <p>See also AUTOJOURNAL connection level property.</p>
PFSYSNAME	File short name, when file has a long name.
SQLUPDATABLE	See SQLLIVE connection property.
DATETYPE	<p>Set the AS/400 default data type for Date fields.</p> <p>See <a href="#">AS/400 Native Data Types</a> for the possible values.</p>
TIMETYPE	Set the AS/400 default data type for Time fields.

	See <a href="#">AS/400 Native Data Types</a> for the possible values.
--	---

**SQLUPDATABLE**

To be able to modify the result of a SQL query, the extended infos of the Connection must contain the SQLLIVE option with a value of 1 or 3.

If SQLLIVE = 1, the files that can be modified must contain the option SQLUPDATABLE = 1.  
 If SQLLIVE = 3, all the files are modifiable by default, but it is possible to exclude a particular file (to prevent any update from a request), with the option SQLUPDATABLE to 0.

**DATETYPE / TIMETYPE**

This option associates all the date and time fields of the file with another type of format for an automatic query formatting in *HRequêteDéfaut* mode. It can also be defined at the connection level (see detail) or at the heading level by the NATIVETYPE option (see below).

**Field Level Properties**

Properties	Values & Description
LF	Logical File name for a key field. File name can be qualified with a library name: <ul style="list-style-type: none"> <li>• LIBRARY/LFILE</li> </ul> Or without library name, to use *LIBL (Preferred): <ul style="list-style-type: none"> <li>• LFILE</li> </ul>
SYSNAME	Short field name, when the field has a long name, and when the field is not a multi member key field.
LFSYSNAME	File short name, when logical file has a long name.
SQLUPDATABLE	See SQLLIVE connection property.
NATIVETYPE	This property is automatically set for some data types, when file description is imported. It tells the system what is the original data type in the DDS. See <a href="#">AS/400 Native Data Types</a> for the possible values.

**LF**

Chemin d'accès du fichier logique correspondant à une clé.  
 LF=Library name/Short name of the logical file

**SYSNAME**

Short name (system) for a non-key field, unique key and key with duplicate  
 SYSNAME=Short name of the field(10 caractères maxi)

**LFSYSNAME**

Nom court (système) pour le fichier logique associé à une rubrique clé composée  
 LFSYSNAME=Nom court du Fichier Logique (10 characters max)

**SQLUPDATABLE**

To be able to modify the result of a SQL query, the extended infos of the Connection must contain the SQLLIVE option with a value of 1 or 3.

If SQLLIVE = 1, the files that can be modified must contain the option SQLUPDATABLE = 1.  
 If SQLLIVE = 3, all the files are modifiable by default, but it is possible to exclude a particular file (to prevent any update from a request), with the option SQLUPDATABLE to 0.



**NATIVETYPE**

This option is automatically inserted when a Date or Time field is associated with another format. See the possible values in the DATETYPE / TIMETYPE option of the extended information of the connection.

For example, if a Date field is associated with a CHAR field (on 8 characters) we will have:

NATIVETYPE=0

**Controlling character case for file and field names**

Property **LITERALCASE** in Connection Extended info, controls the way file and field names are imported (Lowercase / Uppercase).

By default, imported file names are in lower case.

You can control file naming using the following rules with LITERALCASE property:

Value	Description
A	UPPERCASE
a	Lowercase
#nx	Next n characters are in x case (A for upper, a for lower) Example: #3A – Next 3 characters are uppercase.
*x	x case is applied until the end of the name
[	Go to end of name and change direction (Backward).
]	Go to beginning of name and change direction (Forward).
<	Change direction (Backward).
>	Change direction (Forward).

Examples :

A*a	Only First character is upper case.
#3A#3a*A	Position 1 to 3 are Uppercase, Next 3 characters are lowercase, then, next characters are uppercase.
#3A[#3A	First 3 characters are uppercase, Last 3 characters are uppercase too, middle is lowercase.

This property must be set in the connection extended info before file import.

**Example :** Set all file names to uppercase :

```
<EASYCOM>
LITERALCASE=*A
</EASYCOM>
```

**AS/400 Native Data Types**

When a date or time is stored in DB2/400 database in character or numeric format (Not in original Date or Time data type), Easycom needs to know how to convert a date or time constant value given in SQL queries, into the right data type for the AS/400.

Properties NATIVETYPE, DATETYPE and TIMETYPE in extended info can have the following values, depending on the field data type on AS/400:

Value	AS/400 data type
0	Character
1	Integer 16 bits (2 bytes)
2	Integer 32 bits (4 bytes)
3	Float simple precision (4 bytes)
4	Float Double precision (8 bytes)





6	Packed decimal
7	Zoned
8	Date
9	Time
10	Time Stamp
13	Integer 64 bits (8 bytes)

When file description is imported, Field level property **NATIVETYPE** is set to the native data type on AS/400.

In the following example, the field is packed decimal on AS/400:

```
<EASYCOM>
NATIVETYPE=6
</EASYCOM>
```

In this example, the field contains a date value (YYYYMMDD) stored as a packed decimal. You need to change the WinDev data type to "Date" in the Analysis. Easycom will convert WinDev dates To/From Packed decimal.

When NATIVETYPE is not set, properties **DATETYPE** and **TIMETYPE** in the file or connection extended info, tells the system how date and time values are stored in the database.

## Accessing AS/400 data

### WebDev Special : updating records.

In a **WebDev** project, if your application updates AS/400 files using [HModify](#) , you need to:

- Either explicitly lock the record before to update it,
- Or follow the rules described here after:

If you don't explicitly lock records before update with [HModify](#) , you need to customize the management of lock errors with:

```
HOnError ("**", hErrLock, "OnLockError")
```

Insert this statement in the project initialization, after opening the connection

```
HOpenConnection(MyConnection)
HOnError ("**", hErrLock, "OnLockError")
```

The minimum source for the procedure is :

```
// global Procedure
PROCEDURE OnLockError ()
    RETURN opCancel
END
```

When a record update is requested while the record is not locked, WebDev will lock the record and check if it was changed, only if an error procedure exists for lock errors.

Otherwise, WebDev will not try to lock the record, and the AS/400 system will fail.

## Using dynamically HF and/or AS/400 files

To dynamically choose at run time, what database manage a file (HyperFile or AS/400), you have to use functions:

[HOpenConnection](#), [HDescribeConnection](#), [HChangeConnection](#), [HDeclareExtern](#).

Import file descriptions from AS/400.

If files are initially in HyperFile format, export them to AS/400, then, import them.

Keep files in « AS/400 »

**Keep AS/400 files in « AS400 » type in the analysis, even if they are mostly used in HyperFile mode;**

Extended info are visible only when the file has an AS/400 type.

For that, you just have to keep the files attached to a connection in the analysis.

At run time, use function [HDescribeConnection](#) to describe an HyperFile connection, and change the connection with [HChangeConnection](#) for all the files that need to be handled in HF mode.

SQL Query « test » will also be possible, because it runs the « project initialization code ». You just have to run [HChangeConnection](#) in the Project initialization Code. Function [OnTestMode\(\)](#) can be used to know if we are running a test or an application.

**Note:**

In this case, connection defined in the analysis is only used to import file descriptions, edit SQL queries, ...  
You cannot use "LiveData" when your application change dynamically the connections.

## Import File Descriptions (DDS)

### Import DDS

To create file descriptions in the analysis for AS/400 files, import the Data Description Specifications (DDS) from the System I – AS/400 database.

From menu "**Structure of Files**", select "**Import Description of Files / Tables**".

Choose "**AS/400**" as data source and select "**Access data in current format**" to just get the file description, and be connected to the real data on DB2/400.

If a connection description already exists, you can use it, or create a new one.

If you have to create a new connection description, choose "**Native AS/400 Access**" mode when selecting the data source.

If you need some special connection properties, such as a different LIBL, you can create a connection description and set extended info, before to import file descriptions.

## Library where files are searched for

Default value is \*USRLIBL to import descriptions of files located in the current user LIBL.

You can change this default value and type a library name.

When file description is imported, if the file is in the LIBL, file name will be not qualified in the "Main" property of the file extended info. (MAIN=

If the file is not in the LIBL, a dialog box asks you if you want to qualify the file name (keep library name in the path: LIBRARY/FILE), or remove it.

If you don't qualify the file name, you will need to change the LIBL at run time, by changing the [INITLIBL connection property](#), or by changing dynamically the LIBL at run time with [AsExec](#) statement.

## Physical and Logical files.

When a physical file is imported, each dependant logical file having an access path will be considered as an index on the file.

Warning ! Logical files with selection or omission are not handle as indexes, except by setting [IMPORT SELOMIT connection property](#) to True.

You can manually add new indexes to the file, linked to logical files having selections or omissions.

When a logical file is imported, the resulting file has one single index: itself.

## File names

You can control the letter case in file and field names with connection property [LITERALCASE](#)

Take care of not using WinDev reserved keywords.

## Program and Data Queues.

When Selecting Tables to Import, at the end of the file list displayed, you can see the program and data queues described and available on your system. They are prefixed by \*PGM/ and \*DTAQ/.

See [Programs](#) (\*RPC) and [Data Queue](#) (\*DTAQ).

## Constraints:

Referential constraint are imported as Links between tables.

Primary key constraint is imported as Unique Key.



## Import and Export data

### AS400 To HyperFile

To import DB2/400 data into HyperFile database,

- Go to menu "**Structure of Files**" in the Data Model Editor,
- Select "**Import Description of Files / Tables**".
- Choose "**AS/400**" as data source,
- Select "**Convert data to Hyper File Classic or Hyper File Client/Server**"

### Hyperfile To AS400

To export Hyper File data to DB2/400, use the DDS Builder tool.

You can access this tool using the Windows **Start** menu – **Easycom For WinDev 28**,

Or, from the Data Model Editor, Menu "**Structure of Files**" – "**Synchronize and AS/400 – iSeries**".

DDS Builder tool doesn't synchronize the data; It can copy a whole Hyper File table to a DB2/400 table.

## Native Access

### Introduction

AS/400 Native Access allows to access DB2/400 files using standard HyperFile W Language functions from WinDev and WebDev.

SQL access is also supported by Easycom For WinDev, using standard SQL statements from WLanguage.

AS/400 files are attached to AS/400 connections in the analysis.

File description can be imported from DB2/400.

Files can be created on DB2/400, using the description set in the analysis.

WDMAP utility is compliant with AS/400 files.

WinDev Report editor also supports DB2/400 files.

### AS/400 Specific Features

Support of some WinDev or HyperFile features can be different on AS/400 system.

### Files and Fields naming

AS/400 has many restrictions in files and fields naming WinDev doesn't have.

- Accentuated characters are not supported.
- Blank character is not allowed in long names.
- Length of a long name is limited.

Easycom support long names for files and fields. Long names on AS/400 are created by SQL, or they are the DDS Alias for the fields.

By setting Connection property LITERALCASE, you can manage some naming rules for WinDev names.

When file description is exported to AS/400 using the DDS Builder, Easycom converts WinDev names into AS/400 compliant names.

### Record numbers

On AS/400 a record number is not a "stable" value. Relative Record Numbers can be changed by a Physical File reorganization.

Caution when using [HRead](#) statement.



## Crossed records

Crossed records do not exist on AS/400. **HCross** statement is not supported.

## Record lock

AS/400 locks only one record at a time, per open file, per job.

When a record is locked on a file, previous record locked on the same open file, by the same job, is unlocked.

Function **HUnlockRecNum** Unlocks the latest record locked in the open file, whatever is the value of current record.

When a lock is requested on a record already locked by another job, the system waits for a delay before to return an error. This delay (WAITRCD) can be changed by CHGPF command on AS/400, or it can be overridden with OVRDBF command called with ASExec.

File locking is supported by Easycom For WinDev.

## Transactions and Journals

Hyper File journaling doesn't apply to AS/400 files. AS/400 has its own journal system.

Functions **HchangeLogDir**, **HsetLog**, **HhistoryModification**, **HlogInfo**, **HlogRecreate**, **HlogRestart**, **HlogStop**, **HRegenerateFile**, **Hpost** are not supported by Easycom For WinDev.

See [Journals and transaction](#).

## Blobs

When file description is imported, LOB fields are imported as HyperFile memo fields.

File must be journalized, and you need to manage transactions.

When a HyperFile file having memo fields is exported, Easycom creates an additional file on AS/400 to store memos. The name of this additional file is equal to the physical file name, prefixed by 2 underscore characters ( \_\_ ).

## SQL limitations

Query on a Query is not supported.

Filter on a query is not supported.

## Special limitations on HyperFile functions

HCreation	Not supported
HCreationIfNotFound	File is opened, but not created if it doesn't exist.
HWrite	
HCross	Not supported
HFree	Not supported
HChangeLogDir	Hyper File journaling doesn't apply to AS/400 files
HChangeName	Not supported
HSetPosition	Not supported
HDescribeFile	Temporary file cannot be created on AS/400.
HDescribeItem	Temporary file cannot be created on AS/400.
HhistoryModification	Hyper File journaling doesn't apply to AS/400 files
HlogInfo	Hyper File journaling doesn't apply to AS/400 files
HlogRestart	Hyper File journaling doesn't apply to AS/400 files
HlogStop	Hyper File journaling doesn't apply to AS/400 files
Hpost	Hyper File journaling doesn't apply to AS/400 files
HRegenerateFile	Hyper File journaling doesn't apply to AS/400 files

HSecurity	Security is always active on AS/400
HsetLog	Hyper File journaling doesn't apply to AS/400 files
HVersion	Returns always 0

### Data types translations

An AS/400 data type is by default translated into a HF when importing a file description, or when executing an SQL query or using HDeclareExternal.

Default data type mapping can be manually changed by changing an item data type after DDS import, or by setting the field extended info before to export description to AS/400.

DDS type	SQL type	Condition	HF type
A (character)	CHAR		<b>Text</b>
A OPTION(VARYING) (variable length)	VARCHAR		<b>Text</b>
G + CCSID 13488 (Unicode)	GRAPHIC CCSID(13488)		<b>Text Unicode (version 12 and up only)</b>
G + CCSID 13488 + OPTION(VARYING) (Unicode variable length)	VARGRAPHIC CCSID(13488)		<b>Text Unicode (version 12 and up only)</b>
P (Packed decimal) or Z (Zoned)	DECIMAL or NUMERIC	Integer, <= 4 digits and no decimal	<b>2 bytes signed integer</b>
		Integer, <=9 digits and no decimal	<b>4 bytes signed integer</b>
		Integer, <=19 digits and no decimal	<b>8 bytes signed integer</b>
		<= 38 digits and decimal <=6	<b>Numeric</b>
		Others	<b>Text</b>
L	DATE		<b>Date</b>
T	TIME		<b>Time (HHMMSS)</b>
Z	TIMESTAMP		<b>Date and Time</b>
B4 (Short Integer)	SMALLINT		<b>2 bytes signed integer</b>
B9 (Long Integer)	INT		<b>4 bytes signed integer</b>
B19 (Integer 64 bits)	BIGINT		<b>8 bytes signed integer</b>
F	FLOAT		<b>Real 8 bytes</b>
F double precision	DOUBLE		<b>Real 8 bytes</b>
H	BINARY		<b>Binary string</b>
H	VARBINARY		<b>Binary string</b>
N/A	CLOB		<b>Memo text</b>
N/A	BLOB		<b>Other Binary Memo</b>
N/A	DBCLOB + CCSID 13488		<b>Memo Unicode</b>

Note :

Easycom maintains initial HyperFile data types when a file is exported to AS/400 and imported again. Original data type is stored in the field description.

When a file is imported, Field Extended info can contain NATIVETYPE property, to memorize the original AS/400 data type.

## Record Seek

### Logical Files

When you use function `HReadSeek(File, Key_Field, value)`, the logical file associated with the key is open and used. Its name is in the field level extended info, in the file description in the analysis.

Also, when `HFilter` is processed, the logical file matching the filter definition is used. If a selection condition is set for the filter, an OPNQRYF is applied to the logical file.

### Record seek on a query

Warning : `HReadSeek` on a non indexed column of a query will read all the records to find out the right one. This process can be long, and the result could be inconsistent if the column is not sorted.

Option SQLIDX in extended info allows to dynamically create indexes on query columns.

### Composite keys

Seek on a composite key act on AS/400 files exactly like on Hyper File files.

You must build the key value using function `hBuildKeyValue` or pass each value using an array: [value1, value 2, value3...].

Special values `HValMin` and `HValMax` .are also compliant with AS/400 files.

If a sort of sequence is defined for the file or for the job, and if the highest character is not FF (Hexa) for this sort sequence, you need to set extended info HFMAXHEY.

### Reduced composite keys

It is possible to delete fields at the end of the description of the composite key, the search on this key will then use the remaining fields. But it is easier to pass only the first Parameters by completing the terminals by `HValMin` and `HValMax`.

## Filters

### Independent HyperFile contexts.

A file can be used in more than one window with « Independent HyperFile Context » option.  
An open data path is used for each context.

File can have a filter with selection conditions applied on the same key, in each context. In this case, DUPPATH property must be set in the file Extended info.

### Simple filters.

A simple filter (with no selection conditions) uses the logical file associated with the key, to navigate into the defined range of key values.

### Filters with selection conditions.

This filter uses OPNQRYF feature on AS/400.

Filters on composite keys.

To build the composite key value, use function `HBuildKeyValue` .

Use constant keywords `hMinVal` and `hMaxVal` to complete key values.

Following example filters all customers having name beginning with "Dupond" :



```
HFilter(Clients,Name,  
HBuildKeyValue(Clients, Name, "Dupond")+hMinVal, ...  
HBuildKeyValue(Clients, Name, "Dupond")+hMaxVal)
```

### Sequence of sorting

Be careful, if the heading uses a sequence of sorting in which the last character is not the code "FF", see the option HFMAXKEY extended information of the Connection.

### Record Locking

AS/400 can lock only one record per open data path.

This means that, when a record lock is requested on a file by a WinDev program, the previous record locked applied to the same file, by the same program, is released. Except when using independent contexts, or inside a transaction.

Locking the whole file is possible.

When using independent HyperFile contexts, set connection property DUPPATH, to have one open data path per file and per context, and allow each context to have its record lock.

### Delay and retries.

When a program requests a record lock, if the record is already locked, the system waits during a delay before returning an error, in case of the record become available.

This delay can be changed on the system with CHGPF command or overridden with OVRDBF command called with [ASExec](#) function before file open.

#### Example:

```
ASExec ("OVRDBF FILE(SP_CUST) WAITRCD(*IMMED) OVRSCOPE(*JOB) ")
```

If the record lock fails, WinDev tries again. The number of retries is set by [H.NbRetry](#) WinDev keyword.

Warning : Each retry includes the system delay.

### Record modification.

[HModify](#) needs to lock the record before changing its value.

If record wasn't locked previously by an explicit "Read and lock", WinDev will read and lock the record, compare its actual value with the value it had last time current program read it.

If value was changed (by another user), WinDev showup a dialog box, and ask the user what to do.

With WebDev, this dialog box is not showed, and an error is raised. You will have to manage this error case with [HOnError](#), or you can set DRVOPTIMISTIC Connection property in Extended info.

### Constraints and integrity

Integrity check is processed, as it is for HyperFile files, with [HErrorIntegrity](#) after record update, delete or insert.

Function [HSetIntegrity](#) doesn't allow to deactivate AS/400 constraints.

### File export.

Only links of type 0,n -> 1,1 are exported during File creation on AS/400.

You will be prompted to change Links of type 1,0 -> 1,1.

Other links are not exported.

### File description import.

Referential constraint are imported as links of type 0,n -> 1,1.

Additional links can be defined in the analysis after file description import, without creating the constraint on AS/400. Those links will be handled by WinDev and could generate low performances.



## Journals and Transactions

On AS/400, transactions management uses journals and journal receivers.

- Create a journal receiver : CRTJRNRCV
- Create a journal : CRTJRN
- Start file journalization : STRJRNPF

When the file is created by Easycom For WinDev, from its definition in the Analysis, all these operations are automatically done.

In native mode, you cannot use [HTransactionStart](#). Use [SQLTransaction](#).

All files to be included in transactions must have JOURNALED property set in Extended info:

```
<EASYCOM>
JOURNALED=TRUE
</EASYCOM>
```

All imported files have this property set.

Connection property AUTOJOURNAL puts all the files in transaction mode.

Transaction is cancelled in case of error, or in case the program ends abnormally.

It is preferable to open the files inside a transaction.

When transaction is used in a program, it is preferable to not do any insert, update or delete operation out of a transaction.

Start a new transaction: [SQLTransaction\(SQLStart\)](#)

Validate the transaction (COMMIT): [SQLTransaction\(sqlCommit\)](#)

Cancel transaction (ROLLBACK): [SQLTransaction\(sqlRollback\)](#)

## Isolation Level

Default isolation level used by Easycom For Windev is equivalent to the following command:

```
STRCMTCTL LCKLVL (*CHG) .
```

Every record read for update (for a file opened under commitment control) is locked. If a record is changed, added, or deleted, that record remains locked until the transaction is committed or rolled back. Records that are accessed for update operations but are released without being changed are unlocked.

By calling STRCMTCTL with [ASExec](#) function, you can change this default.

## Performances

The most important things involved in Client/Server performances are:

- The number of network access.
- The volume of data moved over the network.

## Configuration and easycom.ini

Use Easycom Configuration. Utility, « Optimisation » tab, to set properties.

If the configuration utility is not installed on the workstation, you can edit "EASYCOM.INI" file.

### Cache size:

You can set the maximum size (in bytes) for the network buffer.

The maximum number of records read at a time is also set by this Optimisation screen.

Easycom will move bloc of records over the network, without exceeding those two maximum values.

In EASYCOM.INI files, these entries are set by:





```
[BUFFERS]
records=xx
size=yyyy
```

The "records = xx" line determines the size of the PC-side cache. Key reading can thus load the specified number of records into a single block thus reducing the number of remote accesses. The maximum size is set by the input "size = yyyy" with a maximum of 64K (65535).

Warning! A cache too large can also reduce performances.

These two properties can also be set by [ASProperty](#) function call.

#### TCP/IP Compression:

Compressing data over TCP/IP can reduce the volume of data and increase performances when the network is slow.

On a high-speed network, this property isn't very useful.

```
[TCP]
Compression=1
```

## SQLIDX

Property SQLIDX in Extended info can be deactivated to reduce the number of SQL statements executed on the AS/400.

## ASProperty

Function [ASProperty](#) sets the cache size for each file.

## File open

When a file is open by WinDev, Easycom open on the AS/400 the file pointed to by MAIN entry in the extended info. Usually, this file is a physical file. If this file doesn't have an index, maybe it will never be used by the application.

To reduce the number of open files on the system, you can set the MAIN entry in the extended info to the most used logical file.

## Lists and combos

If your application is using a lot of lists and combos filled from files that are rarely updated, use tables, or copy those files to local HyperFile files, and synchronise the contains.

## File tables with links.

If a window displays a file with links, WinDev will run a « Read by key » for each linked field in each record in the table.

This will result in a large number of accesses to the AS/400, and the cache will not be used, because WinDev reads one record at a time.

To optimise this feature, we suggest two ways:

- Use SELECT statement with INNER JOIN clause.
- Create a logical join file on AS/400, import it in the analysis, and use it to fill up the table

## Filters

Filters with selection are processed by OPNQRYF. Ranges use access paths on logical files and are faster.

It can be useful to create new logical files on AS/400 to optimise WinDev filters.

## Record locking

Try to lock the records while you are reading it when they are supposed to be updated.



So that, Easycom will not have to read it again, to lock it and check if it was changed since the last read.

## HCreateVue

Function [HCreateVue](#) with condition reads the whole file, and WinDev checks the selection condition on the PC.

So, before running [HCreateVue](#), use function [HFilter](#) with the same selection condition. Easycom will apply an OPNQRYF on the file, and reduce the number of records to read.

## Memos

Memos are stored in separated files. When WinDev reads a record, it reads all the fields, including memos.

If your application is not using the memos, deactivate the memos with function [HGèreMémo](#).

```
<Résultat> = HGèreMémo([<Nom du fichier>, [<Nom de la rubrique>,]]  
<Mode de gestion>)
```

Avec [hMémoOui](#) pour activer et [hMémoNon](#) pour désactiver.

Warning ! This function must be called while the file is closed to take effect. Close the file before to call [HGèreMémo](#).

## Blob

When WinDev reads a record, it gets all the fields, including memos and blobs.

If your application access files with blobs, and doesn't use them, it can be preferable to use SQL queries where blobs are not selected.

Or, you can create a logical file based on the physical file, without blob fields, and import it as a main file in the analysis.

## AS/400 Libraries and Files

### Libraries

It is preferable to always use the user LIBL.

If the file is in the LIBL when its description is imported, the file name is not prefixed with the library name. File name is not qualified in the file Extended info.

In this case, the file must be also in the user LIBL at application run time.

You can use [AsExec](#) function to change the LIBL at run time.

```
AsExec ("ADDLIBLE my_library"),
```

```
AsExec ("RMVLIBLE a_library")
```

The LIBL can be different for each user category. So that, a developer can have a test LIBL, and test his application without going to production data.

If you qualify the file names in the Extended info, library name will be hard coded in the analysis.

When a file is not in the user LIBL, you have to qualify file name in SQL queries, and use

[HQueryWithoutCorrection](#) mode

### User LIBL description.

When the connection is open, the user LIBL for an Easycom job contains:

- Libraries added by INITLIBL connection property, if set.
- Libraries from the user JOBD.
- Libraries from EACJOB Job description if it exists.
- Library EASYCOM.

(See also [Default LIBL](#) )



Then the application can change the LIBL at run time with `ASExec ("ADDLIBL ...`

LIBL changes with `ADDLIBL` do not affect files already open.

## OVRDBF

Using OVRDBF, you can set the library where the file is open.

**Don't forget to override also logical files !**

```
ASExec ("OVRDBF FILE (FILE1) TOFILE (LIBR2/FILE2) MBR (MEMBRE)
OVRSCOPE (*JOB) ")
```

## Libraries and SQL queries with hQueryDefault mode.

Only file name is used in a SQL query.  
So, it is easier to have the file in LIBL.

If the file is not in the LIBL, you can dynamically change the current library before to execute a query, with [ASExec](#).

```
ASExec ("CHGCURLIB MYLIB")
HExecuteQuery (REQ_Requext3, hQueryDefault)
```

## Physical and Logical Files

On AS/400, Data are stored in physical files (PF). A physical file can have one index.

Logical Files (LF) are based on Physical Files.

They point to PF data, they can point to multiple physical files (Multi formats, Join files)

LF can have record selections and omissions.

They can have column selections or expressions.

They can be an access path (Index) to the data.

## Physical Files

When a Physical File description is imported into the Analysis by Easycom, the following rules apply:

- The physical file is the MAIN file.
- If the Physical Files has an index, it is imported also as Index.
- Some dependant Logical Files are **not** imported as indexes
  - LF with selection or Omission
  - LF with field selections
  - Multi format LF,
  - Join Logical Files.
- Other Logical Files having an access path are imported as indexes.

AS/400 file names appear in the file extended info, or field extended info for the indexes.

## Logical Files

A Logical File omitted by the import process can be manually added as an index, in the analysis.  
Set the field as an Index in the file description, and add the following in the field extended info:

```
<EASYCOM>
LF=Logical file name
</EASYCOM>
```

Be careful: All logical files do not point the whole data! Check the selection rules.

A logical file can be imported as a MAIN file. The file will have only one index: Itself.

## Join Files

A join file point to more than one physical file. It cannot be an index to the data. To use it, import its description as a main file.

## Overrides

From your application, you can override a database file before to open it, with function `ASExec ..`. Don't forget `OVRSCOPE (*JOB)` in `OVRDBF` command:

```
ASExec ("OVRDBF FILE(FILE1) ... OVRSCOPE (*JOB) ")
```

## Join Files

A join file is a logical file based on multiple physical files.

It joins together records from different files. It is like a permanent SQL SELECT view.

When the join can be resolved with existing indexes on physical files, the join file doesn't spend any disk space and time.

So, it can be interesting to create join files, and use it with WinDev in file tables with linked fields. Instead of reading by key for each linked field, WinDev will get all the field values in a logical record.

A Join File can be accessed in Read Only mode.

## Logical Files with OMIT or SELECT

Logical Files with omissions or selections are not considered as indexes by Easycom For WinDev.

Date are filtered by omission or selection conditions applied on the file itself.

They are not automatically imported as indexes, but you can manually add them as indexes, as far as you are sure that all the data needed by your application is pointed to by the logical file.

To do that, created the index in the file definition in the analysis, and update the field extended info:

```
<EASYCOM>
LF=Logical file name
</EASYCOM>
```

You can also import those logical files as main file. A file is then created in the analysis, with one single index: itself.

Using `HDeclareExternal`, you can also use those logical files, without importing the definition into the analysis.

## DDM Files

A DDM File (DDMF) is a link from the local AS/400 system to a file on a remote system.

You cannot import a DDMF in the analysis.

You can use a DDMF with `HDeclareExternal` function.

## Multi Format Files

A Multi Format logical file is based on multiple physical files.

All the records do not have the same description (format). WinDev can handle only one format per file.

When a Multi Format file is open, its first format is used by default.

To override this, you can change the file name in the extended info, by adding:

```
*FORMAT=
```

After the file name:

Example:

```
<EASYCOM>
MAIN=MYLIB/CUSTFILE *RFORMAT=format2
</EASYCOM>
```

## System 36 Files

If the System 36 file has an IDDU (Interactive Data Definition Utility), it can be accessed like any AS/400 file.

If the file doesn't have a description, you need to proceed as follow:

- Create an empty AS/400 file having the record description the 36 files should have.
- At the end of the 36 file name, add the extension: `*FMT= Empty_AS400_File`



Example:

```
<EASYCOM>
MAIN=LIB36/MY36F *FMT= DESCLIB/FIC36_DESC
</EASYCOM>
```

In the example above :

- MY36F is a file with no description.
- FIC36\_DESC is an empty AS/400 file, created with the description MY36F should have

## SQL

### Creating Queries

SQL Queries can be created using the wizard, or manually input in a character string.

The *hModifyFile* mode constant can be set when executing a query, if property SQLLIVE is set in connection level extended info (Value 1 or 3).

In this case, files involved in the query are updated when the query result is modified.

By default, additional queries are prepared to be able to change records order in tables and allow the use of magnifiers in tables. This feature can be disabled with property SQLIDX in connection extended info.

Area fields cannot be used with SQL queries.

Filters on queries are not supported.

### Queries and transactions

AS/400 SQL queries can be processed inside transaction.

Use regular WinDev transaction functions:

```
sStmt is string
MyQuery is Data Source
SQLTransaction (sqlStart, MyConnection)
sStmt="UPDATE SP_CUST SET Firstname='Jean' WHERE CUST_ID='C-01'"
HExecuteSQLQuery (myQuery, MyConnection, hQueryDefault, sStmt)
SQLTransaction (sqlRollBack, MyConnection)
```

### Using mode HQueryDefault

The *hQueryDefault* mode is used when files involved in the query are in the analysis, and when the statement doesn't contain any SQL/400 proprietary keyword or function, such as a library name.

If the files are not in the LIBL, use function *ASExec* to add the library in the LIBL, or to change the current library:

Example:

```
ASExec ("CHGCURLIB MYLIBRARY")
HExecuteQuery (Query3, hQueryDefault)
```

### Using mode HQueryWithoutCorrection

When the *hQueryWithoutCorrection* mode is used, the SQL statement is sent directly to SQL/400 without being interpreted or changed by WinDev SQL engine.

You must use this mode when the statement contains some syntax or keywords WinDev doesn't know.

#### **hQueryDefault**

Connection is automatically identified, using the

#### **hQueryWithoutCorrection**

Connection must be given when calling



files involved in the query.

Replace all PC Soft proprietary operator  
(Example: ']=')

Date and time constant are translated into SQL/400 format.

Decimal point is adjusted for decimal constant, depending on the system language.

Alias names are replaced by original column names in the Where, Order by and Group by clauses.

[HExecuteSQLQuery](#).

No replacement is done. You need to type SQL/400 compliant syntax.

No format translation. You need to type data and time values in SQL/400 syntax.

You need to use the right decimal point for your system.

No replacement. Use only original database column names.

## Date and Time as parameter values.

With mode [hQueryDefault](#) date and time values are automatically translated from WinDev to AS/400 format.

```
dToday is Date
HExecuteQuery (REQ_3, hQueryDefault, dToday)
```

When the value is hard coded, it must be in 8 digits format (YYYYMMDD).

```
HExecuteQuery (REQ_3, hQueryDefault, "20080125")
```

See connection extended info DATETYPE and TIMETYPE for information about handling date and time values stored in different data types on AS/400.

With the [hQueryWithoutCorrection](#) mode, date and time values must be given in **\*ISO** format.

Time : HH:MM:SS,

Dates : YYYY-MM-DD,

### Example

```
HExecuteQuery (MyQuery, MyConnect, hQueryWithoutCorrection, "00:00:03")
```

```
HExecuteQuery (MyQuery, MyConnect, hQueryWithoutCorrection, "1970-07-01")
```

## Prepared Queries

### HPrepareSQLQuery

This function declares the query on the database server, to optimize data access. The query is prepared, but not executed. No data is retrieved at this point.

This function optimizes data access in case of multiple execution of a query, with different parameters.

The query is then executed with function [HExecuteQuery](#), after setting parameter values if needed.

To free resources used by the prepared query, use function [HCancelDeclaration](#).

Warning: Some files on AS/400 can remain open. This is a normal behaviour. The AS/400 system optimizes open access paths.

**Queries calling stored procedures, and returning multiple result sets are not supported.**

### HExecuteSQLQuery

This function executes a query created with the wizard, or a statement given as a character string.



Syntax with connection parameter must be used when files used in the SQL statement are not in the analysis, or, if `hQueryWithoutCorrection` is used.

`hQueryWithoutCorrection` must be used if the SQL statement contains library names, or some SQL/400 proprietary keywords.

## Parameters

Parameters name in the statement must be prefixed by a colon (:).

```
//Prepare SQL Query for multiple executions
HPrepareSQLQuery ( Insert , connection ,
hQueryWithoutCorrection... ,
"INSERT INTO employees VALUES (:name, :firstname, :age)" )
...
//Set param values; Execute
Insert.name = EmpName
Insert.firstname = EmpFirstName
Insert.age = EmpAge
HExecuteQuery (Insert)
```

## Example

```
////INSERTION BLOB

HouvreConnexion(MaConnexionpower8)

IdLast est un entier

schaine est une chaîne

Marequete est une Source de Données

AExec("ADDLIBLE CED5")

xes est un Buffer
xes = fChargeBuffer("c:\temp\capture.gif")

schaine="INSERT INTO CEDBL02(nom,photo) VALUES(:nom,:photo)"

SI PAS
HPrépareRequêteSQL(Marequete, MaConnexionpower8, hRequêteSansCorrection, schaine)
ALORS
    Erreur(HErrreurInfo())
FIN

Marequete.nom = "aura"
Marequete.photo = xes

QUAND EXCEPTION DANS
    SQLTransaction(sqlDébut, MaConnexionpower8)

    SI PAS HExécuteRequêteSQL(Marequete) ALORS
        Info(HErrreurInfo())
    FIN

    SQLTransaction(sqlFin, MaConnexionpower8)

    SI HLitDernier(Cedblo2, Id) ALORS
```

```
        IdLast = Cedblo2.Id
        Info(IdLast)
    SINON
        Info("Fichier non trouvé " + ErreurInfo())
    FIN

FAIRE
    SQLTransaction(sqlAnnule, MaConnexionpower8)
FIN
```

## W-Language Functions for AS/400

### Program and Procedure calls

#### *Easycom XML*

##### Introduction

Easycom XML is an interface allowing you to describe and call IBM i procedures and programs. The information exchange between WinDev and Easycom is done using XM. The program parameters and the result of those programs are in XML format.

The programs and structures can be described in RPG or PCML.

This interface is less rigid than the classical AsPgmCall, it also allows you to call procedure from Service Program, as well as pass or receive more complex parameters/results structures. It also has a better error handling and there for it is easier to diagnose and solve program calls problems.

See the following functions for details: [ASXMLDefine](#), [ASXMLLoadDefinition](#), [ASXMLBindSrvPgm](#) and [ASXMLCallPgm](#)

##### ASXMLDefine

Loads the definition of a procedure or a program from PCML or simplified RPG.

##### *Syntaxe*

```
Result = ASXMLDefine(Type, Description, Connection)
```

In French: [ASXMLDefinie](#)

##### *Parameters*

**bResult**

Boolean - True if the call succeeded, False otherwise.

**Type**

Source type, only two values are allowed, 'PCML' or 'RPG'

**Source**

PCML or RPG description of the program or procedure.

**Connection**

Connection – name of the Connection

##### *Details*

The data structures, programs and procedures are defined using a Windev String and then stored into the Easycom Server.



Simplified RPG: Simplified RPG is an identical language to RPG, without any column restrictions, and for which all the instructions must end with a semi-colon. The /COPY instruction can be used to import external RPG definitions.

### Example

Calling procedure S\_FCUST :

```
HOpenConnection(MaConnection1)
sRPG is a string = [
DS_CUST      E DS  extname(S_CUSTOMER);
S_FCUST      PR  LIKEDS(DS_CUST);
TERM1       5P 0
]

IF NOT ASXMLDefine("RPG",sRPG ,MaConnection1) THEN
    Info(ErreurInfo())
ELSE
    Info("The definition has been loaded")
END
```

### ASXMLLoadDefinition

Loads a PCML or RPG definition of a program or procedure from a file located on the System I.

### Syntax

```
Result = ASXMLLoadDefinition(Type, File, Connection)
```

In French : [ASXMLChargeDefinition](#)

### Parameters

#### **bResult**

Boolean - True if the call succeeded, False otherwise.

#### **Type**

Source type, only two values are allowed, 'PCML' or 'RPG'

#### **File**

Full path to the file on the System I.

#### **Connection**

Connection - name of the Connection

### Details

The data structures, programs and procedures are defined in a file on the System I and loaded into the Easycom Server.

Simplified RPG: Simplified RPG is an identical language to RPG, without any column restrictions, and for which all the instructions must end with a semi-colon. The /COPY instruction can be used to import external RPG definitions.

### Example

```
HOpenConnection(MaConnection1)
sFichier is a string
sFichier = "EASYCOMXMP/QRPGLESRC,CVTNW_H"

IF NOT ASXMLLoadDefinition("RPG",sFichier ,MaConnection1) THEN
    Info(ErreurInfo())
ELSE
```

```

END      Info("The definition has been loaded")

```

## ASXMLBindSrvPgm

Binds a service program to the Easycom Job.

### Syntax

```
Result = ASXMLBindSrvPgm(NameOfTheServiceProgram, Connection)
```

In French : [ASXMLAttacheSrvPgm](#)

### Parameters

#### Result

Boolean - True if the call succeeded, False otherwise.

#### NameOfTheServiceProgram

The name of the service program to load. This can be a fully qualified name (library/name), if not, \*LIBL is used to look for the service program.

#### Connection

Connection - name of the Connection

### Details

Binds a service program to the Easycom job associated with the connection. All the entry point of the service program can then be called using the [ASXMLCallPgm](#) function.

### Example

```

HOpenConnection(MaConnection1)
sSrvPgm is a string

sSrvPgm = "EASYCOMXMP/XMPSRVPGM"

IF NOT ASXMLBindSrvPgm(sSrvPgm ,MaConnection1) THEN
Info(ErreurInfo())
ELSE
Info("The service program has been loaded")
END

```

## ASXMLCallPgm

Do a program or procedure call. The program/procedure definition must have been loaded previously through [ASXMLLoadDefinition](#) or [ASXMLDefine](#).

### Syntax

```
Result = ASXMLCallPgm (Program,ParametersIN, [Type],Connection)
```

In French : [ASXMLAppelPgm](#)

### Parameters

**Result**

XMLString- returns the results of the program in XML format. In case of an error, it contains the <asxmlerr> tag with details of the error. In case of a success, it will contain the <returnvalue> and <parameterlist> tags.

**Program**

Name of the program to call.

**ParametersIN**

Input parameters of the program in XML format

**Type**

Type – Optional. Data type. Possible values: JSON or XML. If not present, XML type is used.

**Connection**

Connection - name of the connection

### Details

The exchange of information between the I Series and Windev is done using XML. Input parameters of the program must be sent using an XML format matching the structure expected by the program.

Example of XML input for the ASXMLCallPgm command, to call the RPCSAMPLE program:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Easycom Version="02.02.004">
<Program Name="RPCSAMPLE">
  <ParameterList>
    <OP1>2</OP1>
    <STR1>aura</STR1>
    <OP2>3</OP2>
    <STR2>test</STR2>
    <OP3>4</OP3>
  </ParameterList>
</Program>
</Easycom>
```

Output values are also returned within XML format, with the <PARAMETERLIST> tag :

```
<?xml version="1.0" encoding="UTF-8" ?>
<Easycom Version="2.01">
<Program Name="RPCSAMPLE">
  <ParameterList>
    <OP2 Type="Packed">5.00</OP2>
    <STR2 Type="Char">aura</STR2>
    <OP3 Type="Packed">6.0000</OP3>
  </ParameterList>
</Program>
```

### Complete example

Call the RPCSAMPLE program. The program structure is defined with pcml format.

**Note: into pcml, for the path variable, LIB and PGM must be in lowercase:**

```
path="/QSYS.lib/EASYCOMXMP.lib/RPCSAMPLE.pgm"
```

```
sPCML est une chaîne
sParam est une chaîne
sResultat est un chaîne ANSI
```

```
sPCML = [
```



```

        <pcml version="4.0">
        <program name="RPCSAMPLE" path="/QSYS.lib/EASYCOMXMP.lib/RPCSAMPLE.pgm" >
        <data name="OP1" type="packed" length="5" precision="2" usage="input" />
        <data name="STR1" type="char" length="20" usage="input" />
        <data name="OP2" type="packed" length="5" precision="2" usage="inputoutput" />
        <data name="STR2" type="char" length="30" usage="inputoutput" />
        <data name="OP3" type="packed" length="10" precision="4" usage="output" />
        </program>
        </pcml>
    ]

    sParam = [
        <OP1>%1</OP1>
        <STR1>%2</STR1>
        <OP2>%3</OP2>
        <STR2>%4</STR2>
        <OP3>%5</OP3>
    ]

    STStrpcsample est Structure
    OP1 est réel
    STR1 est chaîne sur 20
    OP2 est réel
    STR2 est chaîne sur 30
    OP3 est réel

    FIN

    stRpcSample est STStrpcsample
    stRpcSample.OP1 = 2
    stRpcSample.STR1="aura"
    stRpcSample.OP2 = 3
    stRpcSample.STR2 = "test"
    stRpcSample.OP3 = 4

    sParam =
    ChaîneConstruit(sParam,stRpcSample.OP1,stRpcSample.STR1,stRpcSample.OP2,stRpcSample.S
    TR2,stRpcSample.OP3)

    SI PAS ASXMLDefinie("PCML",sPCML,PrinciConnexion) ALORS
        Info(ErreurInfo())
    SINON

        sResultat = ASXMLAppelPgm("RPCSAMPLE",sParam,PrinciConnexion)

        Désérialise(stRpcSample,sResultat,psXMLAgrégé)
        Info("OP1="+ stRpcSample.OP1 + " - STR1=" + stRpcSample.STR1 + " - OP2=" +
    stRpcSample.OP2 + " - STR2=" + stRpcSample.STR2 + " - STR3=" + stRpcSample.OP3)
        Info(sResultat)
    FIN
    
```

### ASRtvCall

Call an AS/400 command having return values (Retrieve commands).

### Syntax

```
Result = ASRtvCall(Command [,Connection])
```

French keyword: [ASAppelRtv](#)

## Parameters

### Result

Boolean: True if the command ran successfully, False if it failed.

### Command

Character string: Contain the OS/400 command to execute.

### Connection [optional]

Connection to the AS/400.

## Details

Most of the commands that can be called using this function are the "retrieve" commands (**RTV\***) and "receive" commands (**RCV\***).

You can create your own commands returning result values (CRTCMD).

Commands can return one or more results.

**Example:** RTVJOBA can return multiple attributes of the current job.

```
RTVJOBA USER(&USER) CURLIB(&CURLIB) OUTQ(&OUTQ)
```

Some result variables need to be declared when calling the command.

For it, the command string is prefixed with the variable declarations, separated by semicolons, in the form:

**VARIABLE=Type(n,p).**

VARIABLE is the variable name.

Type can be:

- DEC(n,p) For Decimal type;.  
n = number of digits, including decimal digits.  
p = number of decimal digits.
- CHAR(n) For Character; .  
n = number pf characters.

### Examples:

```
CmdLine ="CCSID=DEC(5 0);RTVJOBA JOB(&JOB) USER(&USER) USRLIBL(&USRLIB) SYSLIBL(&SYSLIB)
CCSID(&CCSID) CURLIB(&CURL)"
```

```
CmdLine =" DATA=CHAR(100);RTVDTAARA DTAARA(MYDTAARA (*N 100)) RTNVAR(&DATA)"
```

You get the result value of a variable with function [ASRtvResult](#).

You need to get and check value of internal result variable **RC** to know if the command ran successfully.

## Example

```
LigneCmd est une chaîne
LigneCmd="CCSID=DEC(5 0);RTVJOBA JOB(&JOB) USER(&USER) USRLIBL(&USRLIB)
SYSLIBL(&SYSLIB) CCSID(&CCSID) CURLIB(&CURL)"
```

```
HouvreConnexion(MaConnexionpower8)
```

```
SI PAS ASAppelRtv(LigneCmd)ALORS
    Info(ErreurInfo(errRésumé))
FIN
```

```
sResultat est une chaîne
sResultat = ASResultatRtv("RC")
SI sResultat = "0" ALORS
    Info(ASResultatRtv("CCSID"))
SINON
    Info("Erreur de la commande : "+sResultat)
```

FIN

## ASRtvResult

This function is used after [AsRtvCall](#) function call.  
It retrieve the values for the result variables after a command call.

## Syntax

```
Result = ASRtvResult(Variable [,Connection])
```

In French: [ASResultatRtv](#)

## Parameters

### Result

Result value (String).

### Variable

Name of the variable from what we retrieve the result value.

### Connection (optional)

Connection to AS/400.

## Detail

**ASRtvResult** retrieves the value of a result variable, after a command call issued with function [ASRtvCall](#).

RC is an internal result variable.

If the command call fails, variable RC contains the CPF error code, otherwise, it contains "0".

So, check RC value before to retrieve any other variable results.

## Examples

### 1) Retrieve job attributes.

```
CmdLine is a string
CmdLine = "RTVJOBA USER(&USER) USRLIBL(&USRLIB) SYSLIBL(&SYSLIB) CURLIB(&CURL)"

ASRtvCall(CmdLine)

sResult is a string
sResult = ASRtvResult ("RC")
IF sResult = "0" THEN
    sUser=ASRtvResult ("user")
    sUserlib= ASRtvResult ("usrlib")
    sSyslib= ASRtvResult ("syslib")
    sCurl= ASRtvResult ("curl")
END
```

### 2) Retrieve system serial number.

```
bRet is a boolean
Result is a string
```



```

var1 is a string
CmdLine is a string

CmdLine = "RTVSYVAL SYSVAL(QSRLNBR) RTNVAR(&VAR1)"
bRet = ASRtvCall (CmdLine)

sResult is a string
sResult = ASRtvResult ("RC")
IF sResult = "0" THEN
    Result = ASRtvResult ("VAR1")
    Info("AS/400 serial number = " + Result)
END
    
```

## AsExec

Execute an OS/400 command.

If the commands returns result values, use function [ASRtvCall](#) .

To call a program, and retrieve modified parameters, use [ASRunRPC](#).

## Syntax

```
Result = ASExec(Command [, Connection])
```

## Parameters

### Result

Boolean: True if the command ran successfully, False if it failed.

### Command

Character string: Contain the OS/400 command to execute.

### Connection [optional]

Connection to the AS/400.

## Detail

This function can execute any command that doesn't open a terminal screen on AS/400.

It is useful to change job attributes, library list, or call programs that do not return result values.

## Examples

```

// Send a message to "QPGMR"
ASExec("SNDMSG MSG('Hello') TOUSR(QPGMR) ")

// Change current library
ASExec("CHGCURLIB PROD2005")

// Add a library in LIBL
ASExec("ADDLIB DEVLIBR")

// Call a program
ASExec("CALL PGM(MYPROG) PARM('00213') ")

// Create a journal receiver, start journalization.

sCmd is a string
sCmd="CRTJRNRCV JRNRCV(EASYCOM/TMPRCV)"
IF NOT ASExec(sCmd) THEN
    
```

```

    IF ExtractString(ErrorInfo (),2,CR)="CPF7010" THEN Info("Receiver already exists") ELSE
    Info(ErrorInfo)
ELSE
sCmd="CRTJRN JRN(EASYCOM/TMPJRN) JRNRCV(EASYCOM/TMPRCV)"
IF NOT AExec(sCmd) THEN
    IF ExtractString (ErrorInfo (),2,CR)="CPF7015" TEHN Info("Receiver already contains this journal
") ELSE Info(ErrorInfo)
END

sCmd="STRJRNP FILE(EASYCOM/SP_CUST) JRN(EASYCOM/TMPJRN)"
IF NOT AExec(sCmd) THEN
    IF ExtractString (ErrorInfo (),2,CR)="CPF7030" THEN Info("File already journalized") ELSE
    Info(ErrorInfo)
END

// Display a file description to a QTEMP file and read it.

gdsOutput is Data Source
AExec ("DSPFD FILE(EASYCOM/SP_CUST) TYPE(*MBR) OUTPUT(*OUTFILE) OUTFILE(QTEMP/OUTPUT)")
HDeclareExternal ("QTEMP/OUTPUT", gdsOutput, MyConnection)
HReadFirst (gdsOutput)

...

AExec ("DLTF FILE(QTEMP/OUTPUT)")
    
```

## AsRunRPC

Calls an AS/400 program or procedure, with parameters exchange.  
 You cannot use **AsRunRPC** to call a program with no parameter. Use **AsExec**.

## Syntax

```
Result = ASRunRPC (ProgramName)
```

In French: [ASLanceRPC](#)

## Parameters

### Result

Boolean: True if the command ran successfully, False if it failed.

### ProgramName

Character string – Program Name. It is also the name of the file imported in the analysis.

Note: This function uses always the connection associated with the file in the analysis.

## Detail

Using this function, WinDev program can call any AS/400 program or procedure, with parameter passing. Value of parameters changed by the called program can be retrieved by WinDev program.

AS/400 Program or procedure must be described using [RPC-DTAQ Description utility](#), and imported in the analysis.

Before to call the program with AsRunRPC, input and input/output parameters must be initialised.

After program call, output, and input/output parameters are updated. You can retrieve values like you retrieve database field values.

## Notes

On AS/400, program is executed in the job associated to the connection.



See [EASYCOM Jobs](#).

If you have to call a program, without retrieving modified parameter values, you can use `AsExec()` function to run CALL command.

```
ASExec("CALL PGM(MyProgram) PARM('My param 1' 'My Param 2')")
```

A maximum execution time can be set for program calls. See EASYCOM Server configuration.

## Example

The description of the RPCSAMPLE program has been imported into the analysis.

```
RPCSAMPLE.OP1=2
RPCSAMPLE.OP2=3
RPCSAMPLE.OP3=0
RPCSAMPLE.STR1="aura"
RPCSAMPLE.STR2=" "

SI ASLancerRPC(RPCSAMPLE) ALORS
    Info("OP1=" + RPCSAMPLE.OP1)
    Info("OP2=" + RPCSAMPLE.OP2)
    Info("OP3=" + RPCSAMPLE.OP3)
    Info("STR1=" + RPCSAMPLE.STR1)
    Info("STR2=" + RPCSAMPLE.STR2)
SINON
    Info(HErreurInfo())
FIN
```

## ASPgmCall

ASPgmCall is an alternative solution to ASRunRPC.

The is the same purpose: calling native programs, using different method.

First of all, importing the program's description in the analysis is not needed.

The description can reside :

- in PCML syntax, inside your program
- or: in PCML syntax, in an IFS file on your system
- or: in the program's description, made using the RPC/DTAQ setup tool.

The input/output values are directly managed using the ASPpgmCall function call, one by one or using a structure variable datatype.

## Syntax

```
Result = ASPgmCall(ProgramDescription, ProgramName [, Connection], Param1 [, Param2
[, Param3, ...] ])
```

In French: [ASAppelPgm](#)

## Parameters

### Result

Boolean: True if function succeed, False otherwise.

### ProgramDescription



Character string:

- When the program to call was described using using [RPC-DTAQ Description utility](#), this parameter has the form: **\*PGM/NAME**. where NAME is the name of the description given in RPC/DTAQ Configuration tool.
- When the program is described in **PCML** syntax in a file, this parameter contains an **IFS Path**.
- When the program is described in **PCML** syntax in the WinDev program, this parameter contains the **PCML description** itself.

### ProgramName

Character string: Real name (and library if needed) of the native program to call (LIB/NAME).

This parameter can be an empty string if the program name to call is the one given by the PCML description, or if \*PGM/NAME is used for the program description.

### Connection [optional]

Connection to AS/400.

### Param1, Param2 , ...

Variables or values for each parameter expected by the native program.

## Details

In the following example, AS/400 program was described in the RPC-DTAQ configuration tool.

Program expects 5 parameters, mapped to WinDev variables OP1, STR1, OP2, STR2, OP3.

```
IF NOT ASPgmCall ("*PGM/RPCSAMPLE", "", connect1, OP1, STR1, OP2, STR2, OP3)
THEN
    Error (ErrorInfo (errFullDetails) )
END
```

For the following example, program description is stored on IFS file "/tmp/rpcsamp1.pcml", in PCML format.

```
IF NOT ASPgmCall ("/tmp/rpcsamp1.pcml", "", cnx, OP1, STR1, OP2, STR2, OP3)
THEN
    Error (ErrorInfo (errFullDetails) )
END
```

PCML description can be created by AS/400 ILE compilers.

Example with RPG ILE compiler:

```
CRTRPGMOD ... PGMINFO(*PCML) INFOSTMF('/tmp/rpcsamp1.pcml')
```

PCML syntax looks like this:

```
<pcml version="1.0">
<!-- PCML source for calling "RPCSAMPLE" program -->
<!-- Program "RPCSAMPLE" and its parameter list -->

<program name="RPCSAMPLE" path="/*LIBL.lib/RPCSAMPLE.pgm">

<data name="Op1" type="length="5" precision="2" usage="input"/>
<data name="Str1" type="char" length=" usage="input"/>
<data name="Op2" type="packed" length="5" precision="2" usage="inputoutput"/>
<data name="Str2" type="char" length="30" usage="inputoutput"/>
<data name="Op3" type="packed" length="10" precision="4" usage="output"/>

</program>

</pcml>
```

## Example 1

Parameters passed to the program or procedure can be mapped to a WinDev structure.

In this example, RPCSAMPLE has been defined with RPC/DataQueue tool.

```
STStrpcsample is structure
rOp1 is real
```

```

sStr1 is string on 20
sOp2 is real
sStr2 is string on 30
rOp3 is real
END
stRpcSample is STStrpcsample

stRpcSample:rOp1 = 2
stRpcSample:sStr1 = "aura"
stRpcSample:sOp2 = 3
stRpcSample:sStr2 = "test"
stRpcSample:rOp3 = 4

HOpenConnection(cnx)

IF NOT ASPgmCall("PGM/RPCSAMPLE", "", cnx, stRpcSample) THEN
    Error(ErrFullDetails)
ELSE
    Info("OP1=" + stRpcSample.rOp1 + " - STR1=" + stRpcSample.sStr1 + " - OP2=" +
        stRpcSample.sOp2 + " - STR2=" + stRpcSample.sStr2 + " - STR3=" +
        stRpcSample.rOp3)
END
    
```

## Example 2

In this example, the RPCSAMPLE program is called, using a PCML description.

sPCML est une chaîne

HouvreConnexion(MaConnexionpower8)

```

sPCML = [
    <pcml version="4.0">
        <program name="RPCSAMPLE" path="/QSYS.lib/EASYCOMXMP.lib/RPCSAMPLE.pgm" >
            <data name="OP1" type="packed" length="5" precision="2" usage="input" />
            <data name="STR1" type="char" length="20" usage="input" />
            <data name="OP2" type="packed" length="5" precision="2" usage="inputoutput" />
            <data name="STR2" type="char" length="30" usage="inputoutput" />
            <data name="OP3" type="packed" length="10" precision="4" usage="output" />
        </program>
    </pcml>
]
    
```

STStrpcsample est Structure

```

OP1 est réel
STR1 est chaîne sur 20
OP2 est réel
STR2 est chaîne sur 30
OP3 est réel
    
```

FIN

```

stRpcSample est STStrpcsample
stRpcSample.OP1 = 2
stRpcSample.STR1="aura"
stRpcSample.OP2 = 3
stRpcSample.STR2 = "test"
stRpcSample.OP3 = 4
    
```

```

SI PAS ASPgmCall(sPCML, "", MaConnexionpower8, stRpcSample) ALORS
    Info(ErreurInfo())
SINON
    Info("OP1="+ stRpcSample.OP1 + " - STR1=" + stRpcSample.STR1 + " - OP2=" +
stRpcSample.OP2 + " - STR2=" + stRpcSample.STR2 + " - STR3=" + stRpcSample.OP3)

FIN
    
```

### Example 3

In this example, we call the following RPG program using nested structures:

```

D DS_A      DS
D MBR1      10a
D MBR2      8p 2
D MBR3      8p 2

D DS_P      DS      qualified
D MBRD      likeds(ds_A)

D PARM1     DS      likeds(ds_P)

D PARMT     S      10a

D PARMA     S      8p 2 Dim(3)
D PARM1     S      8p 2

C *entry    plist
C          PARM      PARM1
C          PARM      PARMT
C          PARM      PARMA
C          PARM      PARM1

/free

PARM1.MBRD.MBR2 = PARM1.MBRD.MBR3;
PARMT = 'azeOK';

PARMA(1) = 1 + PARMA(1);

PARMA(2) = 2 + PARMA(2);
PARMA(3) = 3 + PARMA(3);

PARMI = PARM1 + 1;
return;

/end-free
    
```

From Windev 27, we can directly send a complex Windev structure (Windev structures containing other Windev structures), thus faithfully representing the parameters of their programs (structure description and/or

program of a PCML) even the most complicated, and this without mapping, splitting, into a single variable which will be filled/updated directly by Easycom.

```
sPCML est une chaîne
sPCML = [
<pcml version="6.0">
  <struct name="DS_A">
    <data name="MBR1" type="char" length="10" usage="inherit" />
    <data name="MBR2" type="packed" length="8" precision="2" usage="inherit" />
    <data name="MBR3" type="packed" length="8" precision="2" usage="inherit" />
  </struct>
  <!-- 7 -->
  <struct name="DS_P">
    <data name="MBRD" type="struct" struct="DS_A" usage="inherit" />
  </struct>
  <program name="SAMPLEDS6" entrypoint="SAMPLEDS6">
    <data name="PARM1" type="struct" struct="DS_P" usage="inputoutput" />
    <data name="PARMT" type="char" length="10" usage="inputoutput" />
    <data name="PARMA" type="packed" length="8" precision="2"
usage="inputoutput" count="3" />
    <data name="PARMI" type="packed" length="8" precision="2"
usage="inputoutput" />
  </program>
</pcml>
]
```

```
DS_A est Structure
  mbr1 est une chaîne sur 10
  mbr2 est un réel
  mbr3 est un réel
```

```
FIN
sDS_A est DS_A
```

```
DS_P est Structure
  mbrd est une DS_A
```

```
FIN
sDS_P est DS_P
```

```
sDS_A:mbr1 = "aura"
sDS_A:mbr2 = 1
sDS_A:mbr3 = 3
```

```
sDS_P.mbrd = sDS_A
PARMA est un tableau de 3 réels
PARMA[1] = 10
PARMA[2] = 20
PARMA[3] = 30
PARMT est une chaîne
PARMT = "qsd"
PARMI est un réel
PARMI = 2;
```

```
completeStruc est une Structure
  PARM1 est DS_P
  PARMT est une chaîne
  PARMA est un tableau de 3 réels
  PARM1 est un réel
```

```
FIN
```

```
SAMPLEDS6 est completeStruc
SAMPLEDS6.PARM1 = sDS_P
SAMPLEDS6.PARMT = PARMT
SAMPLEDS6.PARMA = PARMA
```

```

SAMPLEDS6.PARMI = PARMI

HOuvreConnexion(MaConnexion)

ASExec("ADDLIBLE EASYCOMXM3",MaConnexion)

// New way, compatible with complex and nested structures

SI PAS ASPgmCall(sPCML, "SAMPLEDS6",MaConnexion, SAMPLEDS6) ALORS
    Info(ErreurInfo(errComplet))
    Info("PARM1.MBRD.MBR2=" + sDS_P:mbrd:mbr2)
SINON
    Trace("PARM1.MBRD.MBR2=" + SAMPLEDS6:PARM1:mbrd:mbr2)
    Trace("PARM1.MBRD.MBR3=" + SAMPLEDS6:PARM1:mbrd:mbr3)
    Trace("PARMT=" + SAMPLEDS6:PARMT)
    Trace("PARMA[1]=" + SAMPLEDS6:PARMA[1])
    Trace("PARMA[2]=" + SAMPLEDS6:PARMA[2])
    Trace("PARMA[3]=" + SAMPLEDS6:PARMA[3])
    Trace("PARMI=" + SAMPLEDS6:PARMI)
FIN
    
```

Before Windev 27, the program was called as follows:

```

SI PAS ASPgmCall(sPCML, "SAMPLEDS6",MaConnexion, sDS_P, PARMT, PARMA, PARMI) ALORS
    Info(ErreurInfo(errComplet))
    Info("PARM1.MBRD.MBR2=" + sDS_P:mbrd:mbr2)
SINON
    Trace("sDS_P.MBRD.MBR2=" + sDS_P:mbrd:mbr2)
    Trace("sDS_P.MBRD.MBR3=" + sDS_P:mbrd:mbr3)
    Trace("PARMT=" + PARMT)
    Trace("PARMA[1]=" + PARMA[1])
    Trace("PARMA[2]=" + PARMA[2])
    Trace("PARMA[3]=" + PARMA[3])
    Trace("PARMI=" + PARMI)
FIN
    
```

## ASProcedureCall

ASProcedureCall is for calling ILE procedures of a service program (an alternative to ASRunRPC).

The usage is like ASPgmCall, with only difference of an additional parameter for the procedure name.

## Syntax

```

Result = ASProcedureCall(ProcedureDescription, ServiceProgramName, ProcedureName
[, Connection], Param1 [, Param2 [, Param3, ... ]])
    
```

In French : [ASAppelProcedure](#)

## Parameters

### Result

Boolean: True if function succeed, False otherwise.

### ProcedureDescription

Character string:

- When the program to call was described using using [RPC-DTAQ Description utility](#), this parameter has the form: **\*PGM/NAME**. where NAME is the name of the description given in RPC/DTAQ Configuration tool.
- When the program is described in **PCML** syntax in a file, this parameter contains an **IFS Path**.
- When the program is described in **PCML** syntax in the WinDev program, this parameter contains the **PCML description** itself.

#### ServiceProgramName

The name of the service program, this name can be qualified (library/name). If the library is not specified, \* LIBL is used.

#### ProcedureName

String - is the actual name of the procedure, or empty string if the current name is to be used via DescProcedure. Is ignored if DescProcedure is of the form \* PGM / XXX.**Connexion [optionnel]**

Connexion - nom de la connexion

#### Connection [optional]

Connection to AS/400.

#### Param1, Param2 , ...

Variable or values corresponding to the parameters.

## Examples

### Example 1 :

We call the RPCSAMPLE program. Parameters passed to the program are mapped to a WinDev structure:

```

STStrpcsample est Structure
  rOp1 est réel
  sStr1 est chaîne sur 20
  sOp2 est réel
  sStr2 est chaîne sur 30
  rOp3 est réel
FIN
stRpcSample est STStrpcsample

stRpcSample:rOp1 = 2
stRpcSample:sStr1 = "aura"
stRpcSample:sOp2 = 3
stRpcSample:sStr2 = "test"
stRpcSample:rOp3 = 4

HouvreConnexion(MaConnexionpower8)

SI PAS ASAppelProcedure("*PGM/RPCSAMPLE", "", "",MaConnexionpower8, stRpcSample)
ALORS
  Info(ErreurInfo())
SINON
  Info("OP1="+ stRpcSample.rOp1 + " - STR1=" + stRpcSample.sStr1 + " - OP2=" +
  stRpcSample.sOp2 + " - STR2=" + stRpcSample.sStr2 + " - OP3=" +
  stRpcSample.rOp3)
FIN
    
```

### Example 2 :

We call the RPCSAMPLE program. Parameters passed to the program are mapped to a WinDev structure. We use also a service program :



```

STStrpcsample est Structure
    rOp1 est réel
    sStr1 est chaîne sur 20
    sOp2 est réel
    sStr2 est chaîne sur 30
    rOp3 est réel
FIN
stRpcSample est STStrpcsample

stRpcSample:rOp1 = 2
stRpcSample:sStr1 = "aura"
stRpcSample:sOp2 = 3
stRpcSample:sStr2 = "test"
stRpcSample:rOp3 = 4

sSrvPgm est une chaîne

sSrvPgm = "EASYCOMXMP/XMPSRVPGM"

HOuvreConnexion(MaConnexionpower8)

SI PAS ASAppelProcedure("*PGM/RPCSAMPLE", sSrvPgm, "", MaConnexionpower8,
stRpcSample) ALORS
    Info(ErreurInfo())
SINON
    Info("OP1="+ stRpcSample.rOp1 + " - STR1=" + stRpcSample.sStr1 + " - OP2=" +
stRpcSample.sOp2 + " - STR2=" + stRpcSample.sStr2 + " - STR3=" + stRpcSample.rOp3)
FIN
    
```

## IFS functions

### *ASfLoadText - ASfLoadBinary*

Those functions allow to load a whole IFS file into a WinDev variable.

### Syntax

```

sData = ASfLoadText(FileName [,Connection])
sData = ASfLoadBinary(FileName [,Connection])
    
```

In French: [ASfChargeTexte](#), [ASfChargeBinaire](#)

### Parameters

#### **FileName**

String which contains the IFS file path.

#### **Connection (optional)**

Connection to AS/400.



## Detail

ASfLoadText/Binary functions are reading the complete file into a variable. The difference between the two functions is that ASfLoadText is performing conversion, but ASfLoadBinary isn't.

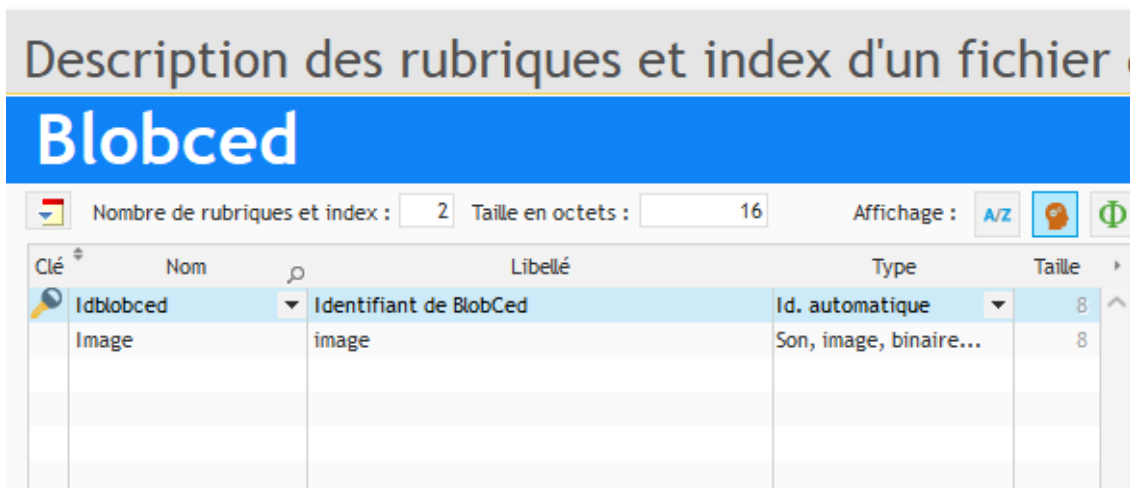
## Example

```
bufImg is Buffer
bufImg = ASfLoadBinary("/images/smt3.jpg")

IF bufImg="" THEN
  Info(Info())
ELSE
  IMG_Image1 = bufImg
END
```

## Complete example

Into this example, we use the following table:



Clé	Nom	Libellé	Type	Taille
Idblobced	Identifiant de BlobCed	Id. automatique		8
	Image	image	Son, image, binaire...	8

An image found on the IFS is loaded into a buffer with [ASfLoadBinary](#), and then a new record is inserted into the table with [HAjoute](#).

The last recording is then read, and the image saved on the PC with [HExtractMemo](#).

An image on the IFS is loaded into a buffer with the [ASfLoadBinary](#) ([ASfChargeBinaire](#)) function, and then a new record is inserted into the table with the [HAjoute](#) function.

The last recording is then read, and the image saved on the PC with [HExtractMemo](#).

```
id est un entier
bufIn est un Buffer

HOuvreConnexion(MaConnexionpower8)

QUAND EXCEPTION DANS
  zsChemin est une chaîne
  zsChemin = "/tmp/Photo.gif"
  bufIn = ASfChargeBinaire(zsChemin, MaConnexionpower8)

  Blobced.Image = bufIn
```

```

SI HAjoute(Blobced) ALORS
    Info("ajoute done!")
SINON
    Info(ErreurInfo())
FIN

// retrieve ID value of the last record
SI HLitDernier(Blobced, Idblobced) ALORS
    Info(Blobced.Idblobced)
    id = Blobced.Idblobced

SINON
    Info("Fichier non trouvé " + ErreurInfo())
FIN

FAIRE
    Info("exception!")
    Info(ExceptionInfo())
FIN

////////////////////////////////////
// Read last record
////////////////////////////////////
QUAND EXCEPTION DANS
    Info(id)
    HRAZ(Blobced, Idblobced)
    SI HLitRecherchePremier(Blobced, Idblobced, id)
        zsCheminc est une chaîne
        zsCheminc = "C:\temp\Photo6.gif"
        // Extrait le document sur le disque
        SI HExtraitMémo(Blobced, Image, zsCheminc) ALORS
            Info("extrait!")
        SINON
            Info(ErreurInfo())
        FIN
    SINON
        Info("Fichier non trouvé " + ErreurInfo())
    FIN
FAIRE
    Info("exception!")
    Info(ExceptionInfo())
FIN

HFermeConnexion(MaConnexionpower8)
    
```

### ASfCreate

Opens an IFS file in creation mode. (ASfOpen also allows this using the mode parameter)

### Syntax

```
result = ASfCreate (FileName [,mode] [,Connection])
```

In French: [ASfCrée](#)

## Parameters

**result: integer**

Number of the opened file (handle). This value must be given to other functions: ASfClose, ASfRead, ASfWrite and so on.

Result is -1 in case of error

**mode: optional integer**

[Creation mode](#) of the file.

The default value is:

[ASfoCreate+ASfoTruncate+ASfoWrite+ASfoModeU\\_RW+ASfoModeG\\_RW](#)

**connection (optional)**

Connection to AS/400.

## Example

```
nres =
ASfCreate("/text/unicode_text_to_create.txt", ASfoUnicode+ASfoTruncate)
IF nres = -1 THEN
  Error(ErrorInfo())
RETURN
END
```

See [example](#).

## ASfWrite

Write into an IFS file.

### Syntax 1

```
Result = ASfWrite(open_id, data1, length)
```

### Syntax 2

```
Result = ASfWrite(open_id, data2)
```

In French : [ASfEcrit](#)

## Parameters

**result: integer**

Number of bytes or characters written.

**open\_id: integer**

Integer provided by ASfCreate or ASfOpen.

**data1: Buffer or string**

Data to write corresponding to the Length parameter.

**data2: string**

Data to write.

**length:**

For the syntax 1, size of the buffer or the string to write. For Unicode string, Length has to be equal to the number of characters.

See [example](#).

### *ASfWriteLine*

Writes a line in an IFS File (writes the text and a new line character).

#### Syntax

```
Result = ASfWriteLine(open_id, text [,length])
```

In French: [ASfEcritLigne](#)

#### Parameters

**result:** integer

Number of actually written characters. -1 in case of error.

**open\_id:** integer

Value provided by ASfCreate or ASfOpen.

**text:** ANSI or Unicode string

Text to write.

**length**

(optional) Number of characters to write. Is useful if a 0 character is present in the string.

See [example](#).

### *ASfClose*

This closes the IFS file. The pending data (if any) is written at this time.

#### Syntax

```
Result = ASfClose(open_id)
```

In French: [ASfFerme](#)

#### Parameters

**result:** boolean

False in case of error.

**open\_id:** integer

Value provided by ASfCreate or ASfOpen.

See [example](#).

### *ASfRead*

Reads into an IFS file.

#### Syntax 1 (recommended)

```
Result1 = ASfRead(open_id, length, data)
```

#### Syntax 2

```
Result2 = ASfRead(open_id, length)
```



In French: [ASfLit](#)

## Parameters

**Result1:** integer

Number of bytes or characters actually read.

**Result2:** Buffer

Result data.

**open\_id:** integer

Value provided by ASfCreate or ASfOpen.

**data:** buffer or string (allocated).

Data space for receiving result.

**length:** integer

Number of bytes or characters to read

The syntax 1 is better because it translates the data into the provided datatype (from or to Unicode if necessary), whenever the open mode.

The number given for <length> and of the result is the "natural" length, which will be:

For syntax 1 :

- several bytes if the the 3<sup>rd</sup> parameter datatype is binary or text
- several characters if the 3<sup>rd</sup> parameter datatype is Unicode

For syntax 2 :

- several bytes if the open mode is binary or text
- several Unicode characters if the open mode is Unicode

## Example

This example is downloading a \*SAVF file into a local directory, from MY\_LIB/SAVF to c:\temp\mysavf.savf.

```

nHdl, cnt is int
buf is Buffer
nLocalFile is int

nHdl = ASfOpen("/QSYS.LIB/MY_LIB.LIB/MYSAVF.FILE", ASfoBinary)
IF nHdl = -1 THEN
    Error(ErrorInfo())
    RETURN
END

nLocalFile = fopen("c:\temp\mysavf.savf", foWrite+foCreate)
IF nLocalFile = -1 THEN
    Error(ErrorInfo())
    RETURN
END

cnt = ASfRead(nHdl, 65000, buf)
WHILE cnt<>0
    fwrite(nLocalFile, buf, cnt)
    cnt = ASfRead(nHdl, 65000, buf)
END

fclose(nLocalFile)
ASfclose(nHdl)
    
```

See [example](#).

## ASfReadLine

Reads a text line in an IFS file.

### Syntax

```
Result = ASfReadLine(open_id)
```

In French: [ASfLitLigne](#)

### Parameters

**result:** Ansi or Unicode string

The read line. The datatype depends on the open mode.

**open\_id:** integer

Value provided by ASfCreate or ASfOpen

See [example](#).

## ASfOpen

Opens an IFS file, and returns an handle to use it with other functions: ASfRead, ASfReadLine, ...

### Syntax

```
result = ASfOpen(FileName [,mode] [,Connection])
```

In French: [ASfOuvre](#)

### Parameters

**result: integer**

Number of the opened file (handle). This value must be given to other functions: ASfClose, ASfRead, ASfWrite and so on.

Result is -1 in case of error.

**mode: optional integer**

[Open mode](#) of the file

The default value is: *ASfoRead*

**Connection (optional)**

Connection to AS/400.

### Example

```
nHdl = ASfOpen("/textes/text_to_read.txt", ASfoUnicode)
IF nHdl = -1 THEN
  Error(Info())
RETURN
END
```

## ASfSaveText/ASfSaveBinary

Those functions are useful for writing a whole IFS file directly from a string or a buffer.



## Syntax

```
result = ASfSaveText(FileName, data [,mode] [,Connection])
result = ASfSaveBinary(FileName, data [,mode] [,Connection])
```

In French: [ASfSauveTexte](#), [ASfSauveBinaire](#)

## Parameters

**Result:** Boolean

Is false in case of error.

**FileName**

Strings which contains the IFS file path.

**Data:** Ansi or Unicode string, or buffer

Variable which contain all the data to put in the file.

**Mode:** optional integer

[Creation mode](#) of the file

The default value is:

[ASfoCreate](#)+[ASfoTruncate](#)+[ASfoWrite](#)+[ASfoModeU\\_RW](#)+[ASfoModeG\\_RW](#)

**Connection (optional)**

Connection to AS/400.

## Example

`buf` is Buffer

```
IF NOT ASfSaveText("/tmp/data.txt", buf, MyConnection) THEN
  Info(Info())
END
```

## Mode for ASfOpen, ASfCreate, ASfSaveText/Binary

This mode values is used to specify how to open or create an IFS file using ASfOpen, ASfCreate, ASfSaveText/Binary.

This mode is a combination (i.e., a sum) of the following constants:

- Open kind
  - [ASfoAdd](#) : add data from end of the existing file
  - [ASfoTruncate](#) truncates the file
  - [ASfoCreate](#) creates the file, which must not exist before
  - [ASfoCreateIfNotExist](#) creates the file if does not exist before, use existing file otherwise
- Usage
  - [ASfoWrite](#) write access
  - [ASfoRead](#) read access
  - [ASfoReadWrite](#) read and write access
  - [ASfoBinary](#) binary : no conversion
  - [ASfoUnicode](#) file is storing Unicode data
- Locking
  - [ASfofoReadLock](#) read locking
  - [ASfoWriteLock](#) write locking
- Rights. This is unix-like rights: read (r), write (w), execute(x) for user(u), group(g) and other(o). This are the same symbolic which are used by the Unix `chmod` command.
  - When creating a file, the object's owner will always be the current user.

CHGAUT command allows adjusting user rights. CHGOWN allows changing object's owner.

```

ASfoModeU_R the object's owner has read access
ASfoModeU_W the object's owner has written access
ASfoModeU_X the object's owner has execution access
ASfoModeU_RW = ASfoModeU_R+ASfoModeU_W
ASfoModeU_RX = ASfoModeU_R+ASfoModeU_X
ASfoModeU_RWX = ASfoModeU_RW+ASfoModeU_X
ASfoModeG_R the object's group has read access
ASfoModeG_W the object's group has written access
ASfoModeG_X the object's group has execution access
ASfoModeG_RW = ASfoModeG_R+ASfoModeG_W
ASfoModeG_RX = ASfoModeG_R+ASfoModeG_X
ASfoModeG_RWX = ASfoModeG_RW+ASfoModeG_X
ASfoModeO_R other users have read access
ASfoModeO_W other users have written access
ASfoModeO_X other users have execution access
ASfoDroitsO_RW = ASfoModeO_R+ASfoModeO_W
ASfoDroitsO_RX = ASfoModeO_R+ASfoModeO_X
ASfoDroitsO_RWX = ASfoDroitsO_RW+ASfoModeO_X
ASfoModeA_R = ASfoModeU_R+ASfoModeG_R+ASfoModeO_R
ASfoModeA_W = ASfoModeU_W+ASfoModeG_W+ASfoModeO_W
ASfoModeA_X = ASfoModeU_X+ASfoModeG_X+ASfoModeO_X
ASfoModeA_RWX = ASfoModeA_R+ASfoModeA_W+ASfoModeA_X
    
```

### Example

Into this example, we create a file on IFS, and we write data into: `ASfCrée`, `ASfEcritLigne` and `ASfFerme`.

After we read the file: `ASfOuvre`, `ASfLitLigne` and `ASfFerme`.

```

nIdOuverture est un entier

nBCarEcritsLigne est un entier
Tmp est une chaîne UNICODE

H0uvreConnexion(MaConnexionpower8)

nIdOuverture =
ASfCrée("/tmp/test_IFS4.txt",ASfCréation+ASfTronque+ASfEcriture+ASfDroitsU
_RW+ASfDroitsG_RW+ASfUnicode)

IF nIdOuverture = -1 ALORS
    Info(ErreurInfo())
    RETOUR
FIN

Tmp = "Write into file on IFS"
nBCarEcritsLigne = ASfEcritLigne(nIdOuverture, Tmp)
Info("Number of characters written = " + nBCarEcritsLigne)

SI PAS ASfFerme(nIdOuverture) ALORS
    Info(ErreurInfo())
FIN

///// Read file on IFS

nIdOuverture = ASfOuvre("/tmp/test_IFS4.txt",ASfUnicode)

IF nIdOuverture = -1 ALORS
    Info(ErreurInfo())
    RETOUR
    
```



```

FIN

sChaineLue est une chaîne UNICODE
sChaineLue = ASfLitLigne(nIdOuverture)
Info(sChaineLue)

SI PAS ASfFerme(nIdOuverture) ALORS
    Info(ErreurInfo())
FIN
    
```

## Profile Management

### ASUserList

List all the AS400 profiles according to the rights of the user opening the connection.

### Syntax

```
Result = ASUserList(ASUsrPrfCollection, Connection)
```

In French: `ASUserList`

### Parameters

#### Result

Boolean – True if successful, False otherwise.

#### ASUsrPrfCollection

A variable of ASUsrPrfCollection type, which will be filled with the result. It is a collection of ASUsrPrf objects.

#### Connection

String – name of the connection.

### Example

```

profilList is ASUsrPrfCollection
myProfil is ASUsrPrf
nb is integer

HOpenConnection(AS400)
ASUserList(profilListe, AS400)

nb = profilList.CollectionASUsrPrf..Occurrence
FOR indice= 1 A nb
    myProfil = profilList.CollectionASUsrPrf[indice]
    TableauAjoute(gtabObjet, myProfil)
FIN
TableAffiche(TABLE_GtabObjet)
    
```

### ASUserListDetailed

*This command is only available with the V7R3 OS on AS400 side.*

Retrieves the list of users visible to the user of the connection according to his rights.

All the properties of the user's profile are available with the ASUsrPrfDetailedCollection object in return of the function.

## Syntax

```
Result = ASUserListDetailed(ASUsrPrfDetailedCollection,Connection, [UserName],  
[UserStatus], [UserGroup], [UserClass],[SpecialAuthority])
```

In French: [ASUserListDetailed](#)

## Parameters

### Result

Boolean – True if successful, False otherwise.

### ASUsrPrfDetailedCollection

An ASUsrPrfDetailedCollection variable that will contain the result of the command.  
This is a collection of objects of type ASUsrPrfDetailed.

### Connection

String – name of the connection.

### UserName

String – Filter on the name of the profiles to recover. For example: \* for all users.

[optional]. If no name is specified, the information of all users will be retrieved (value \* or empty string used).

### UserStatus

Among the following constants:

[ASUsersALLStatus](#) (in French: [ASUtilisateursTousStatuts](#))

[ASDisabledUsers](#) (in French: [ASUtilisateursDésactivés](#))

[ASEnabledUsers](#) (in French: [ASUtilisateursActivés](#))

[optional]. If no value is specified, the [ASUsersALLStatus](#) constant will be used.

### UserGroup

String – Name of the group in which we want to retrieve the profiles corresponding to the search criteria.

[optional]. If no group is specified, the search will take place on all groups (value \* or empty string used).

### UserClass

String – Name of the user class in which we want to retrieve the profiles matching the search criteria.

[optional]. If no user class is specified, the search will take place on all classes (value \* or empty string used).

### SpecialAuthority

Optional. Among the following constants:

[ASALLOBJ](#) : corresponds to \*ALLOBJ

[ASSECADM](#) : corresponds to \*SECADM

[ASJOBCTL](#) : corresponds to \*JOBCTL

[ASSPLCTL](#) : corresponds to \*SPLCTL

[ASSAVSYS](#) : corresponds to \*SAVSYS

[ASSERVICE](#) : corresponds to \*SERVICE

[ASAUDIT](#) : corresponds to \*AUDIT

[ASIOSYSCFG](#) : corresponds to \*IOSYSCFG

Mixing constants is possible:

[ASALLOBJ+ASSECADM+ASJOBCTL+ASSPLCTL+ASSAVSYS+ASSERVICE+ASAUDIT+ASIOSYSCFG](#)



## Notes

1) The following 3 syntaxes are equivalent: we execute the operation on all the groups.

```
ASUserListDetailed(test2, connect,sFiltreName,ASEnabledUsers,"*")
ASUserListDetailed(test2, connect, sFiltreName, ASEnabledUsers,"")
ASUserListDetailed(test2, connect, sFiltreName, ASEnabledUsers)
```

2 The following 3 syntaxes are equivalent: we execute the operation on all the classes for the group "MYGROUP"..

```
ASUserListDetailed(test2, connect, sFiltreName, ASEnabledUsers,"MYGROUP", "*")
ASUserListDetailed(test2, connect, sFiltreName, ASEnabledUsers,"MYGROUP", "")
ASUserListDetailed(test2, connect, sFiltreName, ASEnabledUsers,"MYGROUP")
```

## Example

```
MaConnexion est une Connexion
// Description de la connexion
MaConnexion..Utilisateur = "trsecofr"
MaConnexion..MotDePasse = "trsecofr"
MaConnexion..Serveur = "power8"

MaConnexion..Provider = hAccèsNatifAS400

MaConnexion..InfosEtendues= "<EASYCOM>"+CRLF+"JOBNAME=CED"+CRLF+"</EASYCOM>."

test2 est une ASUsrPrfDetailedCollection

//Retrieve all users
SI PAS ASUserListDetailed(test2,MaConnexion) ALORS
    Info(ErreurInfo(errComplet))
    RETOUR
SINON
    DSPUSR();//procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(test2)
FIN

// Retrieve all users
SI PAS ASUserListDetailed(test2,MaConnexion,"*",ASUtilisateursTousStatuts)
ALORS
    Info(ErreurInfo(errComplet))
    RETOUR
SINON
    DSPUSR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(test2)
FIN

//Retrieve user with the QPGMR name
HOuvreConnexion(MaConnexion);
SI PAS ASUserListDetailed(test2,MaConnexion,"QPGMR") ALORS
    Info(ErreurInfo(errComplet))
    RETOUR
SINON
    DSPUSR();//procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(test2)
FIN
```



```
// Retrieve all disabled users
SI PAS ASUserListDetailed(test2, MaConnexion, "*", ASUtilisateursDésactivés)
ALORS
    Info(ErreurInfo(errComplet))
    RETOUR
SINON
    DSPUSR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(test2)
FIN

// Retrieve all enabled users
SI PAS ASUserListDetailed(test2, MaConnexion, "*", ASUtilisateursActivés) ALORS

    Info(ErreurInfo(errComplet))
    RETOUR
SINON
    DSPUSR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(test2)
FIN

// Retrieve all users with name starting with Q*
SI PAS ASUserListDetailed(test2, MaConnexion, "Q*", ASUtilisateursTousStatuts)
ALORS
    Info(ErreurInfo(errComplet))
    RETOUR
SINON
    DSPUSR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(test2)
FIN

// Retrieve all users within NOGROUP group
SI PAS
ASUserListDetailed(test2, MaConnexion, "", ASUtilisateursTousStatuts, "NOGROUP")
ALORS
    Info(ErreurInfo(errComplet))
    RETOUR
SINON
    DSPUSR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(test2)
FIN

// Retrieve all users within *SECOFR class
SI PAS ASUserListDetailed(test2, MaConnexion, "", ASUtilisateursTousStatuts, "",
"*SECOFR") ALORS
    Info(ErreurInfo(errComplet))
    RETOUR
SINON
    DSPUSR();
    TableauSupprimeTout(test2)
FIN

//Retrieve all users with special Authorities
SI PAS ASUserListDetailed(test2, MaConnexion, "", ASUtilisateursTousStatuts, "",
"", ASJOBCTL+ASIOSYSCFG) ALORS
    Info(ErreurInfo(errComplet))
    RETOUR
SINON
    DSPUSR();
    TableauSupprimeTout(test2)
FIN
```

### ASSignedOnUsersInfoList

Retrieves the list of signed-on users or previously signed-on users with pending printer tasks.

#### Syntax

```
Result = ASSignedOnUsersInfoList (ASSignedOnUsersInfoCollection, Connection,
    [Username], [StationName], [IncludeDisconnectedSuspendedJobs],
    [IncludeSignedOffWithPrintWaiting])
```

In French: [ASSignedOnUsersInfoList](#)

#### Parameters

##### Result

Boolean – True if the operation was performed. False if a problem occurred. ErreurInfo is used to identify the error.

##### ASSignedOnUsersInfoCollection

An [ASignedOnUsersInfoCollection](#) variable that will contain the result of the command. This is a collection of objects of type ASignedOnUsersInfo.

##### Connection

Connection – connection name.

##### Username

Character string (with quotes) – Username of the searched profile(s).

If you don't want to search by name, put "\*\*ALL" or leave empty.

If you want to filter by out by name starting with something, end your string by \* (e.g. "QPG\*").

##### StationName

Character string (with quotes) – StationName researched.

If you don't want to search by station name, put "\*\*ALL" or leave empty.

If you want to filter by out by station name starting with something, end your string by \* (e.g. "QPADEV\*").

##### IncludeDisconnectedSuspendedJobs

Boolean – True if should be included disconnected jobs and suspended group jobs. False otherwise.

##### IncludeSignedOffWithPrintWaiting

Boolean – True if should be included signed-off users with output waiting to print. False otherwise.

#### Example

`assoic` est un `ASSignedOnUsersInfoCollection`

```
ASSignedOnUsersInfoList(assoic,PrinciConnexion,SAI_user,SAI_station,INT_deco,INT_Print)
SAI_Infos = ""
ind est un entier
ind = 1
```

POUR TOUT `assoic` DE `assoic`

```
SAI_Infos = SAI_Infos + RC + ind + ")" + RC +
"Username : " + assoic.userName + RC +
"User Description : " + assoic.userDesc + RC +
"Activity : " + assoic.activity + RC +
"Activity Name : " + assoic.activityName + RC +
"Station Name : " + assoic.stationName + RC +
"Station Desc : " + assoic.displayStationDesc + RC +
"Disconnected Job Allowed : " + assoic.disconnectJobAllowed + RC +
```



```
    "Job Number : " + assoi.jobNumber + RC  
    ind++  
FIN
```

### ASCreateProfil

Create a new AS400 profile. It is necessary that the profile used to open the connection, has the necessary rights for the creation of a profile.

Indeed, a profile of type \* SECADM (like QSECOFR for example) is obligatory for this function.

### Syntax

```
Result = ASCreateProfil(ProfileName,Password, Description, [Source  
Profile],Connection)
```

In French: [ASCreerProfil](#)

### Parameters

#### Result

Boolean – True if the command was successful, false otherwise.

#### NomProfil

String – Name of the new AS400 profile

#### Mot de passe

String – Password for the new profile

#### Description

String – Description of the profile.

#### Profil Source

String – Name of the profile to copy settings from, this is an optional parameter.

#### Connection

String – Name of the connection

### Example

```
MaConnexion est une Connexion  
MaConnexion..Utilisateur = "TSECOFR"  
MaConnexion..MotDePasse = "tsecofr"  
MaConnexion..Serveur = "power8"  
  
MaConnexion..Provider = hAccèsNatifAS400  
HouvreConnexion(MaConnexion)  
  
SI PAS ASCreerProfil("CRTEST", "crtest", "Test", "QPGMR",MaConnexion) ALORS  
    Info(HErreur())  
SINON  
    Info("Profil créé")  
FIN
```

### ASModifyProfil

Modify an AS400 profile. It is necessary that the profile used to open the connection, has the necessary rights for the modification of a profile.



Indeed, a profile of type \* SECADM (like QSECOFR for example) is obligatory for this function.

## Syntax

```
Result = ASModifyProfil(ProfilName,Field,Value,Connection)
```

In French: [ASModifieProfil](#)

## Parameters

### Result

Boolean - True if the command was successful, false otherwise.

### ProfilName

String – Name of the AS400 profile to modify.

### Field

String – Field of the profile to modify, it can be one of the following values :  
PASSWORD,PWDEXP,STATUS,USRCLS,ASTLVL,CURLIB,INLPGM,INLMNU,LMTCPB,TEXT,SPCAUT,SPCENV,DSPSGNINF,PWDEXPITV,LCLPMDMGT,LMTDEVSSN,KBDBUF,MAXSTG,PTYLMT,JOBD,GRPPRF,OWNER,GRPAUT,GRPAUTYP,SUPGRPPRF,ACGCDE,DOCPWD,MSGQ,DLVRY,SEV,PRTDEV,OUTQ,ATNPGM,SRTSEQ,LANGID,CNTRYID,CCSID,CHRIDCTL,SETJOBATR,LOCALE,USROPT,UID,GID,HOMEDIR,EIMASSOC.

### Value

String – New value for the field.

### Connection

String – Name of the connection.

## Example

```
MaConnexion est une Connexion
MaConnexion..Utilisateur = "TSECOFR"
MaConnexion..MotDePasse = "tsecofr"
MaConnexion..Serveur = "power8"

MaConnexion..Provider = hAccèsNatifAS400
HouvreConnexion(MaConnexion)

SI PAS ASModifieProfil("CRTEST", "PASSWORD", "aura",MaConnexion) ALORS
    Info(HErreur())
SINON
    Info("Profil modifié")
FIN
```

## ASDeleteProfil

Delete an AS400 profile. The profile used to open the connection must have the necessary rights to delete a profile.

Indeed, a profile of type \* SECADM (like QSECOFR for example) is obligatory for this function.

## Syntax

```
Result = ASDeleteProfil(NomProfil,Connection)
```

In French: [ASEffaceProfil](#)



## Parameters

### Result

Boolean - True if the command was successful, false otherwise.

### NomProfil

String – Name of the AS400 profile to delete.

### Connection

String – Name of the connection.

## Example

```
MaConnexion est une Connexion
MaConnexion..Utilisateur = "TSECOFR"
MaConnexion..MotDePasse = "tsecofr"
MaConnexion..Serveur = "power8"

MaConnexion..Provider = hAccèsNatifAS400
HOuvreConnexion(MaConnexion)

SI PAS ASEffaceProfil("CRTEST",MaConnexion) ALORS
    Info(HErreur())
SINON
    Info("Profil effacé")
FIN
```

## *ASCheckPassword*

Check if the password given as a parameter is correct.

## Syntax

```
Result = ASCheckPassword(ProfileName,Password,[CCSID],Connection)
```

In French : [ASVerifieMdP](#)

## Parameters

### Result

Boolean – True if the password is correct, false otherwise.

### ProfileName

String – Name of the AS400 profile

### Password

String – Password to check.

### CCSID

Integer – Optional parameter, CCSID of the password if different of the system's one.

### Connection

String – Name of the Connection



## Example

```
MaConnexion est une Connexion
MaConnexion..Utilisateur = "TSECOFR"
MaConnexion..MotDePasse = "tsecofr"
MaConnexion..Serveur = "power8"

MaConnexion..Provider = hAccèsNatifAS400
HOuvreConnexion(MaConnexion)

SI PAS ASVerifieMdp("CRTEST", "aura", MaConnexion) ALORS
    Info("Mot de passe incorrect")
SINON
    Info("Mot de passe correct")
FIN
```

## ASCheckAuthority

Check if an AS400 profile has a given authority on an object.

## Syntax

```
Result =
ASCheckAuthority(ProfileName, Object, Library, ObjectType, Authority, Connection)
```

In French: [ASVerifieDroits](#)

## Parameters

### Result

Boolean – True if the profile has the requested authority, false otherwise.

### ProfileName

String – Name of the profile.

### Object

String – Name of the object on which we're checking the authority.

### Library

String – Library of the object.

### ObjectType

String – Type of the object.

### Authority

String – Authority to check, the following values are allowed:

\*EXCLUDE,\*ALL,\*CHANGE,\*USE,\*AUTLMGT,OBJALTER,\*OBJOPR,\*OBJMGT,\*OBJEXIST,\*OBJREF,\*READ,\*ADD,\*UPD,\*DLT,\*EXECUTE

### Connection

String – Name of the Connection

## Example

```
MaConnexion est une Connexion
MaConnexion..Utilisateur = "TSECOFR"
```



```
MaConnexion..MotDePasse = "tsecofr"  
MaConnexion..Serveur = "power8"  
MaConnexion..Provider = hAccèsNatifAS400  
HouvreConnexion(MaConnexion)  
  
SI PAS ASVerifieDroits("CRTEST", "S_EMPLOYEE", "EASYCOMXMP", "*FILE", "*ALL",  
MaConnexion) ALORS  
    Info("Le profil n'a pas les droits demandés sur l'objet")  
SINON  
    Info("Droits accordés")  
FIN
```

## ASChangePasswd

Change the password for the current user.

### Syntax

```
Result = ASChangePasswd (OldPassword, NewPassword, Connection)
```

In French: [ASChangeMdp](#)

### Parameters

#### Result

Boolean – True if the command is successful.

#### OldPassword

Old password for the current user.

#### NewPassword

New password for the current user.

#### Connection

Connection – name of the connection.

## ASUser

With this function, the program gives to the current connection the authority of another user.

With this function, the real user who sign on, uses his user profile and password to connect, using function [HOpenConnection](#). Then, the connection job adopts the authority of another user known by the program only.

See also EACTCP003 exit program to secure access to Easycom server.

### Syntax

```
Result = ASUser (New_Profile, Password [, Connection])
```

In French: [ASUtilisateur](#)

### Parameters

#### Result

Boolean: True if function succeed, and job is running under new user authority.

#### New\_Profile

User profile: The connection job will adopt the user authority.

#### Password



Password for the new user profile.

**Connection [optional]**

Connection to AS/400.

**Example**

```
HopenConnection(MyConnecton, User, Pwd, "AS400", hNativeAccessAS400,
hOReadWrite, "")
```

```
...
ASUser(SuperID, SuperPWD)
```

User and Pwd are typed by the real user. They validate the user login on the system.

By providing SuperID and SuperPWD with function ASUser(), the program increases the authority of the job on the database. Even is the real user has no authority on the data, the application will be able to access files.

**Object of type AsUsrPrfDetailedCollection**

This type of object is used in return of the **ASUserListDetailed** function. This is a collection of **AsUsrPrfDetailed** objects.

**Properties of AsUsrPrfDetailed**

French name of the property	English name of the property	Type
NomUtilisateur	UserName	string
Statut	Status	string
DateDernierLoginValide	LastSuccessfulSignOn	DateTime
NombreTentativesLoginInfructueuses	NumberFailedLoginAttempts	integer 4 bytes
DateDernierChangementDeMotDePasse	LastPasswordChangeDate	DateTime
IndicateurMDPestNONE	NONEPasswordIndicator	boolean
IntervalValiditéMDP	PasswordIntervalValidity	string
DateExpirationMDP	PasswordExpirationDate	DateTime
NbJoursAvantExpirationMDP	NbDayBeforePasswordExpires	integer 4 bytes
IndicateurExpirationMDP	PasswordExpirationIndicator	boolean
ClasseUtilisateur	UserClass	string
aAutoritéSpéciale	hasSpecialAuthority	boolean
estALLOBJ	isALLOBJ	boolean
estSECADM	isSECADM	boolean
estJOBCTL	isJOBCTL	boolean
estSPLCTL	isSPLCTL	boolean
estSAVSYS	isSAVSYS	boolean
estAudit	isAUDIT	boolean
estSERVICE	isSERVICE	boolean
estIOSYSCFG	ISIOSYSCFG	boolean
Groupe	Group Profile Name	string
Propriétaire	Owner Name	string
AutoritéDuGroupe	GroupAuthority	string
NiveauAssistance	AssistanceLevel	string



NomCURLIB	CurrentLIBName	string
nomMenuInitial	initialMenuName	string
nomLibMenuInitial	initialMenuLibName	string
nomProgrammeInitial	initialProgramName	string
nomLibProgrammeInitial	initialProgramLibName	string
limitationCapacité	capabilitiesLimitation	string
description	description	string
afficheSignOnInfo	displaySignOnInfo	string
sessionLimitéeAUnPoste	limitDeviceSessions	string
mémoireTamponFrappe	keyboardBuffering	string
maxCapacitéMémoirePermise	maxAllowedStorage	integer 4 bytes
mémoireUtilisée	storageUsed	integer 4 bytes
niveauPrioritéMax	highestSchedulingPriority	integer 4 bytes
nomDescriptionJob	jobDescriptionName	string
nomLibDescriptionJob	jobDescriptionLibName	string
codeComptabilité	accountingCode	string
nomFileDAttenteMessages	messageQueueName	string
nomLibFileDAttenteMessages	messageQueueLibName	string
méthodeReceptionMessage	messageQueueDeliveryMethod	string
gravitéMessageFileAttente	messageQueueSeverity	integer 4 bytes
nomFileAttenteSortie	outputQueueName	string
nomLibFileAttenteSortie	outputQueueLibName	string
imprimante	printer	string
environnementSpécial	specialEnvironment	string
nomProgrammeATTN	ATTNProgramName	string
nomLibProgrammeATTN	ATTNProgramLibName	string
idLangage	languageID	string
codePaysOuRégion	countryOrRegionID	string
ccsid	ccsid	string
optionsUtilisateur	userOptions	string
estCLKWD	isCLKWD	boolean
estEXPERT	isEXPERT	boolean
estHLPFULL	isHLPFULL	boolean
estSTSMSG	isSTSMSG	boolean
estNOSTSMSG	isNOSTSMSG	boolean
estROLLKEYS	isROLLKEY	boolean
estPRTMSG	isPRTMSG	boolean
nomSéquenceDeTri	SortSequenceTableName	string
libSéquenceDeTri	SortSequenceTableLibName	string
valeursAuditObjet	ObjectAuditingValue	string
valeursAuditActions	UserActionAuditLevel	string
aCMD	hasCMD	boolean
aCREATE	hasCREATE	boolean

aDELETE	hasDELETE	boolean
aJOBDTA	hasJOBDTA	boolean
aOBJMGT	hasOBJMGT	boolean
aOFCSRV	hasOFCSRV	boolean
aOPTICAL	hasOPTICAL	boolean
aPGMADP	hasPGMADP	boolean
aSAVRST	hasSAVRST	boolean
aSECURITY	hasSECURITY	boolean
aSERVICE	hasSERVICE	boolean
aSPLFDTA	hasSPLFDTA	boolean
aSYSMGT	hasSYSMGT	boolean
typeAutoritéGroupe	GroupAuthorityType	string
nbGroupesSupplémentaires	nbSupplementalGroups	integer 4 bytes
idUtilisateur	userId	unsigned integer 4 bytes
idGroupe	groupId	unsigned integer 4 bytes
aAucunAttributTravailEnvLocal	hasNoneLocaleJobAttributes	string
aSYSVAL	hasSYSVAL	boolean
aCCSID	hasCCSID	boolean
aDATFMT	hasDATFMT	boolean
aDATSEP	hasDATSEP	boolean
aSRTSEQ	hasSRTSEQ	boolean
aTIMSEP	hasTIMSEP	boolean
aDECfmt	hasDECfmt	boolean
estUnGroupeNonVide	isNotEmptyGroup	integer 4 bytes
aAuMoinsUnCertificat	hasAtLeastOneDigitalCertificate	boolean
chrid	chrid	string
nbASPDesc	nbASPStorageDescriptor	integer 4 bytes
gestionMdpLocale	localPasswordManagement	boolean
bloqueChangeMdp	blockPasswordChange	string
estRequisUE	isUERequired	boolean
intervalExpirationUtilisateur	userExpirationInterval	integer 4 bytes
DateExpirationUtilisateur	dateUserExpiration	DateTime
actionExpirationUtilisateur	userExpirationAction	string
maxCapacitéMémoirePermiseLong	maxAllowedStorageLong	integer 8 bytes
mémoireUtiliséeLong	storageUsedLong	unsigned integer 8 bytes

### Object of type *AsUsrPrfCollection*

This type of object is used in return of the [ASUserList](#) function. This is a collection of **AsUsrPrf** objects.

### Properties of **AsUsrPrf**

Property name	Type
UserName	string

GroupName	string
Text	string
GroupNumber	integer
isGroup	integer
hasGroupMembers	integer

### Object of type *ASignedOnUsersInfoCollection*

This type of object is used in return of the `ASSignedOnUsersInfoList` function. This is a collection of `ASignedOnUsersInfo` objects.

### Properties of *ASignedOnUsersInfo*

Property name	Description	Type
userName	User name	string
userDesc	User description	string
activity	Activity	string
activityName	Activity name	string
stationName	Display station name	string
displayStationDesc	Display station description	string
disconnectJobAllowed	Disconnect job allowed indicator. 1 : The job is allowed to be disconnected. 0 : The job is not allowed to be disconnected.	string
jobNumber	JOB number	string

## Printer management

### *ASPrinterList*

Retrieves the list of printers registered on the AS400 and their properties, according to the rights of the user who opened the connection.

### Syntax

```
Result = ASPrinterList(ASPrinterCollection, Connection, [NamePrinter],
[NameOutputQueue])
```

In French: `ASPrinterList`

### Parameters

#### Result

Boolean – True if successful, False otherwise.

#### Connection

String – connection to AS/400

#### ASPrinterCollection

An ASPrinterCollection variable that will contain the result of the command.

This is a collection of objects of type ASPrinter

#### NamePrinter

String – printer name. Optional.

#### NameOutputQueue

String – qualified name of the output queue. Optional.

Example : "QUSRSYS/HPprt"

## Example

```

aspc est une ASPrinterCollection ;

HouvreConnexion(MaConnexionpower8)

//No filter
SI PAS AsPrinterList(aspc, MaConnexionpower8) ALORS
    Info(ErreurInfo());
SINON
    DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(aspc)
FIN

//Name filter (1)
SI PAS AsPrinterList(aspc, MaConnexionpower8,"TEST") ALORS
    Info(ErreurInfo());
SINON
    DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(aspc)
FIN

//Name filter (2)
SI PAS AsPrinterList(aspc, MaConnexionpower8,"QFQOUTQ;TEST") ALORS
    Info(ErreurInfo());
SINON
    DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(aspc)
FIN

//Output Queue filter (1)
SI PAS AsPrinterList(aspc, MaConnexionpower8,"", "QUSRSYS/HPDIJON") ALORS
    Info(ErreurInfo());
SINON
    DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(aspc)
FIN
    
```

```

//Output Queue filter (2)
SI PAS AsPrinterList(asperc, MaConnexionpower8, "",
"QUSRSYS/HPDIJON;*LIBL/QSYSOPR") ALORS
    Info(ErreurInfo());
SINON
    DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(asperc)
FIN

//Both Name & Output Queue filter (1 & 1)
SI PAS AsPrinterList(asperc, MaConnexionpower8,"HPDIJON", "QUSRSYS/HPDIJON")
ALORS
    Info(ErreurInfo());
SINON
    DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(asperc)
FIN

//Both Name & Output Queue filter (2 & 1)
SI PAS AsPrinterList(asperc, MaConnexionpower8, "HPDIJON;TEST",
"QUSRSYS/HPDIJON") ALORS
    Info(ErreurInfo());
SINON
    DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(asperc)
FIN

//Both Name & Output Queue filter (2 & 2)
SI PAS AsPrinterList(asperc, MaConnexionpower8, "HPDIJON;TEST",
"QUSRSYS/HPDIJON;*LIBL/QSYSOPR") ALORS
    Info(ErreurInfo());
SINON
    DSPPTR(); //procédure pour afficher les infos. Voir exemple fourni
    TableauSupprimeTout(asperc)
FIN
    
```

### Object of type *AsPrinterCollection*

This type of object is used in return of the *AsPrinterList* function. This is a collection of **AsPrinter** objects.

### Properties of *AsPrinter*

French name of the property	English name of the property	Type
DeviceName	DeviceName	string
TextDescription	TextDescription	string
OverallStatusCode	OverallStatusCode	integer 4 bytes
OverallStatus	OverallStatus	string
DeviceStatusCode	DeviceStatusCode	integer 4 bytes
DeviceStatus	DeviceStatus	string
OutputQueueName	OutputQueueName	string
OutputQueueLibrary	OutputQueueLibrary	string



OutputQueueStatus	OutputQueueStatus	string
WriterName	WriterName	string
WriterStatusCode	WriterStatusCode	integer 4 bytes
WriterStatus	WriterStatus	string
WriterStarted	WriterStarted	boolean
FormType	FormType	string
CurrentFileName	CurrentFileName	string
CurrentFileUser	CurrentFileUser	string
CurrentFileUserSpecifiedData	CurrentFileUserSpecifiedData	string
isPublishedInNetworkDirectory	isPublishedInNetworkDirectory	boolean

**NOTE:** OutputQueueName and OutputQueueLibrary will have a value if the printer is in use.

## DataQueue functions

### ASDataQueueSend

Writes into a dataqueue (keyed or non keyed).

With the new syntax, the PCML description of the dataqueue is a property of the [ASDataQueue](#) object.

In French: [ASEcritDataQueue](#)

## New syntax recommended since Windev 27

```
result = ASDataQueueSend(ASDataQueue [, Connection], data1 [, Data2,...] [,key])
```

### result

boolean – true if success, false in case of failure

### ASDataQueue

[ASDataQueue](#) type object containing the specification of the data queue (PCML description, name, library...).

The PCML syntax is the same as that used for [ASPgmCall](#).

### Connection

Connection – connection name. Optional.

### Data1

Variable or values corresponding to data to send.

### Data2...

Variable or values corresponding to data to send. Optional.

### Key

String for the key value. Optional.

## Examples with the new syntax

### 1) Keyed Dataqueue

```
sdq_pcm1 est chaîne = [
```

```

        <pcml version="4.0">
        <program name="TESTDQ" >
        <data name="data" type="char" length="200" usage="inputoutput"/>
        <data name="key" type="char" length="10" usage="inputoutput"/>
        </program>
        </pcml>
    ]

```

```
sClé = SAI_Clé1
```

```

asdq est une ASDataQueue
asdq.description      = sdq_pcml
asdq.library          = "EASYCOMXMP"
asdq.name              = "DTAQ_KEY"
asdq.timeout          = 0

```

```

SI PAS ASEcritDataQueue(asdq, SAI_valeur1, sClé) ALORS
    Info(ErreurInfo(errComplet))
SINON
    Info("Ecriture effectuée")
FIN

```

[See Example.](#)

## 2) FIFO Dataqueue

```

sdq_pcml est chaîne =[
    <pcml version="4.0">
    <program name="TESTDQ" >
    <data name="data" type="char" length="50" usage="input"/>
    </program>
    </pcml>
]

```

```

asdq est une ASDataQueue
asdq.description      = sdq_pcml
asdq.library          = "EASYCOMXMP"
asdq.name              = "DTAQ_FIFO"
asdq.timeout          = 0

```

```

SI PAS ASEcritDataQueue(asdq, PrinciConnexion,SAI_valeur3) ALORS
    Info(ErreurInfo(errComplet))
SINON
    Info("Ecriture effectuée")
FIN

```

[See Example.](#)

## Old syntax

```

result = ASDataQueueSend(Definition, DataQueueName [, Connection], data1 [,
Data2,...] [,key])

```

**result**

boolean – true if success, false in case of failure

**Definition**

Multiline string which contains the PCML source corresponding to the data and key (see above example).

The syntax is the same syntax which is used for [ASPgmCall](#).

**DataQueueName**

String representing the dataqueue name.

**Connection**

Connection – connection name. Optional.

**Data1**

Variable or values corresponding to data to send.

**Data2...**

Variable or values corresponding to data to send. Optional.

**Key**

String for the key value. Optional.

### Example with the old syntax

```

sdq_pcml is string =
<pcml version="4.0">
<program name=" ExampleDQK " >
<data name="data" type="char" length="200" usage="input"/>
<data name="key" type="char" length="10" usage="input"/>
</program>
</pcml>
]

IF NOT ASDataQueueSend(sdq_pcml, "*LIBL/DTAQ_KEY", cnxConnection,
sData, sKey) THEN
    Error("Write failed")
END
    
```

### ASDataQueueReceive

Receives data from a dataqueue, for a non-keyed dataqueue.

With the new syntax, the PCML description of the dataqueue is a property of the [ASDataQueue](#) object.

In French: [ASLitDataQueue](#)

### New syntax recommended since Windev 27

```
result = ASDataQueueReceive(ASDataQueue [, Connection], Param1 [, Param2, ... ] )
```

**Result**

boolean – true if success, false in case of failure

**ASDataQueue**

[ASDataQueue](#) type object containing the specification of the data queue (PCML description, name, library...).

The PCML syntax is the same as that used for [ASPgmCall](#).

**Connection**

Connection – connection name. Optional.

**Param1**

Data to receive the value

**Param2, ...**


Data to receive the value. Optional.

## Example with the new syntax

### FIFO Dataqueue

```
sdq_pcm1 est chaîne =[
    <pcm1 version="4.0">
    <program name="TESTDQ" >
    <data name="data" type="char" length="50" usage="input"/>
    </program>
    </pcm1>
]
```

```
data est chaîne sur 50
```

```
asdq est une ASDataQueue
asdq.description      = sdq_pcm1
asdq.library          = "EASYCOMXMP"
asdq.name             = "DTAQ_FIFO"
asdq.timeout          = 0
asdq.remove           = true
```

```
SI PAS ASLitDataQueue(asdq, data) ALORS
    Info(ErreurInfo(errComplet))
SINON
    Info(data)
FIN
```

[See Example.](#)

### Old syntax

```
result = ASDataQueueReceive(Definition, DataQueueName , Timeout, Remove, [,
Connection], Param1 [, Param2, ... ] )
```

#### Result

boolean – true if success, false in case of failure

#### Definition

Multiline string which contains the PCML source corresponding to the data and key (see above example)

The syntax is the same syntax which is used for [ASPgmCall](#)

#### DataQueueName

String representing the dataqueue name.

#### Timeout

Timeout value in case of no data at the moment of the call. 0 for immediate answer.

#### Remove

Tell to remove or just read the dataqueue value.

**Connection**

Connection – connection name. Optional.

**Param1**

Data to receive the value.

**Param2, ...**

Data to receive the value. Optional.

**Example with the old syntax**

```
sdq_pcml is string =
<pcml version="4.0">
<program name="TESTDQ" >
<data name="data" type="char" length="50" usage="input"/>
</program>
</pcml>
]

IF NOT ASDataQueueReceive (sdq_pcml, "*LIBL/DTAQ_FIFO", 2, True ,
cnxConnection, data_read) THEN
Info("Data Queue vide!")
END
```

**ASKeyDataQueueReceive**

Receives data from a dataqueue, using the specified key.  
With the new syntax, the PCML description of the dataqueue is a property of the [ASDataQueue](#) object.

In French: [ASLitDataQueueCle](#)

**New syntax recommended since Windev 27**

```
result = ASKeyDataQueueReceive (ASDataQueue, Operator [, Connection], Data1 [,
Data2...], Key )
```

**Result**

boolean – true if success, false in case of failure

**ASDataQueue**

[ASDataQueue](#) type object containing the specification of the data queue (PCML description, name, library...).

The PCML syntax is the same as that used for [ASPgmCall](#).

**Operator**

String - Key operation. Possible values are:  
EQ: equal, NE: not equal, LE: <=, LT : <, GE : >=, GT : >.

**Connection**

Connection – connection name. Optional.

**Data1**

Data to receive the value.

**Data2...**

Data to receive the value. Optionnel.

**Key**

String – key value.



## Example with the new syntax

```
sdq_pcm1 est chaîne =[
  <pcml version="4.0">
  <program name="TESTDQ" >
  <data name="data" type="char" length="200" usage="inputoutput"/>
  <data name="key" type="char" length="10" usage="inputoutput"/>
  </program>
  </pcml>
]
```

```
data est chaîne sur 50
```

```
asdq est une ASDataQueue
asdq.description      = sdq_pcm1
asdq.library          = "EASYCOMXMP"
asdq.name             = "DTAQ_KEY"
asdq.timeout          = 0
asdq.remove           = true
```

```
SI PAS ASLitDataQueueCle(asdq, "EQ", data, SAI_Clé1) ALORS
  Info(ErreurInfo(errComple))
SINON
  Info(data)
FIN
```

[See Example.](#)

## Old syntax

```
result = ASKeyDataQueueReceive (Definition, DataQueueName, Operator, Timeout,
Remove [, Connection], Data1 [, Data2...], Key )
```

### Result

boolean – true if success, false in case of failure

### Definition

Multiline string which contains the PCML source corresponding to the data and key (see above example)

The syntax is the same syntax which is used for [ASPqmCall](#)

### DataQueueName

String representing the dataqueue name.

### Operator

Key operation. Possible values are:

EQ: equal, NE: not equal, LE : <=, LT : <, GE : >=, GT : >.

### Timeout

Timeout value in case of no data at the moment of the call. 0 for immediate answer.

### Remove

Tell to remove or just read the dataqueue value.

**Connection**

Connection – connection name. Optional.

**Data1**

Data to receive the value.

**Data2....**

Data to receive the value. Optional.

**Key**

Key value.

## Example with the old syntax

```

sdq_pcml is string =
<pcml version=="4.0">
<program name="TESTDQ" >
<data name="data" type="char" length=200" usage="input"/>
<data name="key" type="char" length="10" usage="input"/>
</program>
</pcml>
]

IF NOT ASKeyDataQueueReceive(sdq_pcml, "*LIBL/DTAQ_KEY", "EQ", 2, True
, cnxConnection, data_read, key2) THEN
    Info("Empty data queue!")
END
    
```

## ASDataQueueInfo

Allows you to find information about the specified data queue.

This information is available in the [ASDataQueueInfo](#) type object, passed as a parameter.

## Syntax

```

Result = ASDataQueueInfo(ASDataQueueInfo, Connection, DataQueueName,
[DataQueueLibrary])
    
```

In French: [ASDataQueueInfo](#)

## Parameters

**Result**

Boolean – true if success, false in case of failure.

**ASDataQueueInfo**

ASDataQueueInfo object containing, in return for the command, the information of the specified data queue.

**Connection**

Connection – connection name.

**DataQueueName**

String – qualified name of the data queue.

**DataQueueLibrary**

String – Library containing the dataqueue. Optional if this library is already part of the list of libraries of the JOB corresponding to the connection (\*LIBL).

## Example

1) "DTAQ\_KEY" is part of \*LIBL

```

asdqi    est un ASDataQueueInfo

SI PAS ASDataQueueInfo(asdqi,PrinciConnexion,"DTAQ_KEY") ALORS
      Info(ErreurInfo(errComple))
SINON
      SAI_DQK = "LIBRARY = " + asdqi.dataQueueLib + RC + "Type = " +
asdqi.sequenceDesc + RC + "Description = " + asdqi.dataQueueDescription +
      RC + "Number of messages = " + asdqi.nbMessage + RC + "Key length = " +
asdqi.keyLength
FIN
    
```

2) Specifying the library containing the dataqueue

```

asdqi    est un ASDataQueueInfo

SI PAS ASDataQueueInfo(asdqi,PrinciConnexion,SAI_DQname, SAI_DQlib) ALORS
      Info(ErreurInfo(errComple))
SINON
      SAI_DQK2 = "LIBRARY = " + asdqi.dataQueueLib + RC + "Type = " +
asdqi.sequenceDesc + RC + "Description = " + asdqi.dataQueueDescription +
      RC + "Number of messages = " + asdqi.nbMessage
FIN
    
```

## ASDataQueueClear

Allows you to empty the content of the data queue specified via the [ASDataQueue](#) object.

## Syntax

```
result = ASDataQueueClear(ASDataQueue, Connection)
```

In French: [ASDataQueueClear](#)

## Parameters

### result

boolean – true if success, false in case of failure

### ASDataQueue

[ASDataQueue](#) type object.

### Connection

Connection – connection name.

## Example

```

asdq est une ASDataQueue
asdq.description      = sdq_pcm1
asdq.library          = "EASYCOMXMP"
    
```





```

asdq.name           = "DTAQ_KEY"
asdq.timeout        = 0

SI PAS ASDataQueueClear(asdq, PrinciConnexion) ALORS
Info(ErreurInfo(errComplet))
FIN
    
```

### Object of type ASDataQueue

#### Properties of ASDataQueue

Property name	Description	Type
description	Dataqueue description. PCML.	string
library	Dataqueue library	string
name	Dataqueue name	string
timeout	Timeout. For reading, waiting time in case the data queue is empty.	integer
remove	Remove data after reading.	boolean

### Object of type ASDataQueueInfo

This type of object is used in return of the [ASDataQueueInfo](#) function.

#### Properties of ASDataQueueInfo

Property name	Description	Type
dataQueueDescription	Description dataqueue	string
dataQueueLib	Bibliothèque dataqueue	string
dataQueueName	Nom dataqueue	string
initialNbEntries	Nombre initial d'entrées	integer
isAutoReclaimed	Récupération automatique	boolean
isDDMDataQueue	Description de la station	boolean
isEnforcedDQLocks	Appliquer des verrous de file d'attente de données	boolean
isForceAuxStorage	Whether or not the data queue is forced to auxiliary storage when entries are sent or received for the specified data queue.	boolean
isSenderIdIncluded	Include sender ID.	boolean
keyLength	Key length	integer

lastReclaimDateTime	Last reclaim date and time.	Date Time
maxNbEntriesSpecified	Maximum number of entries specified.	integer
messageLength	Message length.	integer
nbAllocatedEntries	Number of entries currently allocated.	integer
nbMaxAllowedEntries	Maximum number of entries allowed.	integer
nbMessage	Number of messages.	integer
sequence	Sequence.	string
sequenceDesc	Sequence description.	string

## Examples

### 1) Write and Read – Data Queue with key

```
sdq_pcm1 est chaîne =[
    <pcml version="4.0">
    <program name="TESTDQ" >
    <data name="data" type="char" length="200" usage="inputoutput"/>
    <data name="key" type="char" length="10" usage="inputoutput"/>
    </program>
    </pcml>
]
```

```
H0uvreConnexion(MaConnexionpower8)
```

```
sClé = SAI_Clé1
```

```
asdq est une ASDataQueue
asdq.description      = sdq_pcm1
asdq.library          = "EASYCOMXMP"
asdq.name             = "DTAQ_KEY"
asdq.timeout          = 0
asdq.remove           = true
```

```
SI PAS ASEcritDataQueue(asdq, SAI_valeur1, sClé) ALORS
    Info(ErreurInfo(errComplet))
SINON
    Info("Ecriture effectuée")
FIN
```

```
//Read
```

```
data est chaîne sur 50
```

```
SI PAS ASLitDataQueueCle(asdq, "EQ", data, SAI_Clé1) ALORS
    Info(ErreurInfo(errComplet))
SINON
    Info(data)
FIN
```

## 2) Write and Read – Data Queue without key

```
sdq_pcm1 est chaîne =[
    <pcml version="4.0">
    <program name="TESTDQ" >
    <data name="data" type="char" length="50" usage="input"/>
    </program>
    </pcml>
]
```

H0uvreConnexion(MaConnexionpower8)

```
asdq est une ASDataQueue
asdq.description      = sdq_pcm1
asdq.library          = "EASYCOMXMP"
asdq.name              = "DTAQ_FIFO"
asdq.timeout          = 0
asdq.remove           = true
```

```
SI PAS ASEcritDataQueue(asdq, PrinciConnexion,SAI_valeur3) ALORS
    Info(ErreurInfo(errComplet))
SINON
    Info("Ecriture effectuée")
FIN
```

```
//Read
data est chaîne sur 50
```

```
SI PAS ASLitDataQueue(asdq, data) ALORS
    Info(ErreurInfo(errComplet))
SINON
    Info(data)
FIN
```

## UserSpace functions

### ASUserSpaceWrite

Writes structured or simple information into a userspace.

### Syntax

```
result = ASUserSpaceWrite(UserSpaceDesc, UserSpaceName, Offset [, Connection],
param1 [,Param 2 [, Param3, ... ] ] )
```

In French : [ASUserSpaceWrite](#)

### Parameters

#### Result

boolean – true if success, false in case of failure

#### UserSpaceDesc

Multiline string which contains the PCML source corresponding to the data in the userspace at the specified offset

The syntax is the same syntax which is used for [ASPgmCall](#)

**UserSpaceName**

String representing the userspace name.

**Connection (optional)**

Connection to AS/400.

**Offset**

Offset in the userspace

**Param1, Param2, ...**

Data matched to the described structure. Can be a structure (one parameters) or single value(s).

### Example

```

str is string

sus_pcml is string =
<pcml version=="4.0">
<program name="TESTUS" >
<data name="data" type="char" length="45" usage="input"/>
</program>
</pcml>
]

str="uspc test"

IF NOT ASUserSpaceWrite (sus_pcml, "MY_LIB/MY_US", 1, str)
    Error (ErrorInfo ())
END
    
```

### ASUserSpaceRead

Reads structured or simple information from a userspace.

### Syntax

```

result = ASUserSpaceRead (UserSpaceDesc, UserSpaceName, Offset [, Connection],
param1 [, Param 2 [, Param3, ... ] ] )
    
```

In French: [ASLitUserSpace](#)

### Parameters

**Result**

boolean – true if success, false in case of failure

**UserSpaceDesc**

Multiline string which contains the PCML source corresponding to the data in the userspace at the specified offset

The syntax is the same syntax which is used for [ASPgmCall](#)

**UserSpaceName**

String representing the userspace name.

**Connection (optional)**

Connection to AS/400.

**Offset**

Offset in the userspace



**Param1, Parm2, ...**

Data matched to the described structure. Can be a structure (one parameters) or single value(s).

**Example**

```

sus_pcml is string =
<pcml version="4.0">
<program name="US_TEST" >
<data name="data" type="char" length="45" usage="input"/>
</program>
</pcml>
]

IF NOT ASUserSpaceRead(sus_pcml, "MY_LIB/MY_US", 1, res)
    Error(ErrorInfo())
END
    
```

### *ASUserSpaceCreate*

It allows to create a userspace.

**Syntax**

```

Result = ASUserSpaceCreate(Connection, UserSpaceName, LibraryName)
    
```

In French: [ASCreeUserSpace](#).

**Parameters**
**Result**

boolean – true if success, false in case of failure.

**Connection**

Connection to AS/400.

**UserSpaceName**

String representing the userspace name.

**LibraryName**

String representing the library name containing the user space.

**Example**

```

IF NOT ASUserSpaceCreate(MaConnexionpower8,"USPC1","CR")
    Info(ErreurInfo())
END
    
```

### *ASInfoUserSpace*

This function retrieves userspace info.  
Info is stored in an object of type ASUserSpaceInfo.

**Syntax**

```

Result = ASInfoUserSpace(UserSpaceInfo, Connection, UserSpaceName, LibraryName)
    
```



In French: [ASInfoUserSpace](#).

## Parameters

### Result

boolean – true if success, false in case of failure.

### UserSpaceInfo

Object of type ASUserSpaceInfo which will contain the infos of the specified userspace.

### Connection

Connection to AS/400.

### UserSpaceName

string – name of the userspace.

### LibraryName

string – name of the library containing the userspace.

## Example

infoUSPC est un ASUserSpaceInfo

```
SI ASInfoUserSpace(infoUSPC, MaConnexionpower8, "USPC1", "*LIBL") ALORS
```

```
    SI infoUSPC.autoExtend ALORS
```

```
        isAutoExtend = Vrai
```

```
    FIN
```

```
        Info("USPC LIB : " + infoUSPC.uspcLibName, "USPC Space : " +
infoUSPC.spaceSize, "USPC AutoExtend : " + isAutoExtend, "USPC Initial Char value : "
+ infoUSPC.InitialValue);
```

```
SINON
```

```
    Info(ErreurInfo())
```

```
FIN
```

[Voir Exemple.](#)

## ASUserSpaceDelete

This function allows to delete a userspace.

## Syntax

```
Result = ASUserSpaceDelete(Connection, UserSpaceName, LibraryName)
```

In French: [ASSupprUserSpace](#).

## Parameters

### Result

boolean – true if success, false in case of failure.

### Connection

Connection to AS/400.

### UserSpaceName



string – name of the userspace.

**LibraryName**

string – name of the library containing the userspace.

### Example

`ASUserSpaceDelete (MaConnexionpower8, "USPC1", "CR")`

### Object of type *ASUserSpaceInfo*

This type of object is used in return of the **ASInfoUserSpace** function.

### Properties

autoExtend	Automatic extendibility (true/false)
uspclibName	The library in which the user space is located. This is helpful when *LIBL or *CURLIB is specified as the library name in the qualified user space name parameter.
spaceSize	The size of the user space object in bytes.
InitialValue	The initial value to which future extensions of the user space will be set. Zéro, space...

### Example

Create user space, retrieve infos, write, read and delete.

ch est chaîne

```
sus_pcm1 est chaîne = [
    <pcml version=="4.0">
    <program name="TESTUS" >
    <data name="data" type="char" length="45" usage="input"/>
    </program>
    </pcml>
]
```

`H0uvreConnexion(MaConnexionpower8)`

**//Create**

```
SI PAS ASCreeUserSpace(MaConnexionpower8, "USPC1", "CR") ALORS
    Info(ErreurInfo())
FIN
```

isAutoExtend est un booléen = *Faux*

infoUSPC est un *ASUserSpaceInfo*

**//Retrieve infos**

```
SI ASInfoUserSpace(infoUSPC, MaConnexionpower8, "USPC1", "*LIBL") ALORS
```

```
    SI infoUSPC.autoExtend ALORS
```

```
        isAutoExtend = Vrai
```



```
FIN

Info("USPC LIB : " + infoUSPC.uspcLibName,"USPC Space : " +
infoUSPC.spaceSize,"USPC AutoExtend : " + isAutoExtend ,"USPC Initial Char value : "
+ infoUSPC.InitialValue);

SINON
Info(ErreurInfo())
FIN

ch="test uspc"

//Write
SI PAS ASEcritUserSpace(sus_pcml, "CR/USPC1", 1, MaConnexionpower8,ch)
Erreur(ErreurInfo())
FIN

sRes est une chaîne

//Read
SI PAS ASLitUserSpace(sus_pcml, "CR/USPC1", 1, MaConnexionpower8, sRes)
Erreur(ErreurInfo())
SINON
Info(sRes)
FIN

//Delete
ASSupprUserSpace(MaConnexionpower8,"USPC1","CR")
```

## Spool functions

### ASCreateSpool

Creates a binary spool file on the AS/400, from a windows printer file.  
This allows printing a report to the AS/400 spools.

### Syntax

```
Result = ASCreateSpool(prn_FileName, Connection, options [, user_data, user_name,
user_text [, copies, spoolname [, output_queue, output_priority, form_type]])
```

In French: [ASCreeSpool](#).

### Parameters

#### Result

boolean – true if success, false in case of failure.

#### Connection

Connection to AS/400.

#### prn\_FileName

Print file in the format of the destination printer (PCL or other). Use `iDestination` (`iFile`, "file\_name") to create the print file from a classic WinDev print..

#### Options

Basic options for the spool file. The possible valeurs are a combination of the following constants:





ASsplHold : for creating the spool file in hold state.

ASsplSave : for a spool file which will be saved after being printed.

**User\_data** (10 characters)

User data: a short identifier for the spool item. Also known as user reference. This is used to easily retrieve the spool after creation.

**User\_name**

Name of the user which will own the spool. Possible values are:

Name of a user

\*CURRENT (by default) -> current user

**User\_text** (100 characters)

User text. Holds a customized description.

**copies**

Number of copies to be printed.

**spoolname** (10 characters)

Name of the spool file to create.

**Output\_Queue**

Name of the outq. Possible values are:

Name of an OUTQ (\*LIBL/OUTQ, or OUTQ)

\*JOB (by default)

**Output\_priority**

Output priority. Possible values:

1-9

\*JOB (by default)

**Form\_type** (10 characters)

Form type: possible values are: \*STD or any string.

**Example**

```
// redirects the REPORT_Report1 print into a prn file
iDestination(iFile, fTempPath()+"impr.prn")
iPrintReport(REPORT_Report1)
```

```
// example 1
IF NOT ASCreateSpool(fTempPath()+"impr.prn", cnxConnection,0, "QPGMR",
"DOC0099", "Invoice XYZ") THEN
  Error(ErrorInfo(errFullDetails))
END
```

```
// example 2
IF NOT ASCreateSpool(fTempPath()+"impr.prn",
cnxConnection,ASsplHold+ASsplSave, "*CURRENT", "DOC0099", "Invoice
XYZ",2, "MYSPOOL", "PRT_EAC", 1, "FRM01") THEN
  Error(ErrorInfo(errFullDetails))
END
```

**ASGetSpool**

Reads an AS/400 spool file into a local PCL or text document.



## Syntax

```
Result = ASGetSpool(local_FileName, Connection , SpoolName [, job [, spl_number [, spl_format [, paper_size [, supprLF ]]]]] )
```

In French: [ASGetSpool](#).

## Parameters

### Result

boolean – true if success, false in case of failure

### local\_FileName

PC file which will be created by ASGetSpool.

### Connection

Connection to AS/400.

### SpoolName

Name of the spool file. Possible values: \*LAST or a name

### Job

Job identity. Possible value is \* for current job, or nnn/USER/JOBNAME.

### Spl\_number

Number of the spool. Possible values are:

- from 0 to 9999999: spool number
- ASSplOnly (\*ONLY)
- ASSplLast (\*LAST)

### Spl\_format

Format of the result. Possible values are:

- ASSplTxt: text format
- ASSplPCL: PCL format

### Paper\_size

Paper size. Possible values are:

\*NONE \*MFRTPMDL \*LETTER \*LEGAL \*EXECUTIVE \*A3 \*A4 \*A5 \*B4 \*B5 \*CONT80 \*CONT132

### supprLF (boolean)

Remove the last page jump (by default: false)

## Example

```
IF NOT ASGetSpool("c:\temp\spl_test_wd.spl", cnxConnection,
"QPJOBLOG", "108755/SMITH/PRTXJK", 1, ASSplPCL, "", False) THEN
  Error(ErrorInfo(errFullDetails))
END
```

## ASSpoolList

List the AS400 spool files.

## Syntax

```
Result = ASSpoolList(CollectionOfASSpool, User, OutQ, UserDataKey, JobName, jobNumber, Connection)
```

In French: [ASSpoolList](#).

## Parameters

### Result

Boolean- True if the command succeeded false otherwise.

### CollectionOfASSpool

ASSpoolCollection variable, it will be filled with the list of spool.

### User

Name of the user who created the job.

### OutQ

Name of the out queue containing the spool file.

### UserDataKey

User Data Key for the spool file.

### JobName

Name of the job which created the spool file.

### JobNumberI

Number of the job which created the spool file.

### Connection

Connection name of the AS400.

## Example

`listspool` is `ASSpoolCollection`

```
//look for all the spool files
IF NOT ASSpoolList(listspool, "", "", "", "", "", MyConnection)
  Error(ErrorInfo())
END
```

## Object of ASSpoolCollection type

This type of object is used in return of the `ASSpoolList` function. This is a collection of **ASSpool** objects.

## Properties of the ASSpool object

Property	AS400 type
Job name	CHAR(10)
User name	CHAR(10)
Job number	CHAR(6)
Spoiled file name	CHAR(10)
Spoiled file number	BINARY(4)
Spoiled file status	BINARY(4)
Date file was opened (created)	CHAR(7)
Time file was opened (created)	CHAR(6)
Spoiled file schedule	CHAR(1)
Spoiled file system name	CHAR(10)
User-specified data	CHAR(10)

Form type	CHAR(10)
Output queue name	CHAR(10)
Output queue library name	CHAR(10)
Auxiliary storage pool	BINARY(4)
Size of spooled file	BINARY(4)
Spooled file size multiplier	BINARY(4)
Total pages	BINARY(4)
Copies left to produce	BINARY(4)
Priority	CHAR(1)

## Trace Management functions

### *ASSetTrace*

Enable or disable the AS400 trace.

### Syntax

```
Result = ASSetTrace(ActiveTrace, TraceFile, TraceLvl, TimeStamp, Connection)
```

In French: *ASActiveTrace*.

### Parameters

#### Result

Boolean- True if the command is successful.

#### ActiveTrace

Boolean, if true, enable tracing, if false disable it.

#### TraceFile

Name of the trace file, it has to be in the following format: LIBRARY/FILE.

#### TraceLvl

Level of detail for the trace, has to be in between 1 and 4, 4 providing the highest level of details.

#### TimeStamp

Boolean, if true, each line of the trace will be timestamp

#### Connection

Connection – name of the AS400 Connection

### Example

```
//activate trace
IF NOT ASSetTrace(True, "EASYCOM/TRACE", 4, True, MyConnection)
    Error(Info())
FIN
```

### *ASIfTraceActive*

Check if the trace is activated.

### Syntax

```
Result = ASIfTraceActive(Connection)
```

In French: [ASSiTraceActive](#).

## Parameters

### Result

Boolean- True if the trace is activated.

### Connection

Connection – name of the AS400 Connection.

## *ASGetTraceLib*

Retrieve the name of the library used for trace.

## Syntax

```
Result = ASGetTraceLib(Connection)
```

In French: [ASGetTraceLib](#).

## Parameters

### Result

String – Name of the library used for the trace.

### Connection

Connection – name of the AS400 Connection.

## *ASGetTraceFic*

Retrieve the name of the file used for trace.

## Syntax

```
Result = ASGetTraceFic(Connection)
```

In French: [ASGetTraceFic](#).

## Parameters

### Result

String – Name of the file used for the trace.

### Connection

Connection – name of the AS400 Connection.

## *ASGetTraceLvl*

Retrieve the current level of detail for trace.

## Syntax

```
Result = ASGetTraceLvl(Connection)
```

In French: [ASGetTraceLvl](#).

## Parameters

### Result

Integer – Level of detail used for the trace.

### Connection

Connection – name of the AS400 Connection.

## Example

HouvreConnexion(MaConnexionpower8)

```
SI ASIfTraceActive(MaConnexionpower8) ALORS
    sLib est une chaîne
    sLib = ASGetTraceLib(MaConnexionpower8)
    Info(sLib)

    sFic est une chaîne
    sFic = ASGetTraceFic(MaConnexionpower8)
    Info(sFic)

    nLvl est un entier
    nLvl = ASGetTraceLvl(MaConnexionpower8)
    Info(nLvl)
```

FIN

## Job Management functions

### ASJobLogList

List the objects of a job log AS/400.

## Syntax

```
Result = ASJobLogList(ASJobLogCollection, JobName, User, JobNumber,
    NumberOfMessages, Direction, Connection)
```

In French: [ASJobLogList](#).

## Parameters

### Result

Boolean – true if the command was successful.

### ASJobLogCollection

ASJobLogCollection variable which will contains the results.

### JobName

Filter using the name of the job, special values allowed : \*, \*INT. Per default \* is used.

### User

Name of the user, or leave empty when the JobName used is a special value \* or \*INT.

### JobNumber

Number of a specific job or leave empty when the JobName used is a special value \* or \*INT.

### NumberOfMessages

Maximum number of messages to be returned.

### Direction

Reading directions, you have to user \*NEXT or \*PRV. Per default \*NEXT is used.

**Connection**

Connection – name of Connection

**Example**

`listJobLog` is a `ASJobLogCollection`

```
//Per default behaviour, list all the job logs
IF NOT ASJobLogList(listJobLog, "", "", "", "", "", MaConnection1)
  Error(ErrorInfo())
END
```

**ASJobList**

List jobs of an AS400 system.

**Syntax**

```
Result = ASJobList(ASJobCollection, JobName, User, JobNumber, Type, Status,
Connection)
```

In French: `ASJobList`.

**Parameters**

**Result**

Boolean – true if the command was successful.

**ASJobCollection**

ASJobCollection variable which will contain the list of ASJob

**JobName**

Filter using a job name, special values allowed are \*CURRENT, \*ALL. \*ALL is used per default.

**User**

Filter using the name of the user which created the job, special values allowed are \*CURRENT, \*ALL. \*CURRENT is used per default.

**JobNumber**

Filter using the job number, special values allowed \*, \*INT. \* is used per default.

**Type**

Filter using job type, special values allowed are \*, A, B, I, M, R, S, W, X. \* is used per default.

**Status**

Filter using the status of the job, special values are \*ACTIVE, \*JOBQ, \*OUTQ, \*ALL. Per default, \*ACTIVE is used.

**Connection**

Connection – name of the Connection

**Example**

`listJob` is a `ASJobCollection`

```
//per default behaviour, list all the active jobs for the current user
IF NOT ASJobList(listJob, "", "", "", "", "", MaConnection1)
  Error(ErrorInfo())
END
```



**Object of type ASJobCollection**

This type of object is used in return of the [ASJobList](#) function.

An ASJobCollection contains a certain number of ASJob objects; those objects have the following properties:

Property	Description
Job_name	Job name used
Job_user_name	User name used
Job_number	Job number used
Job_internal_id	Internal job identifier
Job_status	Status
Job_type	Job type
Job_subtype	Job subtype
Job_info_status	Job information status
Job_act_job_sts	Active job status
Job_alw_multi_threads	Allow multiple threads
Job_act_endjob_sts	Active job status for jobs ending
Job_brkmsg	Break message handling
Job_cancel_key	Cancel key
Job_ccsid	Coded character set ID
Job_cntryid	Country or region ID
Job_cpu_time	Processing unit time used, if less than 2,147,483,647 milliseconds
Job_usrprf	Current user profile
Job_completion_sts	Completion status
Job_pool_id	Current system pool identifier
Job_char_id_ctrl	Character identifier control
Job_process_unit_time	Processing unit time used - total for the job
Job_process_unit_time_db	Processing unit time used for database - total for the job
Job_datetime_active	Date and time job became active
Job_datetime_in	Date and time job entered system
Job_datetime_sched	Date and time job is scheduled to run
Job_datetime_jobq	Date and time job was put on this job queue
Job_datfmt	Date format
Job_datsep	Date separator
Job_dbcs_cap	DBCS-capable
Job_ddm_handle	DDM conversation handling
Job_dftwait	Default wait
Job_devrcyacn	Device recovery action
Job_devname	Device name
Job_dftccsid	Default coded character set identifier
Job_decfmt	Decimal format
Job_datetime_end	Date and time job ended
Job_endsev	End severity
Job_endsts	End status



Job_exitkey	Exit key
Job_func_name	Function name
Job_func_type	Function type
Job_signed_job	Signed-on job
Job_grpprfname	Group profile name
Job_grpprfname_sup	Group profile name - supplemental
Job_inqmsgreply	Inquiry message reply
Job_account_code	Job accounting code
Job_date	Job date
Job_desc_name	Job description name - qualified
Job_queue_name	Job queue name - qualified
Job_queue_pty	Job queue priority
Job_switches	Job switches
Job_jobmsgqfl	Job message queue full action
Job_jobmsgq_size	Job message queue maximum size
Job_usrid	Job user identity
Job_usrid_setting	Job user identity setting
Job_end_reason	Job end reason
Job_log_pending	Job log pending
Job_type_enhanced	Job type - enhanced
Job_langid	Language ID
Job_loglvl	Logging level
Job_logclpgm	Logging of CL programs
Job_logsev	Logging severity
Job_logtext	Logging text
Job_mode_name	Mode name
Job_max_proc_unit_time	Maximum processing unit time
Job_max_tmp_stg_k	Maximum temporary storage in kilobytes
Job_max_threads	Maximum threads
Job_max_tmp_stg_m	Maximum temporary storage in megabytes
Job_mem_pool_name	Memory pool name
Job_msgrpl	Message reply
Job_interactive_trs	Number of interactive transactions
Job_db_lckwait	Number of database lock waits
Job_mch_lckw	Number of internal machine lock waits
Job_nondb_lckw	Number of nondatabase lock waits
Job_aux_ioreq	Number of auxiliary I/O requests
Job_outq_name	Output queue name - qualified
Job_outq_pty	Output queue priority
Job_prttext	Print text
Job_prtdevname	Printer device name
Job_purge	Purge
Job_prd_retcode	Product return code

Job_prog_retcde	Program return code
Job_pending_sgnset	Pending signal set
Job_process_id	Process ID number
Job_response_time	Response time total
Job_runpty	Run priority (job)
Job_routing_data	Routing data
Job_strseq	Sort sequence table - qualified
Job_sts_msghdl	Status message handling
Job_sts_jobq	Status of job on the job queue
Job_sbmjob	Submitter's job name - qualified
Job_sbmmmsgq	Submitter's message queue name - qualified
Job_sbsd	Subsystem description name - qualified
Job_syspoolid	System pool identifier
Job_spclenv	Special environment
Job_sgnblk_mask	Signal blocking mask
Job_sgnsts	Signal status
Job_svrtype	Server type
Job_splfile_action	Spooled file action
Job_timsep	Time separator
Job_timeslice	Time slice
Job_timeslice_end	Time-slice end pool
Job_tmpstgk	Temporary storage used in kilobytes
Job_time_db_lckw	Time spent on database lock waits
Job_time_mch_lckw	Time spent on internal machine lock waits
Job_time_nondb_lckw	Time spent on nondatabase lock waits
Job_threadcnt	Thread count

### ASChangeJobName

Customize the « function » field displayed in the WRKACTJOB listing for the Easycom sub-system corresponding to the connection.

### Syntax

```
Result = ASChangeJobName(NewName, Connection)
```

In French : [ASChangeNomJob](#).

### Parameters

#### Result

Int – -1 in case of Error, 0 otherwise

#### NewName

Name which will be displayed in the WRKACTJOB results on as400

#### Connection

Connection – name of the AS400 Connection

## Example

```

MaConnexion est une Connexion
// Description de la connexion
MaConnexion..Utilisateur = "aura"
MaConnexion..MotDePasse = "aura"
MaConnexion..Serveur = "power8"
MaConnexion..Provider = hAccèsNatifAS400
MaConnexion..InfosEtendues= "<EASYCOM>"+CRLF+"JOBNAME=CED"+CRLF+"</EASYCOM>"
HOuvreConnexion(MaConnexion)
    
```

//WRKACTJOB result :

Opt	S-syst/trav	cours	Type	% UC	Fonction	Etat
█	EASYCOM	QSYS	SBS	0,0		DEQW
█	CED	QPGMR	BCH	0,0	PGM-EASYCOM	TIMW
█	EASYCOMD	QTCP	ASJ	0,0	PGM-EASYCOMD	SELW
█	NISSA01	QPGMR	BCH	0,0	PGM-EASYCOM	TIMW

```

SI PAS ASChangeNomJob("TEST", MaConnexion) ALORS
    Info(ErreurInfo())
    
```

FIN

//WRKACTJOB result after ASChangeNomJob :

Opt	S-syst/trav	cours	Type	% UC	Fonction	Etat
█	EASYCOM	QSYS	SBS	0,0		DEQW
█	CED	QPGMR	BCH	0,0	USR-TEST	TIMW
█	EASYCOMD	QTCP	ASJ	0,0	PGM-EASYCOMD	SELW
█	NISSA01	QPGMR	BCH	0,0	PGM-EASYCOM	TIMW

```

HFermeConnexion(MaConnexion)
    
```

## ASSubmitBC I

Submit a Batch Immediat job (BCI). Be careful, if you want to pass data to the program you're calling, you need to base your program on the example provide (EASYCOMXMP/QCLSRC TESTPUTRQ).

**It is not possible to directly call a program with parameter such as with the ASExec function.**

## Syntax

```

Result = ASSubmitBCI(ProgName, Library, User, Data, Multi-Thread, CurrLIBL,
Connection)
    
```

In French : [ASSoumetBCI](#).

## Parameters

### Result

ANSI String– ID of the job if it started correctly, otherwise, it's an empty string.

### ProgName

Name of the AS400 program name to start

### Library

Library of the AS400 program



**User**

Name of the user to be used to start the job

**Data**

Data to be passed on to the job.

**Multi-thread**

Boolean– if true authorize use of multi-thread in the job.

**CurrLIBL**

Boolean–if true, keep the current LIBL for the job execution.

**Connection**

Connection – name of the AS400 Connection

## Example

```
JobId = ASSoumetBCI(pgm,lib,sUser,Data,bMultiThread,bCurrlibl)
```

### *ASSubmitPJ*

Submit a pre-start job (PJ). Be careful, if you want to pass data to the program you're calling, you need to base your program on the example provide (EASYCOMXMP/QCLSRC TESTPUTRQ).

**It is not possible to directly call a program with parameter such as with the ASExec function.**

## Syntax

```
Result = ASSubmitPJ(Program, User, Data, SBS, Connection)
```

In French : [ASSoumetPJ](#) .

## Parameters

**Result**

ANSI String– ID of the job if it started correctly, otherwise, it's an empty string.

**Program**

Name of the AS400 program name to start

**User**

Name of the user to be used to start the job

**Data**

Data to be passed on to the job.

**SBS**

Name of the subsystem containing the pre-start job.

**Connection**

Connection – name of the AS400 Connection.

## Example

```
JobId = ASSoumetPJ(pgm,sUser,Data,sbs)
```

### *ASJobStatus*

Allows to obtain the Status of a Job launched in BIC or PJ.

#### Syntax

```
Result = ASJobStatus (JobID, Connection)
```

In French : [ASEtatJob](#).

#### Parameters

**Result**

String – Status of the job.

**JobID**

Job Id as return by ASSubmitBIC ou ASSubmitPJ.

**Connection**

Connection – name of the AS400 Connection.

### Locks management functions

#### *ASJobRecordLocks*

List all the locks of an AS/400 job in progress.

#### Syntax

```
Result = ASJobrecordLocks (ASJobRecordLockCollection, JobName, UserName, JobNumber, Connection)
```

In French: [ASJobrecordLocks](#).

#### Parameters

**Result**

Boolean- True if the command is successful.

**ASJobRecordLockCollection**

Variable of type ASJobRecordLockCollection which will contain the results.

**JobName**

JOB name.

**UserName**

Username.

**JobNumber**

Job number.

**Connection**

Connection – name of the AS400 Connection.

#### Example

```
lockList est une ASJobRecordLockCollection  
nbLock est un entier  
monLock est un ASJobRecordLock
```



```

AsJobRecordLocks(lockList, jb.Job_name, jb.Job_user_name, jb.Job_number, PrinciCo
nnexion)
nbLock = lockList.CollectionASJobRecordLock..Occurrence
POUR i =1 A nbLock
    monLock = lockList.CollectionASJobRecordLock[i]
    TableauAjoute(gtabAsJobRecordLock, monLock)
FIN
    
```

## ASRecordLocks

List the locks on a database file.

### Syntax

```
Result = ASRecordLocks (ASRecordLockCollection, File, Library, Member, Connection)
```

In French : [ASRecordLocks](#) .

### Parameters

#### Result

Boolean- True if the command is successful.

#### ASRecordLockCollection

Variable of type ASRecordLockCollection which will contain the results.

#### File

File on which locks are listed.

#### Library

Library, it is possible to use \*LIBL or \*CURLIB.

#### Member

Member concerned; it is possible to use \*FIRST.

#### Connection

Connection – name of the AS400 Connection

### Example

```

lockList est une ASRecordLockCollection
monObjet est un ASRecordLock

nb est un entier
Sablrier(Vrai)
SI ASRecordLocks(lockList, "SP_CUST", "*LIBL", "*FIRST", PrinciConnexion) ALORS
    nb = lockList.CollectionASRecordLock..Occurrence
    POUR nIndex= 1 A nb
        monObjet = lockList.CollectionASRecordLock[nIndice]
        TableauAjoute(gtabAsRecordLock, monObjet)
    FIN
FIN

TableAffiche(TABLE_recordlocks)
Sablrier(Faux)
    
```

## ASObjLocks

List the locks on an object.

## Syntax

```
Result = ASObjLocks(ASObjLockCollection, Connection, ObjectName, LibraryName,
ObjectType, [MemberName], [objPathName],[objASPName])
```

In French: [ASVerrouObjets](#).

## Parameters

### Result

Boolean- True if the command is successful.

### ASObjLockCollection

Variable of type ASObjLockCollection which will contain the results. ASObjLock collection.

### Connection

Connection – name of the AS400 Connection.

### ObjectName

String - Name of the object for which locks will be fetched.

### LibraryName

String - Name of the library containing the object for which locks will be searched.

It is possible to use the value \*LIBL or \*CURLIB.

### ObjectType

String – Type of the object whose locks are being searched. Example: \*FILE, \*LIBL or \*MSGQ..

See Appendix for a complete list of usable types.

### MemberName

String – If applicable, put the name of the member whose locks are being sought.

Possible values:

\*ALL, \*FIRST, \*NONE or the member's name if ObjectType=\*FILE.

\*NONE for other types.

[Optional] if this parameter is not specified then the default value used is \*NONE.

### objPathName

String – [Optional] Physical path of the object whose specific locks are to be known.

If used the object name (**ObjectName**) must be \*OBJPATH, the member's name (**MemberName**) to \*NONE and the library (**LibraryName**) and the type (**ObjectType**) empty.

### objASPName

String – [Optional] Name of the ASP (Auxiliary Storage Pool) where to find the object.

If used: must be \* if **LibraryName**=\*LIBL or \*CURLIB or if **ObjectName**=\*OBJPATH.

If the object is a library or an ASP or if \*SYSBAS is specified, the library must be QSYS.

## Note

If MemberName is not specified and ObjectType=\*FILE, then we will look for locks on the file itself.

If you want the locks on the members of the file, you have to put: ObjectType=\* FILE and MemberName=\*ALL.

## Example

### 1st method to browse the collection result (recommended):

```
sObjname est une chaîne sur 10;
sObjlibname est une chaîne sur 10;
sObjname = "EASYCOMLOG";
sObjlibname = "EASYCOM";
```



```

sFullQualifiedObjName est une chaîne sur 20;
sFullQualifiedObjName = Complète(sObjname,10," ") + Complète(sObjlibname,10," ");
sObjType est une chaîne = "*FILE";
sMemberName est une chaîne = "*ALL";

sResLockSkel est une chaîne = [
    Job Name . . . . . : %1
    Job Username . . . . . : %2
    Job Number . . . . . : %3
    Lock State . . . . . : %4
    Lock Status . . . . . : %5
    Lock Status Desc . . . . . : %6
    Lock Type . . . . . : %7
    Lock Type Desc . . . . . : %8
    Member Name . . . . . : %9
    Share . . . . . : %10
    Share Desc . . . . . : %11
    Lock Scope . . . . . : %12
    Lock Scope Desc . . . . . : %13
    Thread Identifier . . . . . : %14
    Lock %15 on %16
]

lockColl est une ASObjLockCollection ;

H0uvreConnexion(MaConnexionpower8)

//SI PAS ASObjLocks(lockColl, MaConnexionpower8,sObjname,
sObjlibname,sObjType,sMemberName,"","*SYSBAS") ALORS
SI PAS ASObjLocks(lockColl, MaConnexionpower8,sObjname, sObjlibname,sObjType) ALORS
//SI PAS ASVerrousObjets(lockColl, MaConnexionpower8,"TSECOFR",
"QUSRSYS","*MSGQ","","","*SYSBAS") ALORS
//SI PAS ASVerrousObjets(lockColl, MaConnexionpower8,"QCQMONMQ",
"QSVMS","*MSGQ","","","*") ALORS
    Erreur(ErreurInfo(errComplet))
SINON
    nInd est un entier
    nInd = 1
    POUR TOUT objlock DE lockColl

        sResLock est une chaîne;
        sResLock = ChaîneConstruit(sResLockSkel,
        objlock.JobName,
        objlock.jobUsername,
        objlock.JobNumber,
        objlock.LockState,
        objlock.LockStatus,
        objlock.lockStatusDescription,
        objlock.locktype,
        objlock.locktypeDescription,
        objlock.MemberName,
        objlock.share,
        objlock.shareDescription,
        objlock.LockScope,
        objlock.lockScopeDescription,
        objlock.threadId,
        nInd,
        lockColl..CollectionASObjLock..Occurrence)
        Info(sResLock);
        nInd++
    
```

FIN



FIN

### 2nd method to browse the collection result:

```

/**/ Récupéré lock objet **/
sObjname est une chaîne sur 10;
sObjlibname est une chaîne sur 10;
sObjname = "EASYCOMLOG";
sObjlibname = "EASYCOM";
sFullQualifiedObjName est une chaîne sur 20;
sFullQualifiedObjName = Complète(sObjname,10," ") + Complète(sObjlibname,10," ");
sObjType est une chaîne = "*FILE";
sMemberName est une chaîne = "*ALL";

sResLockSkel est une chaîne = [
    Job Name . . . . . : %1
    Job Username . . . . . : %2
    Job Number . . . . . : %3
    Lock State . . . . . : %4
    Lock Status . . . . . : %5
    Lock Status Desc . . . . . : %6
    Lock Type . . . . . : %7
    Lock Type Desc . . . . . : %8
    Member Name . . . . . : %9
    Share . . . . . : %10
    Share Desc . . . . . : %11
    Lock Scope . . . . . : %12
    Lock Scope Desc . . . . . : %13
    Thread Identifier . . . . . : %14
    Lock %15 on %16
]

lockColl est une ASObjLockCollection ;

HOuvreConnexion(MaConnexionpower8)

SI PAS ASObjLocks(lockColl, MaConnexionpower8,sObjname,
sObjlibname,sObjType,sMemberName,"","*SYSBAS") ALORS
//SI PAS ASObjLocks(lockColl, MaConnexionpower8,sObjname, sObjlibname,sObjType) ALORS
//SI PAS ASVerrousObjets(lockColl, MaConnexionpower8,"TSECOFR",
"QUSRSYS","*MSGQ","","","*SYSBAS") ALORS
//SI PAS ASVerrousObjets(lockColl, MaConnexionpower8,"QCQMONMQ",
"QSVMS","*MSGQ","","","*") ALORS
    Erreur(ErreurInfo(errComplet))
SINON

POUR I = 1 À lockColl..CollectionASObjLock..Occurrence
    sResLock est une chaîne;
    objlock est un ASObjLock;
    objlock = lockColl[I];
    sResLock = ChaîneConstruit(sResLockSkel,
    objlock.JobName,
    objlock.jobUsername,
    objlock.JobNumber,
    objlock.LockState,
    objlock.LockStatus,
    objlock.lockStatusDescription,
    objlock.locktype,
    objlock.locktypeDescription,
    objlock.MemberName,

```

```
objlock.share,
objlock.shareDescription,
objlock.LockScope,
objlock.lockScopeDescription,
objlock.threadId,
I,
lockColl..CollectionASObjLock..Occurrence)
Info(=ResLock);
```

FIN

FIN

### Object of type ASObjLockCollection

This type of object is used in return of the [ASObjLocks](#) function.

An ASObjLockCollection contains a certain number of ASObjLock objects.

### Properties of the ASObjLock object

French name of the property	English name of the property	Type
jobname	jobname	string
jobUsername	jobUsername	string
jobNumber	jobNumber	string
lockState	lockState	string
lockStatus	lockStatus	string
lockStatusDescription	lockStatusDescription	string
locktype	locktype	string
locktypeDescription	locktypeDescription	string
memberName	memberName	string
share	share	string
shareDescription	shareDescription	string
lockScope	lockScope	string
lockScopeDescription	lockScopeDescription	string
threadId	threadId	string

### Note

For the following properties: lockStatus, locktype, shareDescription, and lockScope, return raw information and additional description:

LockStatus :

- "The lock is currently held by the job or thread."
- "The job or thread is waiting for the lock (synchronous)."
- "The job or thread has a lock request outstanding for the object (asynchronous)."

LockType:



- "Lock on the object"
- "Lock on the member control block"
- "Lock on the access path used to access a member's data"
- "Lock on the actual data within the member"

Share Description:

- "The file is not shared, the file is a physical file, or the field is not applicable to object type."
- "The file is shared."

Lock Scopes :

- "Job Scope"
  - "Thread Scope"
- "Lock Space Scope"

## Output Queues/Remote Output Queues/Writers functions

### *ASOutputQueueList*

List Output Queues \*OUTQ.

### Syntax

```
Result = ASOutputQueueList(ASOutputQueueCollection, Connection,
[RemoteFilter,OutputQueueName])
```

In French: [ASOutputQueueList](#).

### Parameters

#### Result

Boolean - True if the command was successful, False if an error occurred.  
The ErrorInfo function is used to identify the error.

#### ASOutputQueueCollection

Collection of queues (Output Queues \* OUTQ) that will contain the results.

#### Connection

Connection – connection name.

#### RemoteFilter

Optional. Value among the following constants:

**ASOutQRemoteOnly:** Filter only remote \* OUTQs (Remote Output Tails)

**ASOutQLocalOnly:** Filters only the local \* OUTQs

**ASOutQBothRemoteAndLocal:** No filter, matches the combination

ASOutQRemoteOnly + ASOutQLocalOnly

#### OutputQueueName

Character string (with double quotes) - Name of the output file(s) searched. Optional.

If you do not want to filter by name: leave empty or put "".

If you want to do a rough search (starting with) end with \* (ex: "PRT \*").

### Example

`gTest` est un `ASOutputQueueCollection`

```
SI PAS ASOutputQueueList(gTest, MaConnexion1) ALORS
//SI PAS ASOutputQueueList(gTest, MaConnexion1, ASOutQBothRemoteAndLocal) ALORS
```



```
//SI PAS ASOutputQueueList(gTest, MaConnexion1, ASOutQRemoteOnly) ALORS
//SI PAS ASOutputQueueList(gTest, MaConnexion1, ASOutQLocalOnly) ALORS
//SI PAS ASOutputQueueList(gTest, MaConnexion1, ASOutQBothRemoteAndLocal, "HP*") ALORS
//SI PAS ASOutputQueueList(gTest, MaConnexion1, ASOutQRemoteOnly, "HP*") ALORS
Info(ErreurInfo(errComple))
FIN
```

## ASOutputQueueClear

Empty an Output Queue \*OUTQ.

### Syntax

```
Result = ASOutputQueueClear(ASOutputQueue, Connection)
```

In French : [ASOutputQueueClear](#) .

### Parameters

#### Result

Boolean - True if the command was successful, False if an error occurred.  
The ErrorInfo function is used to identify the error.

#### ASOutputQueue

Output Queue (\*OUTQ).

#### Connection

Connection – connection name.

### Example

```
gTest est un ASOutputQueueCollection
nResSelect est un entier = TableSelect(TABLE_CollectionASOutputQueue)
asoq est une ASOutputQueue = gTest.CollectionASOutputQueue[nResSelect]
//1 : &Oui
//2 : &Annuler
SELON Dialogue("Êtes-vous sûr(e) de vouloir mettre cette file d'attente à blanc?")
// &Oui
CAS 1
    SI PAS ASOutputQueueClear(asoq, MaConnexion1) ALORS
        Info(ErreurInfo(errComple))
    FIN
    Proc_Gestion_ASOutQ(nResSelect)
// &Annuler
CAS 2
    RETOUR
FIN
```

## ASOutputQueueRelease

Release an Output Queue (\*OUTQ).

### Syntax

`Result = ASOutputQueueRelease(ASOutputQueue, Connection)`

In French : [ASOutputQueueRelease](#) .

## Parameters

### Result

Boolean - True if the command was successful, False if an error occurred.  
The ErrorInfo function is used to identify the error.

### ASOutputQueue

Output Queue (\*OUTQ).

### Connection

Connection – connection name.

## Example

```
gTest est un ASOutputQueueCollection
nResSelect est un entier = TableSelect(TABLE_CollectionASOutputQueue)
asoq est une ASOutputQueue = gTest.CollectionASOutputQueue[nResSelect]
SI PAS ASOutputQueueRelease(asoq, MaConnexion1) ALORS
    Info(ErreurInfo(errComple))
FIN
```

## ASOutputQueueHold

Hold an Output Queues (\*OUTQ).

## Syntax

`Result = ASOutputQueueHold(ASOutputQueue, Connection)`

In French : [ASOutputQueueHold](#) .

## Parameters

### Result

Boolean - True if the command was successful, False if an error occurred.  
The ErrorInfo function is used to identify the error.

### ASOutputQueue

Output Queue \*OUTQ.

### Connection

Connection – connection name.

## Example

```
gTest est un ASOutputQueueCollection
nResSelect est un entier = TableSelect(TABLE_CollectionASOutputQueue)
asoq est une ASOutputQueue = gTest.CollectionASOutputQueue[nResSelect]
SI PAS ASOutputQueueHold(asoq, MaConnexion1) ALORS
    Info(ErreurInfo(errComple))
FIN
```

## ASWriterInfo

Allows you to retrieve information about a writer.

### Syntax

```
Result = ASWriterInfo(ASWriterDetailed, Connection, printerName, writerName)
```

In French : [ASWriterInfo](#).

### Parameters

#### Result

Boolean - True if the command was successful, False if an error occurred.

The ErrorInfo function is used to identify the error.

#### ASWriterDetailed

Detailed writer.

#### Connection

Connection – connection name.

#### printerName

Character string (with double quotes) - Name of the printer you are looking for its writer details.

#### writerName

Character string (with double quotes) - Name of the writer you are looking for details.

### Example

```
aswtr est un ASWriterDetailed
SI PAS ASWriterInfo(aswtr, MaConnexion1, "HP", "HP") ALORS
    Info(ErreurInfo(errComple))
FIN
```

## ASWriterStart

Start an editor. Equivalent to STRPRTWTR.

### Syntax 1

```
Result = ASWriterStart(ASWriter, ASOutputQueue, Connection)
```

In French: [ASWriterStart](#).

### Parameters

#### Result

Boolean - True if the command was successful, False if an error occurred.

The ErrorInfo function is used to identify the error.

#### ASWriter

Writer.

#### ASOutputQueue

Ouput Queue (\*OUTQ).

#### Connection

Connection – connection name.

## Syntax 2

```
Result = ASWriterStart(ASWriterDetailed, Connection)
```

In French: [ASWriterStart](#).

## Parameters

### Result

Boolean - True if the command was successful, False if an error occurred.

The ErrorInfo function is used to identify the error.

### ASWriterDetailed

Detailed writer.

### Connection

Connection – connection name.

## Example

```
gTest est un ASOutputQueueCollection
nResSelectedQ est un entier = TableSelect(TABLE_CollectionASOutputQueue)
nResSelectedWtr est un entier = TableSelect(TABLE_CollectionASWriter)
asoq est une ASOutputQueue = gTest[nResSelectedQ]
aswtr est un ASWriterDetailed = asoq.WriterCollection[nResSelectedWtr]
SI PAS ASWriterStart(aswtr, MaConnexion1) ALORS
    Info(ErreurInfo(errComplet))
FIN
```

## ASWriterEnd

Stops a writer (works with a remote writer too). Equivalent to an ENDWTR.

## Syntax 1

```
Result = ASWriterEnd(ASWriter, Connection, [stopOption])
```

In French: [ASWriterEnd](#).

## Parameters

### Result

Boolean - True if the command was successful, False if an error occurred.

The ErrorInfo function is used to identify the error.

### ASWriter

Writer.

### Connection

Connection – connection name.

### StopOption

Optional. Value among the following constants:

**ASCntId**: \* CNTLD. Default value.

**ASImmed**: \* IMMED

**ASPageEnd**: \* PAGEEND

## Syntax 2

```
Result = ASWriterEnd(ASWriterDetailed, Connection, [stopOption])
```



In French: [ASWriterEnd](#).

## Paramètres

### Result

Boolean - True if the command was successful, False if an error occurred.  
The ErrorInfo function is used to identify the error.

### ASWriterDetailed

Writer detailed.

### Connection

Connection – connection name.

### StopOption

Optional. Value among the following constants:

**ASCntId**: \* CNTRLD. Default value.

**ASImmed**: \* IMMED

**ASPageEnd**: \* PAGEEND

## Example

`gTest` est un `ASOutputQueueCollection`

`nResSelectedQ` est un `entier` = `TableSelect(TABLE_CollectionASOutputQueue)`

`nResSelectedWtr` est un `entier` = `TableSelect(TABLE_CollectionASWriter)`

`aswtr` est un `ASWriterDetailed` =

`gTest[nResSelectedQ].WriterCollection[nResSelectedWtr]`

```
//SI PAS ASWriterEnd(aswtr, MaConnexion1, ASCntId) ALORS
//SI PAS ASWriterEnd(aswtr, MaConnexion1) ALORS
//SI PAS ASWriterEnd(aswtr, MaConnexion1, ASPageEnd) ALORS
SI PAS ASWriterEnd(aswtr, MaConnexion1, ASImmed) ALORS
    Info(ErreurInfo(errComple))
FIN
```

## ASWriterRestart

Restarts an editor. Equivalent to an ENDWTR then a status check until stop proof and then a STRPRTWTR.

### Syntax 1

```
Result = ASWriterRestart (ASWriter, ASOutputQueue, Connection, [stopOption])
```

In French: [ASWriterRestart](#).

## Parameters

### Result

Boolean - True if the command was successful, False if an error occurred.  
The ErrorInfo function is used to identify the error.

### ASWriter

Writer.

### ASOutputQueue

Output Queue \*OUTQ.

### Connection





Connection – connection name.

**StopOption**

Optional. Value among the following constants:

**ASCntId:** \* CNTLD. Default value.

**ASImmed:** \* IMMED

**ASPageEnd:** \* PAGEEND

**Syntax 2**

`Result = ASWriterRestart(ASWriterDetailed, Connection, [stopOption])`

In French: `ASWriterRestart.`

**Parameters**

**Result**

Boolean - True if the command was successful, False if an error occurred.

The ErrorInfo function is used to identify the error.

**ASWriterDetailed**

Writer detailed.

**Connection**

Connection – connection name.

**StopOption**

Optional. Value among the following constants:

**ASCntId:** \* CNTLD. Default value.

**ASImmed:** \* IMMED

**ASPageEnd:** \* PAGEEND

**Example**

`gTest` est un `ASOutputQueueCollection`

`nResSelectedQ` est un `entier` = `TableSelect(TABLE_CollectionASOutputQueue)`

`nResSelectedWtr` est un `entier` = `TableSelect(TABLE_CollectionASWriter)`

`asoq` est une `ASOutputQueue` = `gTest[nResSelectedQ]`

`aswtr` est un `ASWriterDetailed` = `asoq.WriterCollection[nResSelectedWtr]`

SI PAS `ASWriterRestart(aswtr, MaConnexion1)` ALORS

`Info(ErreurInfo(errComplet))`

FIN

***ASWriterStartRemote***

Allows to start a remote writer.

**Syntax 1**

`Result = ASWriterStartRemote(ASWriter, ASOutputQueue, Connection)`

In French: `ASWriterStartRemote.`

**Parameters**

**Result**



Boolean - True if the command was successful, False if an error occurred.  
 The ErrorInfo function is used to identify the error.

**ASWriter**

Writer.

**ASOutputQueue**

Ouput Queue \*OUTQ.

**Connection**

Connection – connection name.

**Syntax 2**

```
Result = ASWriterStartRemote(ASWriterDetailed, Connection)
```

In French : [ASWriterStartRemote](#).

**Parameters**

**Result**

Boolean - True if the command was successful, False if an error occurred.  
 The ErrorInfo function is used to identify the error.

**ASWriterDetailed**

Detailed writer.

**Connection**

Connection – connection name.

**Example**

```
gTest est un ASOutputQueueCollection

nResSelectedQ est un entier = TableSelect(TABLE_CollectionASOutputQueue)
nResSelectedWtr est un entier = TableSelect(TABLE_CollectionASWriter)
asoq est une ASOutputQueue = gTest[nResSelectedQ]

aswtr est un ASWriterDetailed = asoq.WriterCollection[nResSelectedWtr]

SI PAS ASWriterStartRemote(aswtr, MaConnexion1) ALORS
    Info(ErreurInfo(errComple))
FIN
```

**ASWriterRestartRemote**

Restarts a remote writer.

Use if you want to restart a remote writer because if you stop it, you will not be able to restart it.

**Syntax 1**

```
Result = ASWriterRestartRemote (ASWriter, ASOutputQueue, Connection, [stopOption])
```

In French : [ASWriterRestartRemote](#).

**Parameters**

**Result**



Boolean - True if the command was successful, False if an error occurred.  
 The ErrorInfo function is used to identify the error.

**ASWriter**

Writer.

**ASOutputQueue**

Ouput Queue \*OUTQ.

**Connection**

Connection – connection name.

**StopOption**

Optional. Value among the following constants:

**ASCntld:** \* CNTLD. Default value.

**ASImmed:** \* IMMED

**ASPageEnd:** \* PAGEEND

## Syntax 2

```
Result = ASWriterRestartRemote(ASWriterDetailed, Connection, [stopOption])
```

In French: [ASWriterRestartRemote](#).

## Parameters

**Result**

Boolean - True if the command was successful, False if an error occurred.  
 The ErrorInfo function is used to identify the error.

**ASWriterDetailed**

Writer detailed.

**Connection**

Connection – connection name.

**StopOption**

Optional. Value among the following constants:

**ASCntld:** \* CNTLD. Default value.

**ASImmed:** \* IMMED

**ASPageEnd:** \* PAGEEND

## Example

`gTest` est un `ASOutputQueueCollection`

```
nResSelectedQ est un entier = TableSelect(TABLE_CollectionASOutputQueue)
```

```
nResSelectedWtr est un entier = TableSelect(TABLE_CollectionASWriter)
```

```
asoq est une ASOutputQueue = gTest[nResSelectedQ]
```

```
aswtr est un ASWriterDetailed = asoq.WriterCollection[nResSelectedWtr]
```

```
SI PAS ASWriterRestartRemote(aswtr, MaConnexion1,ASImmed) ALORS
```

```
Info(ErreurInfo(errComple))
```

```
FIN
```

### Objet of ASOutputQueueCollection type

This type of object is used in return of the `ASOutputQueueList` function. This is a collection of `ASOutputQueue` objects.



### Properties of the ASOutputQueue object

Property name	Description	AS400 Type
OutputQueueName	Output queue name	CHAR(10)
OutputQueueLibName	Output queue library name	CHAR(10)
fileOrder	Order of files on queue	CHAR(10)
displayAnyFile	Display any file	CHAR(10)
jobSeparator	Job separators	BINARY(4)
operatorControlled	Operator controlled	CHAR(10)
dataQName	Data queue name	CHAR(10)
dataQLibName	Data queue library name	CHAR(10)
AuthToCheck	Authority to check	CHAR(10)
NbOfFiles	Number of files	BINARY(4)
OutputQueueStatus	Output queue status	CHAR(10)
TextDescription	Text description	CHAR(50)
NbSpoolFilePageSpec	Number of spooled file pages specified	BINARY(4)
NbWritersStartedToQ	Number of writers started to queue	BINARY(4)
NbWritersToAutoStart	Writers to autostart	BINARY(4)
RemoteSysNameType	Remote system name type	CHAR(1)
RemoteSysName	Remote system name	CHAR(255)
RemotePrinterQ	Remote printer queue	CHAR(128)
MsgQName	Message queue name	CHAR(10)
MsgQLibName	Message queue library name	CHAR(10)
ConnectionType	Connection type	BINARY(4)
DestinationType	Destination type	BINARY(4)
VMMVSClass	VM/MVS class	CHAR(1)
FormControlBuff	Forms control buffer	CHAR(8)
HostPrintTransform	Host print transform	CHAR(1)
ManufacturerTypeAndModel	Manufacturer type and model	CHAR(17)
WorkstationCustomObjName	Workstation customizing object name	CHAR(10)
WorkstationCustomObjLibName	Workstation customizing object library name	CHAR(10)
SpoolASPAttribute	Spooled file auxiliary storage pool attribute	CHAR(1)
NbPageEtriesReturned	Number of page entries returned	BINARY(4)
DestinationOptions	Destination options	CHAR(128)
WriterTypeStartedQueue	Writer type started to queue	CHAR(1)
PrintSeparatorPage	Print separator page	CHAR(1)
LongRemotePrinterQueue	Long remote printer queue	CHAR(255)
ImgConf	Image configuration	CHAR(10)
ImgConfLib	Image configuration library	CHAR(10)

NetDirPublishingStatus	Network directory publishing status	CHAR(1)
SpooledFileAspId	Spooled file auxiliary storage pool ID	BINARY(4)
SpooledFileAspDeviceName	Spooled file auxiliary storage pool device name	CHAR(10)
SplFileMaxPageEntries	Spooled file maximum page entries	CHAR(*)

**Objet of ASWriterCollection type**

This is a collection of **ASWriter** objects.

**Properties of the ASWriter object**

Property name	Description	AS400 Type
WriterJobName	Writer job name	CHAR(10)
WriterUserName	Writer job user name	CHAR(10)
WriterJobNumber	Writer job number	CHAR(6)
WriterJobStatus	Writer job status	CHAR(10)
PrinterDeviceName	Printer device name	CHAR(10)

**Objet of ASWriterDetailedCollection type**

This is a collection of **ASWriterDetailed** objects.

**Properties of the ASWriterDetailed object**

Property name	Description	AS400 Type
startedbyUser	Started by user	CHAR(10)
writingStatus	Writing status	CHAR(1)
waitingForMsgStatus	Waiting for message status	CHAR(1)
heldStatus	Held status	CHAR(1)
endPendingStatus	End pending status	CHAR(1)
holdPendingStatus	Hold pending status	CHAR(1)
betweenFileStatus	Between file status	CHAR(1)
betweenCopiesStatus	Between copies status	CHAR(1)
waitingForDataStatus	Waiting for data status	CHAR(1)
waitingForDeviceStatus	Waiting for device status	CHAR(1)
onJobQueueStatus	On job queue status	CHAR(1)
typeOfWriter	Type of writer	CHAR(1)
writerJobName	Writer job name	CHAR(10)
writerJobUserName	Writer job user name	CHAR(10)
writerJobNumber	Writer job number	CHAR(6)
printerDeviceType	Printer device type	CHAR(10)

nbOfSeparator	Number of separators	BINARY(4)
drawerForSeparators	Drawer for separators	BINARY(4)
alignForms	Align forms	CHAR(10)
outputQueueName	Output queue name	CHAR(10)
outputQueueLibName	Output queue library name	CHAR(10)
outputQueueStatus	Output queue status	CHAR(1)
formType	Form type	CHAR(10)
msgOption	Message option	CHAR(10)
autoEndWriter	Automatically end writer	CHAR(10)
allowDirectPrinting	Allow direct printing	CHAR(10)
msgQName	Message queue name	CHAR(10)
msgQLibName	Message queue library name	CHAR(10)
changesTakeEffect	Changes take effect	CHAR(10)
nextOuputQueueName	Next ouput queue name	CHAR(10)
nextOuputQueueLibName	Next ouput queue library name	CHAR(10)
nextFormtype	Next form type	CHAR(10)
nextMessageOption	Next message option	CHAR(10)
nextFileSeparators	Next file separators	BINARY(4)
nextSeparatorDrawers	Next separator drawer	BINARY(4)
spoolFileName	Spooled file name	CHAR(10)
jobName	Job name	CHAR(10)
userName	User name	CHAR(10)
jobNumber	Job number	CHAR(6)
spoolFileNumber	Spooled file number	BINARY(4)
pageBeingWritten	Page being written	BINARY(4)
totalPages	Total pages	BINARY(4)
copiesLeftToProduce	Copies left to produce	BINARY(4)
totalCopies	Total copies	BINARY(4)
messageKey	Message key	CHAR(4)
initPrinter	Initialize printer	CHAR(1)
printerDeviceName	Printer device name	CHAR(10)
jobSystemName	Job system name	CHAR(8)
spoolFileCreateDate	Spooled file create date	CHAR(7)
spoolFileCreateTime	Spooled file create time	CHAR(6)

## Files management functions

### *ASOpenFileInfoList*

Used to generate a list of \*FILE objects which are currently open in the connection (current JOB), in a specific JOB, or in the thread of the JOB specified in parameter.

### Syntax

```
Result = ASOpenFileInfoList(ASOpenFileInfoCollection, Connection, [QualifiedJobID, threadID])
```

In French: [ASOpenFileInfoList](#).

## Parameters

### Result

Boolean - True if the command was successful, False if an error occurred.  
The `ErrorInfo` function is used to identify the error.

### ASOpenFileInfoCollection

Collection of `ASOpenFileInfo` which will contain the results for each file.

### Connection

Connection – connection name.

### QualifiedJobID

Optional. Character string (with quotes).  
Qualified JOB name (3 elements): `jobNumber/username/jobname`.

### ThreadID

Optional. Thread number (integer) of the JOB specified within `QualifiedJobID` parameter.

## Example

```
// Current JOB
asopnfilecollec est une ASOpenFileInfoCollection

SI PAS ASOpenFileInfoList(asopnfilecollec, MaConnexion) ALORS
    Info(ErreurInfo(errComplet))
SINON
    I est un entier = 1
    POUR TOUT asopnf DE asopnfilecollec
        Trace(I, " ", asopnf.fileName, asopnf.threadId)
        I = I+1
    FIN
FIN

// Specific JOB
asopnfilecollec2 est une ASOpenFileInfoCollection

SI PAS ASOpenFileInfoList(asopnfilecollec2, MaConnexion, "653011/QSYS/QINTER") ALORS
    Info(ErreurInfo(errComplet))
SINON
    J est un entier = 1
    POUR TOUT asopnf2 DE asopnfilecollec2
        Trace(J, " ", asopnf2.fileName + " FROM " + asopnf2.mbrOrDeviceName,
asopnf2.threadId)
        J = J+1
    FIN
FIN

// Thread of a specific JOB
asopnfilecollec3 est une ASOpenFileInfoCollection

SI PAS ASOpenFileInfoList(asopnfilecollec3, MaConnexion, "653011/QSYS/QINTER",
Val("2")) ALORS
    Info(ErreurInfo(errComplet))
SINON
    K est un entier = 1
    POUR TOUT asopnf3 DE asopnfilecollec3
        Trace(K, " ", asopnf3.fileName, asopnf3.threadId)
        K = K+1
```

FIN  
FIN

**Objet of ASOpenFileInfoCollection type**

This type of object is used in return of the [ASOpenFileInfoList](#) function. This is a collection of **ASOpenFileInfo** objects.

**Properties of the ASOpenFileInfo object**

Property name	Description	AS400 type
fileName	File name	CHAR(10)
fileLib	File library	CHAR(10)
mbrOrDeviceName	Member or device name	CHAR(10)
fileType	File type	CHAR(10)
recordFormat	Record format	CHAR(10)
activationGrpName	Activation group name	CHAR(10)
threadId	Thread identifier	CHAR(8)
openOption	Open option	CHAR(1)
activationGrpNumber	Activation group number	BINARY(8)
writeCount	Write count	BINARY(8)
readCount	Read count	BINARY(8)
readWriteCount	Write/read count	BINARY(8)
otherIoCount	Other I/O count	BINARY(8)
rrn	Relative record number	BINARY(8)
nbSharedOpens	Number of shared opens	BINARY(8)
objAuxStoragePoolNumber	Object auxiliary storage pool number	BINARY(4)
libAuxStoragePoolNumber	Library auxiliary storage pool number	BINARY(4)
objAuxStoragePoolName	Object auxiliary storage pool name	CHAR(10)





libAuxStoragePoolName	Library auxiliary storage pool name	CHAR(10)
-----------------------	-------------------------------------	----------

## AS400 Info and Errors Messages (CPA, CPF...) Functions

### ASMessageReceive

Retrieve an info or error message in a message queue via its key (Filled an ASMessage).

#### Syntax

```
Result = ASMessageReceive(ASMessage, MessageQueueName, MessageQueueLib, MessageKey, Connection)
```

In French: [ASMessageReceive](#).

#### Parameters

##### Result

Boolean - True if the operation was performed, False if a problem occurred.

The ErrorInfo function is used to identify the error.

##### ASMessage

AS/400 - IBMi message.

##### MessageQueueName

Character string (with quotes) - Name of the message queue of the message \*MSGQ.

##### MessageQueueLib

Character string (with quotes) - LIB of the message queue of the message \*MSGQ.

##### MessageKey

Integer - Message key identifier.

##### Connection

Connection – connection name.

#### Example

```
asog est une ASOutputQueue = gTest[nResSelectedQ]
aswtr est un ASWriterDetailed = asog.WriterCollection[nResSelectedWtr]
SI aswtr.messageKey <> 0 ET aswtr.messageKey <> -1 ALORS
    asmsg est un ASMessage

    SI PAS ASMessageReceive(asmsg, aswtr.msgQName, aswtr.msgQLibName,
        aswtr.messageKey, MaConnexion1) ALORS
        Info(ErreurInfo(errComple))
        RETOUR

FIN
```

### ASMessageRetrieve

Allows you to find an AS/400 message skeleton with its parameters, sizes and types (Filled ASMessageHelp).

#### Syntax 1

```
Result = ASMessageRetrieve(ASMessageHelp, MessageIdentifier, Connection)
```

## Syntax 2

```
Result = ASMessageRetrieve(ASMessageHelp, MessageIdentifier, MessageFileName, MessageLib, Connection)
```

## Syntax 3

```
Result = ASMessageRetrieve(ASMessageHelp, MessageIdentifier, MessageFileName, MessageLib, displayFormattingChars, Connection)
```

In French: [ASMessageRetrieve](#).

## Parameters

### Result

Boolean - True if the operation was performed, False if a problem occurred.  
The ErrorInfo function is used to identify the error.

### ASMessageHelp

AS/400 - IBM i Message skeleton.

### MessageIdentifier

Character string (with quotes) - Identifying code of the message (e.g., CPF1234).

### MessageFileName

Character string (with quotes) – Name of the File containing the message.

### MessageLib

Character string (with quotes) – LIB of the File containing the message.

### Connection

Connection – connection name.

## Example

```
asmsgH est un ASMessageHelp  
SI PAS ASMessageRetrieve(asmsgH, "CPF1124", MaConnexion1) ALORS  
Info(ErreurInfo(errComplet))  
RETOUR
```

## ASMessageReply

Allows you to reply to a message received via its message queue and its key (ASMessage received via ASMessageReceive) with the possibility of having retrieved a list of possible answers, the default answer, etc. via ASMessageHelp.

## Syntax

```
Result = ASMessageReply(ASMessage, Reply, MessageQueueName, MessageQueueLib, MessageKey, removeMessage, Connection)
```

In French: [ASMessageReply](#).

## Parameters

### Result

Boolean - True if the operation was performed, False if a problem occurred.  
The ErrorInfo function is used to identify the error.

### ASMessage



Message AS/400 - IBMi.

**Reply**

Character string (with quotes) - Reply to send for the message.

**MessageQueueName**

Character string (with quotes) - Name of the message queue of the message \*MSGQ.

**MessageQueueLib**

Character string (with quotes) - LIB of the message queue of the message.

**MessageKey**

Integer - Message key identifier.

**removeMessage**

Boolean - TRUE if the message must be removed from the message queue after the reply, FALSE otherwise.

**Connection**

Connection – connection name.

**Example**

`gAsmsg` est un `ASMessage`

```
SI PAS ASMessageReply(gAsmsg, reply, "QSYSOPR", "QSYS", gAsmsg.messageKey, Faux,
MaConnexion1) ALORS
    Info(ErreurInfo(errComplet))
SINON
```

**ASMessageHistoryLog**

Allows you to list and filter the history of any message stored on the AS/400.

**Syntax 1**

```
Result = ASMessageHistoryLog (ASMessageHistoryLogCollection, Connexion)
```

**Syntax 2**

```
Result = ASMessageHistoryLog (ASMessageHistoryLogCollection,
ASMessageHistoryLogFilter, Connexion)
```

In French : `ASMessageHistoryLog`.

**Parameters**

**Result**

Boolean - True if the operation was performed, False if a problem occurred.  
The `ErrorInfo` function is used to identify the error.

**ASMessageHistoryLogCollection**

Collection of [ASMessageHistoryLog](#).

**ASMessageHistoryLogFilter**

Object to filter desired logs to be returned.

**Connection**

Connection – connection name.

**Example**

```
asmsgHlc est un ASMessageHistoryLogCollection
SI PAS ASMessageHistoryLog(asmsgHlc, MaConnexion1) ALORS
```



```
Info(ErreurInfo(errComplet))  
RETOUR
```

### *ASErrorHelp and ASErrorData*

These functions return extended information about AS/400 error message.

**ASErrorHelp** returns the complete formatted error message.

**ASErrorData** returns the parameter data to know the position and length of the message data, see commands DSPMSGD or WRKMSGF on AS/400.

Information returned completes information returned by [ErrorInfo](#) et [HErrorInfo](#),  
See [Errors management](#).

### Syntax

```
Result = ASErrorHelp([Connection])  
Result = ASErrorData(Position, Length [,Connection])
```

In French: [ASErreurAide](#) et [ASErreurDonnee](#)

### Parameters

#### Result

Character string: Result information.

#### Position

Integer: Offset in the message data to retrieve.

#### Length

Integer: Length of the data to retrieve.

#### Connection (optional)

Connection to AS/400.

### Example

Retrieve message text and message data on a constraint error.

```
s1 is strings  
constr_name is string  
constr_parmfil, constr_parmlib is string  
constr_fil, constr_lib is string  
  
s1 = HErrorInfo(hErrMessage)  
IF ExtractString(s1, 6, CR) = "Message: CPF503A" THEN  
  // Nom de la contrainte  
  constr_name = NoSpace(ASErrorData(176, 258))  
  constr_parmfil = NoSpace(ASErrorData(448, 10))  
  constr_parmlib = NoSpace(ASErrorData(458, 10))  
  constr_fil = NoSpace(ASErrorData(10, 10))  
  constr_lib = NoSpace(ASErrorData(20, 10))  
  Info("Error on constraint: "+constr_name+CR+"Parent File:  
"+constr_parmlib+"/"+constr_parmfil+CR+"File: "+constr_lib+"/"+constr_fil)
```



END

```
Info("AS/400 Error:" +Middle(ExtractString(s1, 7, CR),14)+CR+"Help:"+CR+
ASErrorHelp())
```

### Objet of ASMessage type

AS/400 message or error message (29 properties).

### Properties of the ASMessage object

Property name	Description	AS400 type
messageSeverity	Message severity	BINARY(4)
messageIdentifier	Message identifier	CHAR(7)
messageType	Message type	CHAR(2)
messageKey	Message key	CHAR(4)
messageFileName	Message file name	CHAR(10)
messageFileLibSpec	Message file library specified	CHAR(10)
messageFileLibUsed	Message file library used	CHAR(10)
sendingJob	Sending job	CHAR(10)
sendingJobUserProfile	Sending job's user profile	CHAR(10)
sendingJobNumber	Sending job's number	CHAR(6)
sendingPgmName	Sending program name	CHAR(12)
dateSent	Date sent	CHAR(7)
timeSent	Time sent	CHAR(6)
microSecondes	Microseconds	CHAR(6)
sendingUserProfile	Sending user profile	CHAR(10)
textCCSID	CCSID conversion status indicator for text	BINARY(4)
dataCCSID	CCSID conversion status indicator for data	BINARY(4)
alertOption	Alert option	CHAR(9)
msgOrMsgHelpCCSID	CCSID of message or message help	BINARY(4)
rplcmtDataOrImpromptuMsgTxtCCSID	CCSID of replacement data or impromptu message text	BINARY(4)
RplcmtDataOrImpromptuMSgtxtLenRet	Length of replacement data or impromptu message text returned	BINARY(4)
RplcmtDataOrImpromptuMSgtxtLenAvail	Length of replacement data or impromptu message text available	BINARY(4)
msgReturnedLen	Length of message returned	BINARY(4)

msgAvailableLen	Length of message available	BINARY(4)
msgHlpRetLen	Length of message help returned	BINARY(4)
msgHlpAvailableLen	Length of message help available	BINARY(4)
rplcmtDataOrImpromptuTxt	Replacement data or impromptu text	CHAR(*)
message	Message	CHAR(*)
messageHelp	Message help	CHAR(*)

### *Objet de type ASMessageHelp*

AS/400 message skeleton with positional parameters, their types, and their sizes (32 properties including 2 parameter tables).

### **Properties of the ASMessageHelp object**

Property name	Description	AS400 type
messageSeverity	Message severity	BINARY(4)
alertIndex	Alert index	BINARY(4)
alertOption	Alert option	CHAR(9)
logIndicator	Log indicator	CHAR(1)
messageID	Message ID	CHAR(7)
nbOfSubstitutionVars	Number of substitution variable formats	BINARY(4)
CCSIDconversionStatusIndicText	CCSID conversion status indicator of text	BINARY(4)
CCSIDconversionStatusIndicRplcmtData	CCSID conversion status indicator of replacement data	BINARY(4)
CCSIDTextReturned	CCSID of text returned	BINARY(4)
replyType	Reply type	CHAR(10)
maxReplyLen	Maximum reply length	BINARY(4)
maxReplyDecimalPosition	Maximum reply decimal positions	BINARY(4)
nbValidReplyValues	Number of valid reply values entries returned	BINARY(4)
nbSpecialValidReplyValues	Number of special reply values returned	BINARY(4)
messageCreationDate	Message creation date	CHAR(7)
messageCreationLevelNumber	Message creation level number	BINARY(4)
messageModificationDate	Message modification date	CHAR(7)
messageModificationLevelNumber	Message modification level number	BINARY(4)
storedMessageCCSID	Stored CCSID of message	BINARY(4)

nbDumpListEntries	Number of dump list entries returned	BINARY(4)
defaultPgmName	Default program name	CHAR(10)
defaultPgmLibName	Default program library name	CHAR(10)
defaultReply	Default reply	CHAR(*)
message	Message	CHAR(*)
messageHelp	Message help	CHAR(*)
subVarFormat	Substitution variable formats	CHAR(*)
validReplies	Valid reply value entries	CHAR(*)
specialValidReplies	Special reply value entries	CHAR(*)
lowerRangeReplyValues	Lower range reply value	CHAR(*)
upperRangeReplyValues	Upper range reply value	CHAR(*)
relationalTestEntryFormat	Relational test entry	CHAR(*)
dumpListEntries	Dump list entries	CHAR(*)

### Objet of *ASMessageHistoryLogCollection* type

This type of object is used as parameter of the [ASMessageHistoryLog](#) function. This is a collection of **ASMessageHistoryLog** objects.

### Properties of the *ASMessageHistoryLog* object

Property name	Description	AS400 Type
msgSeverity	Message Severity	BINARY(4)
msgId	Message ID	CHAR(7)
msgTypeId	Message Type ID (01, 02, 03...)	CHAR(2)
msgType	Message Type as text (e.g. for 01 : Completion, 02: Diagnostic, ...)	CHAR(10)
msgFileName	Message File Name	CHAR(10)
msgFileLib	Message File Library	CHAR(10)
dateSent	Date sent	CHAR(7)
timeSent	Time Sent	CHAR(6)
fromJob	From Job	CHAR(10)
fromJobUser	From Job user	CHAR(10)
fromJobNumber	From Job number	CHAR(6)
fromUser	From User	CHAR(10)
dataStatus	Data Status (OK, Unauthorized, Damaged, ...)	CHAR(1)
firstLevelTxt	First level Text message	CHAR(*)

### Objet of *ASMessageHistoryLogFilter* type

This type of object is used as parameter of the [ASMessageHistoryLog](#) function.

### Properties of the ASMessageHistoryLogFilter object

Property name	Description	AS400 Type
startDateLog	Start date	CHAR(10)
startTimeLog	Start time (accurate to microsecond)	CHAR(10)
endDateLog	End date	CHAR(10)
endTimeLog	End time (accurate to microsecond)	CHAR(10)
iMsgSeverity	Specific message severity (from 0 to 99)	BINARY(4)
arrMsgIds	Specific message IDs (up to 100 message IDs, e.g. CPF1124)	CHAR(*)
bOmitMsgIds	If (previously defined) message IDs list should be SELECTED or OMITTED	CHAR(*)
arrJob	Specific JOBS, up to 5 JOB filters (could be defined by : jobname, username/jobname or jobnumber/username/jobname)	CHAR(*)
arrMsgTypes	Specific message types list, up to 9 types (9/9) (Message status likes : *COMP Completion, *RPY Reply, ...)	CHAR(*)
bOmitMsgTypes	If (previously defined) message types list should be SELECTED or OMITTED	CHAR(*)

### Objects management functions

#### *ASObjectList*

List all the objects of an AS400 library.

#### Syntax

```
Result = ASobjetsList(ASObjetCollection, Library, Name, Type, Connection)
```

#### Parameters

##### Result

Boolean – true if the command was successful.

##### ASObjectCollection

ASObjectCollection variable which will contain the results.

##### Library

Filter using a Library name, special values allowed are \*ALL, \*ALLUSR, \*CURLIB, \*LIBL, \*USRLIBL. Per default, \*LIBL is used.





**Name**

Filter using the name of the object, special values allowed are \*ALL, \*ALLUSR, \*IBM. Per default, \*ALL is used.

**Type**

Filter using the type of object, special value allowed: \*ALL. Per default, \*ALL is used.

**Connection**

Connection – name of connection.

## Example

`listJob` is a `ASObjectCollection`

```
//per default behaviour, list all objects in *LIBL.  
IF NOT ASObjetsList(listJob,"", "", "", MaConnection1)  
  Errorr(ErrorrInfo())  
END  
  
// list all programs into the CR library  
IF NOT ASObjetsList(listJob,"CR", "", "*PGM", MaConnexion1)  
  Erreur(ErreurInfo(errComple))  
FIN
```

## Debug and error management

### Errors management

Errors are mainly managed by the WinDev functions:

`HError`, `HErrorDuplicates`, `HErrorInfo`, `HErrorIntegrity`, `HErrorLock`,  
`HErrorModification`, `HErrorPassword`, `HErrorStatusModification`

### WRKACTJOB

Using AS/400 command WRKACTJOB, you can see the job properties for the connection, and get some additional information on errors:

- Job log,
- Open files,
- Locked files and records,

### Traces

By generating a Trace File on Easycom server, you can see native error messages raised by the system.

### CPF messages

Functions `ASErrorHelp` and `ASErrorData` return the native AS/400 error message (CPFxxxx).

### ErrorInfo et HErrorInfo

`hError` returns the last code WinDev raised.

Error code **73001** is returned in case of AS/400 native access error. Use `ErrorInfo` or `hErrorInfo` to get details, with `hErrNative` or `hErrNativeMessage` parameters.



## How to diagnose errors

In case of [connectivity errors](#), there are several ways to search:

- in the EASYCOMD job history. To see it, do WRKACTJOB, then option 5 on EASYCOMD, then option 10 (job's history), and type F10. Type F1 on the suspicious messages to get more information.
- In the EACMSGQ messages. To see it, enter DSPMSG EASYCOM/EACMSGQ.
- In the QSYSOPR messages. To see it enter DSPMSG QSYSOPR. Unexpected, failures or licensing messages will appear here.
- In the QEZJOBLOG OutQ. A spool file is generated in this outq if Easycom failed to start properly (error -4 on the client), or if the easycom job unexpectedly stops.

To see the spool file, do the following commands:

- WRKOUTQ OUTQ(QEZJOBLOG)
- Type F18 (to go at the end), and then F11.
- There should be a line with the station name, with the corresponding user, date and time.

Type 5 on then entry to display the spool. It contains information, warning and error messages of the job.

- In the LOGFILE file, LOGFILE member in the EASYCOM library. This file will contain all TCP/IP failures, with OS/400 errors codes. To see that file, use: DSPPFM EASYCOM/LOGFILE MBR(LOGFILE)

This file can be downloaded using FTP or [Easycom configuration](#) client on Windows.

**In case of errors during processing**, an Easycom logfile can be useful. It can be setup using [CFGEAC](#) or [Easycom configuration](#). The contents of this file can help to understand what is performed on the server, see parameters, additional error messages, ...

The Easycom job history can also help a lot. To see it, use WRKACTJOB, option 5, then 10 and type F10.

If the Easycom job stops too quickly to see a job history, use JOBLOG(\*YES) in [CFGEAC](#) command to setup EASYCOM to always have an spool generated in QEZJOBLOG (see above to consult it).

If the Easycom job aborts unexpectedly there should be a spool file in QEZJOBLOG (see above).

If the EASYCOM or the EASYCOMD job fails in a loop, try to see what is involved in the call stack. It is available by using WRKACTJOB, then option 5, then option 11.

**In case of licensing errors**, do DSPMSG QSYSOPR if the information provided on the client is not sufficient.

## Easycom Trace File

Trace file enables EASYCOM carried out operations to be displayed on client or server side. AS/400 EASYCOM server processes elementary requests applied to tables or procedures.

It receives a process request from the network and returns a response.

The requests and responses are recorded in a trace file, it can be used as basis to analyze data flow between client and server.

Lines starting with << indicates client request.

Lines starting with >> indicates AS/400's answer.

```
<<EACopen (EASYCOM/SP_CUST,4194309,-1) B Requête.
9 Fields, 0 key fields
EAC_NO_CVT - mode=
>>Ret=1; Err=0; Msg=; Int=0 B Réponse.
```

In AS/400 trace, if time option was selected, all requests and responses are time in hh:mm:ss.ms format.



```
<<15:48:45.566: EACread(1,p(2275),91,34144281,(null),0,p(100))
```

In response, data "Clk=x" indicates AS/400 CPU time spend to process the request.

```
>>15:48:45.574: Clk=8, Len=619; Ret=5; Err=0; Msg=; Int=
```

**Trace file header (common to all sessions)**

This trace file part is always the same for all EASYCOM sessions.

**Time is : 03/27/2000 - 17:22:10**

**Easycom Server Version is : 4.5712, Link is TCP/IP**

**Client licence is : D\$WINDEV10 , Easycom Library is : EASYCOM**

**JobName=ALBATROS, User=QPGMR , QCCSID=297 Heart Beat freq :10**

**Easycom Log File TRACE/SR, level 1**

```
>>Ret=1; Err=0; Msg=; Int=0
```

```
<<RTV_AS_VER(p(4))
```

```
>>Ret=4; Err=0; Msg=; Int=0
```

```
<<WriteTableEBCDI(49 42 4D 43 43 53 49 44 20 30 20 31 32 35 32 00 00 00 00 ... (256))
```

**Build Table from CCSID:0 to 1252**

```
open IBMCCSID01252, IBMCCSID00000000100
```

```
<<WriteTableASCII(49 42 4D 43 43 53 49 44 20 31 32 35 32 20 30 00 00 00 00 ... (256))
```

**Build Table from CCSID:1252 to 0**

```
open IBMCCSID00000, IBMCCSID012520000100
```

```
<<ReadTableASCII(p(256))
```

```
>>Ret=0; Err=0; Msg=; Int=
```

```
<<ReadTableEBCDIC(p(256))
```

```
>>Ret=0; Err=0; Msg=; Int=0
```

```
<<EACSqlDeclare(2A 45 41 43 20 43 56 54 20 4E 4F 00 ,12)
```

**Statement:\*EAC CVT NO**

```
>>Ret=1; Err=0; Msg=; Int=0
```

```
<<EACSqlBegin()
```

```
>>Ret=0; Err=0; Msg=; Int=0
```

**Physical file opening trace**

```
<<EACopen(EASYCOM/SP_CUST,4194309,-1)
9 Fields, 0 key fields
EAC_NO_CVT - mode=rr+
>>Ret=1; Err=0; Msg=; Int=
<<EACgetdesc(1,p(65000),65000,939786240,(null))
>>Ret=9; Err=0; Msg=; Int=
```

**Logical file opening trace**

```
<<EACopen(EASYCOM/SP_CUST_UN,4194309,-1)
LF with 1 Data Members, 1 Record Formats
9 Fields, 1 key fields
EAC_NO_CVT - mode=rr+
>>Ret=2; Err=0; Msg=; Int=
<<EACgetdesc(2,p(65000),65000,939786240,(null))
>>Ret=9; Err=0; Msg=; Int=
```

**File records reading trace**

```
<<EACread(2,p(816),102,34144264,(null),0,p(32))
```

```
VERB=_EAC_NEXT LOCK=OFF RECS=8 FILE=EASYCOM/SP_CUST_UN
```

```
RRN=2 RRN=4 RRN=5 RRN=6 RRN=7 RRN=8 RRN=9 RRN=10
```

```
>>Ret=8; Err=0; Msg=; Int=0
```

Read operation type is indicated by « VERB=

Where xxxx may be:

FIRST, NEXT, PREV, LAST, KEY\_EQ, KEY\_GE, KET\_GT, ...

"**LOCK=** indicates if operation is carried out with or without record locks.

"**RECS=** indicates maximum records number requested for the response.

This number is directly linked to "Records= data" in the "Easycom.ini" file "Buffers section" on client PC.

"**RRN=** indicates the records read number.

In response, "Ret=n" indicates the records number actually returned, to the read request.

If the read operation fails because of an input/output error, the message is stored in the trace, and the records actually read are returned.

```
<<EACread(2,p(3570),102,34144291,(null),0,p(140))
```

```
VERB=_EAC_NEXT LOCK=OFF RECS=35 FILE=EASYCOM/SP_CUST_UN
```

```
RRN=11 RRN=12 RRN=13 RRN=14 RRN=15 RRN=16 RRN=17 RRN=18 RRN=19 RRN=20 RRN=54 +...
```

```
... RRN=55 RRN=56
```

```
**SIGIO** Msg:CPF5001
```



>>Ret=13; Err=5001; Msg=CPF5001; Int=0

In this example, 35 records are requested, but only 13 are available until file end.

To obtain the detailed error message, use [DSPMSGD](#) command.

#### SQL request opening trace

```
<<EACopen(SELECT * from SP_CUST where LASTNAME>'M',4194309,-1)
Statement : SELECT * from SP_CUST where LASTNAME>'M'
Cursor 0
>>Ret=2; Err=0; Msg=; Int=
<<EACgetdesc(2,p(65000),65000,939786240,(null))
>>Ret=9; Err=0; Msg=; Int=
```

### Common error messages

#### **Connection error 10060 (274C Hexa) : Connection Time out**

The called TCP/IP address does not exist on the network.

AS/400's TCP/IP address or name must be checked.

If an AS/400 machine name is used, check that it is properly referenced on DNS servers.

#### **Connection error 10061 (274D Hexa) : Connection Refused**

IP address or name of AS/400 must be checked.

EASYCOM system proper launch on AS/400 must be checked.

Subsystem must be launched with command:

**STRSBS EASYCOM/EASYCOM**

If the subsystem was started, check if EASYCOMD job runs.

If not, it must be started with the command:

**STREACD EASYCOM**

Or subsystem must be stopped restarted.

Connection must be tested using EASYCOM configuration or administration tool.

If the EASYCOMD job can't be started, messages that EASYCOM generate have to be checked using the commands:

**DSPMSG EASYCOM/EACMSGQ** or **DSPPFM EASYCOM/LOGFILE**

Default EASYCOM port number is 6077.

If this number is already used, use [CFGEACSBS](#) to configure another port, and change [client configuration](#) to select the port number.

#### **Connection error 11001 (2AF9 Hexa): Host not found**

On TCP/IP network AS/400 can be identified with its name at DNS level or host file. This error occurs when this name is used as IP address in connection parameters and is not found and associated with the right IP address.

AS/400 name, host file, DNS servers, must be checked or an IP address in xxx.xxx.xxx.xxx format must be used.

#### **Where is the Hosts file located?**



Usually in C:\WINDOWS\system32\drivers\etc repertory.

This file contains IP addresses relation to host names. Each entry must set on his proper line. The IP address must be placed in the first column, followed by the related host name. The IP address and the host name must be separated with one space at least.

Moreover, comments can be inserted on their proper lines or after computer name. They are indicated with '#' symbol.

Example:

```
194.206.10.1 main.as # main AS/400 server
194.206.10.2 test.as # AS/400 test server
194.206.10.100 serveur.info1
194.206.10.101 poste_x
....
```

**DNS Server**

Allows checking a DNS server address in connection network Internet (TCP/IP) Protocol properties.

**Easycom Dll not found**

- In the WinDev Program directory for a development configuration.
- In the application directory for a deployment,
- Or in any directory present in the PATH, for deployment.

**You have no free Connection on EASYCOM Serveur**

This error message is raised when you reach the maximum number of simultaneous sessions authorized by your Easycom [license](#).

Each session is a job in EASYCOM sub system. With WRKACTJOB command, you can check the number of jobs running, and who is using Easycom.

Warning ! One single application can open multiple connections.

**Error codes**

**Easycom For WinDev errors summary**

WinDev functions `HErrInfo()` and `HErrInfo(hErrNativeMessage)` return a detailed error message where you can see an "Error Category".

	<b>Category</b>	<b>Description</b>
SQL	1	SQL native error.
SIGNAL	2	Signal Error. example CPF9810 : Library not found.
MULTIREC	3	Error while writing cached records.
APPC	4	APPC error (No longer used)
INTERNAL	5	Easycom internal error.
TCPIP	6	TCP/IP error (connection, ...)
UNKNOWN	7	General error.
EACWD	8	Specific Easycom For WinDev Error.

**Category 1 - SQL Error**

See IBM documentation about DB2/UDB database for more details.



### Category 2 - Signal (CPF...)

Use function [ASErrorHelp](#) or [ASErrorData](#) to get more information about the signal message.

With AS/400 command WRKACTJOB, you can manage the Easycom job, and see the job log, where you can find more details.

Easycom Trace file can also help you in getting more information.

### Category 3 - Error writing cached records

This error can raise only if you set CACHEDINSERT to true, with function [ASProperty](#)

### Category 5 - Internal

Error code	Description
257	You may not open another file then specified for the demonstration
258	License key is not valid. Please do DSPMSG QSYSOPR to have full details if needed (which kind of license is required)
260	License key expired. Please do DSPMSG QSYSOPR to have full details if needed (which kind of license is required)
261	No free connection. The number of allowed simultaneous connection was reached, and this new connection is not allowed.
263	License key not found. There is no license for the product currently used. Please do DSPMSG QSYSOPR to have full details if needed (which kind of license is required)
275	There is no license for this option. An option of the product is required but not found. DSPMSG QSYSOPR can contain more information if needed.
1	Parameter error. There was an invalid request sent to Easycom. This can be caused by unexpected usage of Easycom, a bug in the application or a bug in the Easycom upper stack (specific part to a product, like Delphi, WinDev, PHP, ...)
2	Memory allocation error. This is usually caused by incorrect size during memory allocation on the server. The possible reasons are: unexpected usage of Easycom, a bug in the application or a bug in the Easycom upper stack (specific part to a product, like Delphi, WinDev, PHP, ...)
3	File not opened. Attempt on a non-opened file. This is probably an application or easycom bug.
522	Cannot convert a NULL parameter. Problem during iSeries <-> client conversion on a NULL value.
527	Problem during ALCOBJ action. ALCOBJ was requested but failed
528	Failed to create an object. An object creation attempt failed.
529	Timeout on pgm call. A timeout was defined for a pgm call (using CFGEAC or by client application), and this timeout was reached. The program call was cancelled, with possible non closed context. Restarting the connection is recommended.
530	Procedure not found. A procedure call was requested, but the procedure was not found in the service program

### Category 6 -TCP/IP Errors

Negative error codes mean an Easycom protocol error during TCP/IP connection, and positive ones mean native TCP/IP errors.

Native codes (positive) change depending on the client platform (Windows, Linux, AIX, iSeries, ...).

All errors come with a local error text, and most of the time with a specific error text coming from the iSeries.

Here are negative codes :

Error code	Description
-1	Error while submitting the job. SBMJOB made by the EASYCOMD job failed. Additional error text coming from iSeries will is provided with this error. The EASYCOMD job history should contain all information on that failure.

-2	Security not valid. This error can occur if wrong user, password, password disabled, etc. The detailed reason is specified as text.
-4	submitted job did not answer, or failed to initialize data queues The most common reason for this is that the job failed to run. It was submitted but ended before beginning to communicate with the client. This is usually caused by wrong user's jobd. The full reason can be found in the QZEJOBLOG OUTQ of the system. To see it, do the following commands: <ul style="list-style-type: none"> <li>○ WRKOUTQ OUTQ(QEZJOBLOG)</li> <li>○ Type F18 (to go at the end), and then F11.</li> <li>○ There should be a line with the station name, with the corresponding user, date and time.</li> <li>○ Type 5 on it to see the errors.</li> </ul>
-5	Password is expired. If the client program 'catches' this error, it can perform a custom password change dialog box, and send the password with the new connection request. The password send by the application will have the following form in this case: oldpassword@newpassword
-6	Internal reject 1. Unexpected error, caused by a bug in EASYCOMD. Please contact help support. Restarting EASYCOM subsystem is recommended.
-7	Failed to init the library list. Errors occurred when installing the libraries that are defined in the user's jobd. You can consult the QEZJOBLOG outq for more information (see error -4)
-8	SSO error. SSO profile is expired (re-signoon required), or not supported by EASYCOMD.
-9	The server cannot accept Kerberos tickets. EIM SSO is not configured, or the EASYCOMD LDAP connection failed. Do DSPMSG EASYCOM/EACMSGQ to see if there are Kerberos-related messages. Check that you see 'Eim='
-10	Timeout on read. Communication error: the read request timed out. The connection was probably broken.
-11	Logon cancelled. This error occurs when message boxes are enabled and when the user clicks on 'cancel'.
-12	Connection broken. The connection was broken by peer.
-13	Kerberos negotiation protocol failure. There was an unexpected Kerberos error when connecting. Check EIM configuration, and check if the same user is working using IBM Client Access in EIM mode.
-14	Kerberos error on client. The client failed to generate a ticket to send to the server. Additional text should explain the reason.
-15	Kerberos error on server. The server did not recognize the ticket or failed to grant it.
-16	OS/400 incompatible version. The OS/400 version is not compatible with the current request.
-17	Unexpected error while submitting (state unknown). Unexpected error probably caused by a bug. Please contact help support.
-18	Kerberos authentication out of hours. See CFGEACEIM to setup the valid hours for Kerberos authentication.
-19	Out of hours by exit program. The EACLOG001 exit program returned that the login is not valid at this time.
-20	Denied by exit program. The EACLOG001 exit program return that the login is denied
-21	Not processed by exit program. The EACLOG001 exit program returned that the login is invalid.
-22	Kerberos authentication is not supported by the server. See that CFGEACEIM enabled *EIM mode and that EASYCOMD started properly (DSPMSG EASYCOM/EACMSGQ).
-23	Kerberos authentication is mandatory. The Easycom server was configured to accept only Kerberos authentication, but a regular login was attempted.
-24	Failed to use the target library. The library specified by the target program property (Program= in easycom.ini, in section [general]) was not usable, because nonexistent or other reason.
-25	Awake on private job failed. The application attempted awaking a job that was registered by the setting, but it fails. A new connection is required. Note: this error currently can appear only with Easycom For PHP.
-26	SSL required on this server. The SSL negotiation was not setup or failed, but is required on the server. This can be marked as required using the <a href="#">EACTCPP01</a> exit program or using CFGEAC command.
-27	SSL server error. The SSL negotiation failed because the server detected an error. There are probably some information in the EACMSGQ message queue (type DSPMSG EASYCOM/EACMSGQ on a terminal)
-28	SSL negotiation was made, but a failure is detected while passing the connection to the Easycom job.
-29	SSL client error. The SSL negotiation failed because of an error on the client. More additional information is provided in the error message text.



-30	SSL sequence error. The SSL negotiation sequence was detected as invalid
-31	SSL protocol error. An SSL error is detected during SSL handshake.
-32	SSL error: SSL not supported on the platform
-33	EIM was mandatory for login
-34	SSL authentication is mandatory
-35	SSL authentication error (bad certificate, expired, ...)
-36	EIM error
-37	No valid authentication provided. This means that all kind of accepted authentication methods failed.

Here are most common TCP/IP error codes:

Windows error code	AS/400 error code	Description
<a href="#">10061</a>	ECONNREFUSED 3425	Connection refused. No connection could be made because the target machine actively refused it. This usually results from trying to connect to a service that is inactive on the foreign host - i.e. one with no server application running.
<a href="#">10060</a>	ETIMEDOUT 3447	Connection timed out. A connection attempt failed because the connected party did not properly respond after a period, or established connection failed because connected host has failed to respond.
<a href="#">11001</a>	HOST_NOT_FOUND 5 (host error category)	Host not found. No such host is known. The name is not an official hostname or alias, or it cannot be found in the database(s) being queried. This error may also be returned for protocol and service queries and means the specified name could not be found in the relevant database.
10053	ECONNABORTED 3424	Connection aborted. An established connection was aborted by the software in your host machine, possibly due to a data transmission timeout or protocol error.
10064	EHOSTDOWN 3428	Host is down. A socket operation failed because the destination host was down. A socket operation encountered a dead host. Networking activity on the local host has not been initiated. These conditions are more likely to be indicated by the error WSAETIMEDOUT.
10050	ENETDOWN 3433	Network is down. A socket operation encountered a dead network. This could indicate a serious failure of the network system (i.e. the protocol stack that the WinSock DLL runs over), the network interface, or the local network itself.

The localized error text is available at runtime and can be shown by the application or Easycom dialog boxes.

NB: most connection problem is caused by routers or firewalls installed on the client stations, or on the network.

Easycom is using one TCP/IP connection by default on tcp port 6077.

### Category 8 - Easycom For WinDev Errors

## Migrating from previous WinDev versions.

### Migrating Windev 5.5 applications

**Warning:** To migrate a WinDev 5.5 project, you first need to migrate it to WinDev 7.5.

Alias files used with WinDev 5.5 (\*.AS) are still supported by Easycom For WinDev 28, but they are usually replaced by extended info.

#### Non supported functions



Obsolete function	Replaced by
<b>ASError</b>	<a href="#">HErrorInfo</a> ,
	Extended by <a href="#">ASErrorHelp</a> et <a href="#">ASErrorData</a>
<b>ASConnect</b>	<a href="#">HOpenConnection</a>
<b>ASDisconnect</b>	<a href="#">HCloseConnection</a>
<b>ASSource, ASReadAlias ASWriteAlias</b>	File extended info.
<b>ASSQLImmed</b>	<a href="#">HExecuteQuery</a>

#### Function Return values

All the functions return booleans (True / False), except [ASErrorHelp](#) and [ASErrorData](#) .  
**Alias Files.**

File extended info replace Alias Files.

You can still use Alias files by setting ONLYALIAS option in extended info, or with [ASProperty](#) function.

#### Migrating from WinDev 7 and up

WinDev 28 migrates automatically from older versions.

Easycom has a unique DLL for each WinDev version (eac750as.dll, eac800as.dll, ... eac2700as.dll, eac2800as.dll). So, different versions can be installed on the same system.

Easycom server must be updated to be compliant with latest WinDev versions.

Newest Easycom server versions are compatible with ancient WinDev versions.

### Migrating from WinDev 7, 8, 9, 10, 11

WinDev 28 migrates automatically from older versions.

Easycom has a unique DLL for each WinDev version (eac750as.dll, eac800as.dll, ... eac2700as.dll, eac2800as.dll). So, different versions can be installed on the same system.

Easycom server must be updated to be compliant with latest WinDev versions.

Newest Easycom server versions are compatible with ancient WinDev versions.

## Programs and Data Queues

### Introduction

AS/400 native programs and procedures can be called from WinDev programs using functions [ASRunRPC](#) , or [ASPgmCall](#)

If you use **ASRunRPC** to call AS/400 native programs or procedure, you need to describe program parameters with [RPC/DTAQ Configuration Tool.](#), and import the description into the analysis, like a file, using \***PGM** as library name.

Data Queues are handled like database files from WinDev programs. A file exists in the analysis for each data queue to access.

Data Queues also must be described with [RPC/DTAQ Configuration Tool.](#), and then imported into the analysis, by using \***DTAQ** as library name.

Descriptions for Programs and Data Queues are stored on your AS/400 system in files **YPROCHDR** and **YPROCPARMS**, located in the Easycom Job LIBL. (Default is EASYCOM library).

[RPC/DTAQ Configuration Tool.](#) allows you to export and import descriptions to help porting description from one AS/400 to another one.



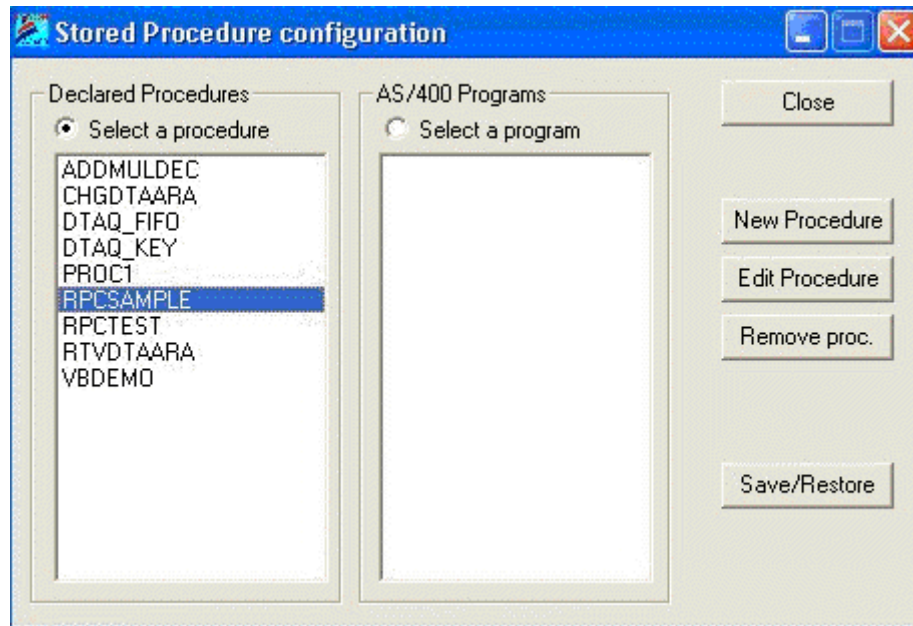
### AS/400 Native Programs Description

EASYCOM enables AS/400 native programs calling, CL or RPG programs or stored procedures.

To perform this, EASYCOM needs these programs description stored on AS/400 in YPROCHDR and YPROCPARMS files in EASYCOM library.

Programs description and data queues are built by DTAQ-RPC constructor. The basic principle is to specify all parameters, types and uses (input, output, input/output) required to call the program.

The first screen displays the existing procedures (stored on AS/400) and enables to create, modify or deleted them. The descriptions can be saved in a PC text file in view of a later transfer to another AS/400.



A new name is assigned to the procedure. It does not need to match with the associated program name.

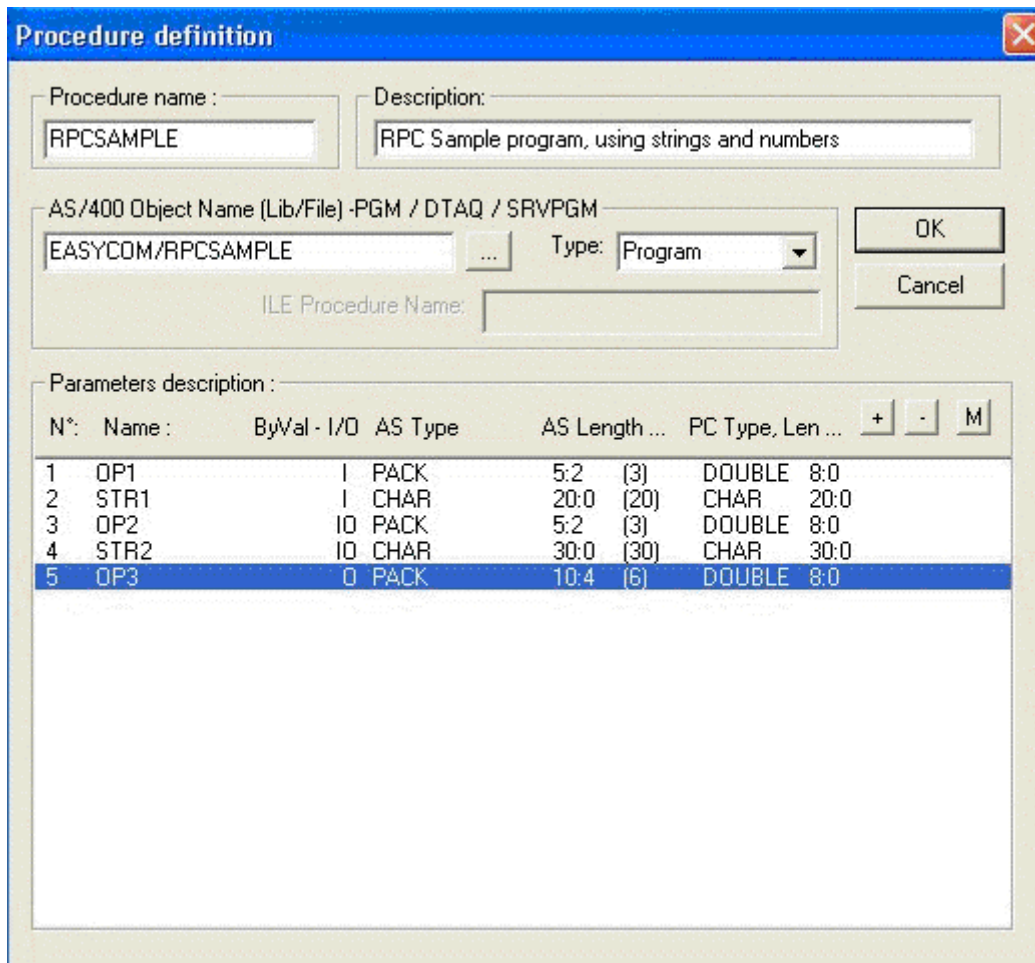
A native AS/400 program (CL, RPG, COBOL, C etc.) is associated to the procedure.

The library may be omitted or replaced by \*LIBL.

The description is a free text, which will be seen when client workstations browse through the procedures.

Each program calling type and size parameters are described.

Each parameter may be considered as a database table field.



**Procedure definition**

Procedure name :  Description:

AS/400 Object Name (Lib/File) -PGM / DTAQ / SRVPGM  
 ... Type:

OK  
Cancel

ILE Procedure Name:

Parameters description :

N°	Name	ByVal	I/O	AS Type	AS Length ...	PC Type, Len ...	+	-	M
1	OP1	I		PACK	5:2 (3)	DOUBLE 8:0			
2	STR1	I		CHAR	20:0 (20)	CHAR 20:0			
3	OP2	IO		PACK	5:2 (3)	DOUBLE 8:0			
4	STR2	IO		CHAR	30:0 (30)	CHAR 30:0			
5	OP3	O		PACK	10:4 (6)	DOUBLE 8:0			

Each parameter can be considered as a field in a database table.

It therefore has a name, by which it can be referred to by the application.

Parameters designed to provide values for the called program are considered as input parameters (IN).

Parameters designed to receive a value on returning from the call are considered as output parameters (OUT).

Parameters that are modified by the program are both input and output (IN/OUT) parameters.

By default, all the parameters in an AS/400 program are both input and output. The logic of the program can change this property.

If a calling parameter of the program is a structure (DS: Data Structure), each field of the DS has to be described individually.

For the first field only, the box to be ticked is: This field is a parameter or the 1st Field.

The type of parameter expected by the AS/400 program must be specified exactly:

- CHAR : Character data type.
- BIN2 : 16-bit numeric data type.
- BIN4 : 32-bit numeric data type.
- PACK : Condensed numeric data type (DECIMAL).  
This is the format in which CL handles numerical data (CL \*DEC type).
- ZONED : Extended numeric data type (NUMERIC).
- DATE : AS/400 date in the yyyy-mm-dd format.

- TIME : Time in hh:mm:ss format.
- FLOAT : Numeric value in single-precision floating point.
- DOUBLE : Numeric value in double-precision floating point.
- TIMESTP : Elapsed time field.
- GRAPHIC : Character type data, not to be converted.
- EXTERNAL DS : A structure described by an external data structure, i.e. a physical file.

### Calling ILE Procedures

The ILE procedure must be described, as a program (Type \*PGM).

The AS/400 object in "AS/400 Object Name" field must be \*SRVPGM type.  
It must be "Service Program" type.

The first described parameter is the procedure returned value.  
Only returned values type "Integer 32 bits" are accepted.

Then, the parameters are described, as for a OPM program.

16 parameters maximum are accepted for the procedure, plus returned value.



**Procedure definition**

Procedure name : PROC1      Description:

AS/400 Object Name (Lib/File) -PGM / DTAQ / SRVPGM  
 AURA/SRVPGM01      Type: Service Program

ILE Procedure Name: Proc1

Parameters description :

N°	Name	ByVal - I/O	AS Type	AS Length ...	PC Type, Len ...	+	-	M
Ret	RET		IO BIN4	0:0 (4)	INT 32 4:0			
1	P1		IO BIN4	0:0 (4)	INT 32 4:0			
2	P2		IO BIN4	0:0 (4)	INT 32 4:0			
3	P3		IO BIN4	0:0 (4)	INT 32 4:0			
4	P4		IO BIN4	0:0 (4)	INT 32 4:0			
5	P5		IO BIN4	0:0 (4)	INT 32 4:0			
6	P6		IO BIN4	0:0 (4)	INT 32 4:0			
7	P7		IO BIN4	0:0 (4)	INT 32 4:0			
8	P8		IO BIN4	0:0 (4)	INT 32 4:0			
9	P9		IO BIN4	0:0 (4)	INT 32 4:0			
10	P10		IO BIN4	0:0 (4)	INT 32 4:0			
11	P11		IO BIN4	0:0 (4)	INT 32 4:0			
12	P12		IO BIN4	0:0 (4)	INT 32 4:0			
13	P13		IO BIN4	0:0 (4)	INT 32 4:0			
14	P14		IO BIN4	0:0 (4)	INT 32 4:0			
15	P15		IO CHAR	200:0 (200)	CHAR 200:0			

**Parameter field edition**

Field Name: RET      Field usage :  
 In     In/Out  
 Out

This field is a parameter or the 1st field of a Data Structure      Function Result

AS/400 Format      PC Format

Type : BIN4      Type : INT 32

Digits : 0      Length : 4

Decimals : 0      By Value :       Decimals : 0

External DS  
 Library:      File: ...

OK      Cancel

For each parameter, option "By Value" has been added.

This option is valid only for parameters type "32 Bits integer".

When checked, this option indicates that the procedure receives this parameter in value, and not in address.

## Moving procedures and DTAQ descriptions between AS/400

When a developer works on an AS/400, he creates procedures and data queues which will subsequently have to be used on another AS/400, these procedures descriptions and data queues must be transferred to the other AS/400.

Descriptions are stored in three files: YPROCHDR, YPROCPARMS and YPROCPGM. They are stored in AS/400 EASYCOM library (default). They can also be placed in another library. In this case they will be searched in connected profile LIBLE.

If two AS/400s are connected, the above three files can obviously be transferred directly from one to the other.

Otherwise, the DTAQ-RPC manufacturer offers a facility to import/export descriptions from or to text files, that means that the necessary descriptions can be saved on the developer's workstation and then restored on the client's.

## Using DATA QUEUES

Accessing a Data queue with Easycom For WinDev, is like accessing a database file.

- Use `HAdd()`. Function to add a new entry to a data queue.
- Use `hRead HReadNext()`. Or `HReadSeek()`. To read a data queue entry.

Data queues can be created with different sequence:

- **\*FIFO**, for First In – First Out,
- **\*LIFO**, for Last In – First Out,
- **\*KEYED**, where each entry is indexed by a key value.

Data queue entries must be described using [RPC - DATAQ configuration tool](#) , then imported into the analysis, like a database table.

### *Adding an entry to the data queue.*

Whatever is the sequence of the data queue, an entry is added with function `HAdd`

Onlu data field needs to be updated.

Example :

```
// Add entry to DataQueue
DTAQ_FIFO.DATA = data
HAdd(DTAQ_FIFO)
```

### *Reading from a FIFO or LIFO data queue*

Read the data queue entry with function `HReadSeek`.

```
HReadSeek(DTAQ_FIFO, TIMEOUT, "000000")
```

When TIMEOUT key field is 0, the system returns immediately the value, or set `HOut` to `True` is the data queue is empty.

Set TIMEOUT key value to several seconds to wait before to exit with `HOut = True`.

Example:

```
WaitStrg is string

WaitStrg = timeout
HReadSeek(DTAQ_FIFO, TIMEOUT, WaitStrg)
```



```

IF HOut = True THEN
    Info("DataQueue is empty !")
ELSE
    // Value is here
    data_lue=DTAQ_FIFO.DATA
END
    
```

#### Reading from a KEYED data queue

The « Virtual key » of a keyed data queue is composed as follow:

- TIMEOUT field: Number of seconds to wait before returning with `HOut = True`
- FILER : must be blank.
- ORDER: a 2-character field containing the comparison operator (EQ, NE, GT, GE, LT, LE).
- Key value to search for.

ORDER field has the following meaning :

EQ	=	Equal to
NE	<>	Non equal to
GT	>=	Greater than
GE	>	Greater or Equal
LT	<	Less than
LE	<=	Less or Equal

#### Example

```

// Writing to keyed data queue.
DTAQ_KEY.DATA=data
DTAQ_KEY.KEY= key
HAdd(DTAQ_KEY)
    
```

```

KeyVal is string
// Reading from keyed data queue
KeyVal=HBuildKeyValue(DTAQ_KEY,DTAQ_KEY,timeout,"","EQ",key2)
HReadSeekFirst(DTAQ_KEY,DTAQ_KEY,KeyVal)
IF HOut THEN
    Info("DataQueue is empty !")
ELSE
    EntryValue=DTAQ_KEY.DATA
END
    
```

#### Using DATA AREAS

Use [AsExec](#) function to change a data area value.

Use [ASRtvCall](#) and [ASRtvResult](#) functions to retrieve a Data Area value.

#### Example

```

eacret is int
CmdLine is string
RetVal is string

// Connect to AS/400
    
```





```
IF NOT
HOpenConnection("MyConnection",user,pwd,ipadress,hNativeAccessAS400,hOReadWrite,") THEN
  Info(HErrorInfo())
END

// Write to DATAARA
CmdLine="chgdtara dtaara(easycom/dtaara) value("Hello World")
IF NOT AExec(CmdLine) THEN
  Info(ErrorInfo())
END

// Retrieve DTAARA value
CmdLine = "rtvdtara dtaara(easycom/dtaara *ALL) rtnvar(&var1)"
eacret = ASRtvCall(CmdLine)
IF NOT eacret THEN
  Info(ErrorInfo())
END

// Lecture du Result
RetVal = ASRtvResult("var1")
IF NOT RetVal = "0" THEN
  Info("New DTAARA value is : " + RetVal)
ELSE
  Info("L'appel a échoué")
END
HCloseConnection("MyConnection")
```

## Application Deployment

### Deploying WinDev applications

Installation of the client part of Easycom will be done by the WinDev deployment process.

When you create the deployment procedure for your application with WinDev IDE, AS/400 native access DLL (**eac2800as.dll**) is always selected in the list of mandatory DLLs, as soon as an AS/400 file is used in the project, or an Easycom function is called.

We suggest using this standard WinDev mechanism to install Easycom client for your application deployments.

So, you don't have to run Easycom installation procedure on users' workstations.

Alternatively, you can also copy DLL file **eac2800as.dll** into Windows directory. If more than one application is using Easycom, all the applications will use the same DLL, and future updates of Easycom Client DLL will be easier.

Only file **eac2800as.dll** is mandatory.

#### **Easycom.ini : Local configuration file**

Easycom.ini file contains several settings (Cache size, Default system name, ...).

You can copy it from your development environment to the user workstations, or you can include it in your application deployment procedure.

### Deploying on another AS/400 Server

**Easycom server must be installed on the deployment system.**

Run the same [installation procedure](#) you used to install your development system.

Deployment license must be registered. You need an activation key.

### Programs and Data Queues descriptions

If your application is deployed on another AS/400, and if it uses program calls or Data Queues I/O, you need to port the descriptions from your development system to the deployment one.



Descriptions are stored in files YPROCHDR and YPROCPARMS in Easycom Library (by default). Library where YPROCHDR and YPROCPARMS might have been changed with the "[RPC-DTAQ configuration](#)" utility.

You can copy those two files from the development system to the deployment.

Or you can use the "[RPC-DTAQ configuration](#)" utility to save descriptions and restore them on the target system.

## WebDev

### WebDev Deployment

On a WebDev server, Easycom dialog boxes and message boxes are disabled. So, you must manage connections and all errors from your program.

### Deployment

On a development computer, where Webdev28 and the AS400 native access are installed, you should find the EASYCOM DLLs (eac2800as \*.\*) here (by default):

C:\WebDev28\Programs\Framework\Win32x86 for 32-bit DLLs.

C:\WebDev28\Programs\Framework\Win64x86 for 64-bit DLLs.

Copy the DLLs, depending on your case, 32 and/or 64 bits, on your Web server: in C:\Webdev 28, next to the other Windev/Webdev DLLs.

### Prestarts jobs

To speedup connection time, [Prestarts jobs](#) are recommended.

### Record Locking

When a record is modified by your application, while it was already modified by another user, WebDev cannot open a dialog box to ask the user to confirm the update.

So, you need to manage record locking errors from your application.

See [WebDev Special Record Locking](#).

Option DRVOPTIMISTIC connection extended info can also be used.

## Data types conversion and formats

### ASCII and EBCDIC Sort sequences

By default, File sort sequence is based on EBCDIC character sets.

- EBCDIC sequence is: Lowercases – Uppercases - Digits.
- ASCII sequence is Digits – Uppercases – Lowercases.

To get an "ASCII Like" sequence with AS/400 files, you need to specify the right Sort Sequence when creating access paths on AS/400.

### Example

```
CRTL F FILE (EASYCOM/SP_CUST_CU) SRTSEQ(*LANGIDSHR)
```

**SRTSEQ(\*LANGIDSHR)** is a sort of sequence in which, digits are at the beginning of the sequence, lowercases and uppercases characters are mixed.



## Fields and Formats

### Dates

If NULL is allowed for a date field, a "blank" value will be considered as a null.

Otherwise, a blank value is converted to "01.01.0001" when writing, and "01.01.0001" is converted to blank when reading.

See DATETYPE and TIMETYPE connection extended info for more details.

On SQL queries, when *HQueryWithoutCorrection* is used, **\*ISO** format must be used: YYYY-MM-DD (2005-12-25) ; HH:MM:SS (17:59:59) .

### Memos and BLOB

Memo management is different between AS / 400 files with imported blobs in the analysis and Hyper File files with memos exported to the AS/400.

#### Imported files

A BLOB field on the imported AS/400 file is considered as a memo.

Caution: A file with BLOB fields must be journalized.

#### Exporting Files

When files are exported to AS/400, an additional file is created for each file having memo fields.

The name of this additional file is made of the first 8 characters of the main file name, prefixed by 2 underscore characters ( \_\_ ).

In the main file, memo field is replaced by a numeric field containing the memo ID.

Memo files are automatically managed by Easycom when the main file is accessed in native mode.

When the file is managed with SQL, you only get the memo ID. You need to manage the memo file to get the memo value.

When your WinDev program doesn't need memo values, it is recommended to disable memo management with [HSetMemo](#) function.

### UNICODE

UNICODE is supported by Easycom since version 4.58.56 .

When using SQL queries, if a "Unicod literal value" is given in a statement for a UNICOD field, then, the CCSID for the Easycom job on AS/400 must support Unicod (1147, 13488, ...).

You can set default CCSID for all Easycom jobs with **CFGEAC** command on AS/400.

Or you can dynamically change CCSID job with:

```
ASExec ("CHGJOB CCSID ( . . . ) ")
```

### CCSID

AS/400 system converts database data to and from job CCSID (Coded Character Set Identifier), except when CCSID is 65535.

Then, Easycom converts data to and from PC Code Page.

Easycom job default CCSID is equal to the User CCSID, as defined in the user profile.

You can change this default with CFGEAC command, or by call "CHGJOB" command from your WinDev program with ASExec function.



## Codepage

Easycom identifies automatically how to convert data between the job CCSID, and the Windows code page.

You can change the default by giving a conversion table.

Install "Easycom International Pack" to get conversion tables (\*.CPG files).

Each file in the pack provides conversion rules between an EBCDIC CCSID and an ASCII character set.

Example: E037ANSI.CPG = Conversion between EBCDIC 37 and ANSI ASCII.

You can change the conversion table for all Easycom connections, using "**Easycom.ini**" file.

[GENERAL]

ConvTable=c:\Program Files\Easycom\WinDev10\E037ANSI.cpg

To change conversion table for one connection only, change the connection extended info in the analysis.

<EASYCOM>

CODEPAGEFILE=c:\Program Files\Easycom\WinDev10\E037ANSI.cpg

</EASYCOM>

## Data types equivalences

An AS/400 data type is by default translated into a HF when importing a file description, or when executing an SQL query or using HDeclareExternal.

Default data type mapping can be manually changed by changing a item data type after DDS import, or by setting the field extended info before to export description to AS/400.

DDS type	SQL type	Condition	HF type
A (character)	CHAR		Text
A OPTION(VARYING) (variable length)	VARCHAR		Text
G + CCSID 13488 (Unicode)	GRAPHIC CCSID(13488)		Text Unicode (version 12 and up only)
G + CCSID 13488 + OPTION(VARYING) (Unicode variable length)	VARGRAPHIC CCSID(13488)		Text Unicode (version 12 and up only)
P (Packed decimal) or Z (Zoned)	DECIMAL or NUMERIC	Integer, <= 4 digits	2 bytes signed integer
		Integer, <=9 digits	4 bytes signed integer
		Integer, <=19 digits	8 bytes signed integer
		<= 15 digits	8 bytes Real
		<= 17 integer digits, and less than 6 decimal digits.	Currency
		<= 38 digits or SQL with SQLFULLPRECISION property set.	Numeric (version 12 and up only)
		Others	Text
L	DATE		Date
T	TIME		Time (HHMMSS)
Z	TIMESTAMP		Date and Time

B4 (Short Integer)	SMALLINT		<b>2 bytes signed integer</b>
B9 (Long Integer)	INT		<b>4 bytes signed integer</b>
B19 (Integer 64 bits)	BIGINT		<b>8 bytes signed integer</b>
F	FLOAT		<b>Real 8 bytes</b>
F double precision	DOUBLE		<b>Real 8 bytes</b>
H	BINARY		<b>Binary string</b>
H	VARBINARY		<b>Binary string</b>
N/A	CLOB		<b>Memo text</b>
N/A	BLOB		<b>Other Binary Memo</b>
N/A	DBCLOB + CCSID 13488		<b>Memo Unicode</b>

Note :

Easycom maintains initial HyperFile data types when a file is exported to AS/400 and imported again. Original data type is stored in the field description.

When a file is imported, Field Extended info can contain NATIVETYPE property, to memorize the original AS/400 data type.

## EASYCOM Server

### Easycom server

Easycom server is a Software engine running on System I – AS/400.

It is compliant with all the Easycom connectors and drivers for many development tools:

WinDev & WebDev	PHP
Delphi	.NET
Java	OLE DB

This is the core of Easycom technology.

Basically, when installed, configured, and running, Easycom is a TCP/IP service running in a subsystem. It is listening on a TCP Port, waiting for client connections.

Easycom Client modules are running on Windows, PASE, AIX, Linux, and many other platforms.

Easycom technology is owned by Aura Equipements company, France.

## Installing and configuring EASYCOM

### Installation

See [installation part](#).

### Configuration

See [configuration part](#).

### EASYCOM behavior

#### *EASYCOM Server configuration files*

AS/400 EASYCOM library contains all EASYCOM objects.



When the server is launched the first time a single additional object is created: EASYCOM object, \*FILE type in QGPL library.

An EASYCOM library entry is automatically added to the online libraries list (ADDLIB) for each job started on client workstations. Therefore, no explicit addition to the 'users' JOBD is required.

The following objects may be modified in the EASYCOM library

AURA	*FILE	Modified when the user licence is registered.
EACSESSION	*FILE	Modified when the user licence is registered.
YPROCHDR	*FILE	Modified by adding native AS/400 program description to be called by client applications. This file can be moved to another library listed in the jobs LIBL.
YPROCPARMS	*FILE	Modified when AS/400 native programs calling parameters are described. This file can be moved to another library listed in the jobs LIBL.
CFGEAC	*DTAARA	Holds parameters setup by CFGEAC command
CFGEACSSO	*DTAARA	Holds parameters setup by CFGEACSSO command
EASYCOM	*SBSD	Updated when CFGEACSB is used
EAC_EIM	*USRSPC	Holds parameters setup by CFGEACSSO command in *EIM mode.

If the AS/400 EASYCOM server is updated, these objects are eventually upgraded to new format, but the settings are kept except for EASYCOM subsystem.

## IPv6 connectivity

Easycom is fully compatible with IPv6 networking.

There is no special configuration to do to allow IPv6 connections. The only requirement is using a minimum version for EASYCOMD, EASYCOM programs, and the OS/400 version:

- EASYCOM program must be 4.60.10 or above
- EASYCOMD program must be 3.0.3 or above
- OS/400 version must be V5R3 or above

To have full IPv6 connection the Easycom client must also be compatible with IPv6. You need to check the documentation for each client products.

This is recommended to use a name instead of an IP address. However, this is possible to specify an ip address in IPv6 syntax, like follows:

2001:db8::1428:57ab

or

[2001:db8::1428:57ab] :6077

with 6077 as a port number.

You can check whenever the connection was made in IPV6 or not using the NETSTAT command of the OS/400.

You also can see it in the log file (generated by CFGEAC command or by the client application).

Notes :

- The IP address will appear in the IPv6 syntax in all exit programs
- The IP version is available in two new exit programs: EACTCPP01 and EACLOG002
- Connections in V6 and V4 are both accepted by default. You can use EACLOG002 to deny connections if needed.

## Pre-start jobs

"Pre-start" jobs, anticipate job starting, and speedup EASYCOM connection. Its use is particularly well suited if applications are frequently connected and disconnected (i.e., Web applications).

**Advantages:**

- Faster connection Start (quasi-instantaneous connection).



- Possible to initialize a custom environment before the connection is started (however the username is not known at this stage).

### Disadvantages:

- The jobname is equal to the name that was chosen in the ADDPJE command, cannot be changed during the connection.
- The effective user is not visible in WRKACTJOB displayed list except on V5R4 and above. Effective user can be known with WRKACTJOB option 5, and option 1 (job status).

Here is the required commands to setup prestart jobs for Easycom (this will end all active connections):

**ENDSBS SBS(EASYCOM) OPTION(\*IMMED)**

**ADDPJE SBS(D(EASYCOM/EASYCOM) PGM(EASYCOM/EASYCOM) INLJOBS(10) JOB(EASYCOMPJ)  
JOB(D(EASYCOM/EACJOB) CLS(EASYCOM/EACCLS)**

**CFGEACSBS SBS(EASYCOM) MODE(\*PJ) PJSTART(\*YES)**

To return to backward configuration, stop the subsystem, remove PJ definition using RMVPJE, and then run again CFGEACSBS with **PJSTART(\*NO)**.

### *IPL process*

EASYCOM subsystem must be active to accept client connections.  
EASYCOMD job must be active in the subsystem.

See your system administrator to start EASYCOM subsystem during the IPL of your system.  
One way to have EASYCOM started at IPL, is by changing QSTRUP program in QSYS library.  
Retrieve the source of this program  
RTVCLSRC QSYS/QSTRUP ...  
Change the source, by adding command "STRSBS EASYCOM/EASYCOM" after TCP/IP is started.  
And compile:  
CRTCLPGM ...

TCP/IP must be on when Easycom subsystem is started!

Starting Easycom subsystem, automatically starts EASYCOMD job.  
If EASYCOMD is not started, check EASYCOMD \*JOB, and see message queue EACMSGQ in Easycom library:  
DSPMSG EACMSGQ

### *Default LIBL*

Easycom Client job is started with an initial library list in the following order:

- Libraries from System library list.
- Libraries resulting from EACJOB job description, if it exists.
- Libraries resulting from the Job Description assigned to the user.
- Easycom library, if it is not already in one of the above lists.

**Warning!** The initial LIBL was changed with EASYCOM Server **version 4.58.80**

Client application can change the initial LIBL for its Easycom job.  
It uses remote command function API to run ADDLIBL, RMVLIBL or CHGLIBL.

### *Default CCSID, SRTSEQ*

The default behavior of Easycom is using the default CCSID of the system. It is possible to give other values of use the values of the user profile using the [CFGEAC](#) command.

### *Timeout on call program*

Using CFGEAC program allows configuring a maximum execution time (timeout) for a program call.

When the program execution is lasting above configured time, the program call is simply cancelled, and an error is reported to the client.



Warning: when the timeout condition is met, some files and/or resources allocated by the called program may not be released, and never will, until the Easycom job is ended.

### *Protecting access to EASYCOM*

EASYCOM takes care of security at user level: an EASYCOM program can be used on an AS/400 after username and password validation on the system. The processes carried out by the program will be executed under the user identity. All AS/400 permissions of that user will be applied.

All AS/400 rights will be applied.

Each connection opened requires a valid profile and password, or a Kerberos ticket if this kind of connection is allowed and configured.

Advanced security settings is available using the following Easycom Exit Programs:

- [EACP003](#) : authorizes a program according to a complementary password (independent of the profile). This allows locking the easycom server for only a set of applications and/or developers.
- [EACTCP003](#) and [EACTCP002](#) : controls the client settings like IP address, changes effective user or performs specific job submission.
- [EACTCPP01](#): controls security just before validating the login (avoids login exchange if the client is denied from this IP address or protocol).
- [EACLOG002](#): controls security just after the login as been validated by Easycom
- [EACSSO001](#) : controls Easycom Single Sign-On system.

You need to change this value with "Easycom configuration" utility, if the system is the default one, or in the connection properties of your application.

### *Single sign on - EIM*

#### What is EIM ?

The Single Sign On (SSO) in the EIM mode is the implementation of the IBM Single sign-on system.

The main idea is that there is one unique credential management server, Kerberos. When the user is connecting to its station the Kerberos server gives him a **ticket**. When EIM is used during connection that **ticket** is used in place of user/password. This ticket is validated by the Kerberos server (from the iSeries job), and a corresponding OS/400 user is given from the ticket username (the windows login).

So, the user password is not used anymore, and best security is to put **\*\*NONE**" to the password. This way the user **must** use a Kerberos authentication to connect to the system.

#### EIM Installation on IBM i

EIM Installation on AS/400 consists of the following steps:

- Create a domain in EIM
- Add the domain in the domain management
- Create a source user registry definition in EIM.
- Create a user identifier in EIM.
- Create a target association in EIM for the user identifier.
- Create a source association in EIM for the user identifier.
- Test the connection to the EIM domain controller
- Configure the EIM Identity Token Connection Factory

All required information can be found here:

<https://www.ibm.com/docs/en/was/9.0.5?topic=eim-configuring>

Once it works with Client Access you can setup Easycom.





## EIM with Easycom

In order to use EIM with Easycom you need to do the following:

1. Install and configure it in the AS/400 and the domain controller.
2. Grant the TCP user to access the keytab file. QTCP is the user for EASYCOMD job.

```
CHGAUT
OBJ('/QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab')
USER(QTCP) DTAAUT(*R)
```

3. Enable the Kerberos authentication:

```
CFGEACAUTH LIB(EASYCOM) KERBAUTH(*ON)
```

Note: Instead of Kerberos authentication you also can use client certificate authentication, with certificates registered in the EIM database.

4. Configure Easycom to use EIM on the server,

```
CFGEACEIM LIB(EASYCOM) ACTIVE(*YES) EIM_LDAPU(administrator)
EIM_LDAPPW(xxx)
```

5. Optionally define an exit program [EACLOG002](#)
6. Update applications to use EIM by using \*KERBAUTH special value for the login.

EIM implementation on client is very simple. All you need is to specify "\*KERBAUTH" special value for the user id, and a recent client DLL. The password has no importance (can be blank or any value).

There are special TCP/IP error codes (negative) for different Kerberos errors (ticket expired, ...), with corresponding native error text (coming from i5 or from client).

For testing you can type \*KERBAUTH in place of the username and leave a blank password. After this, you can put that special value in your client/server programs.

See also

[EIM Installation on AS/400](#)

[EIM common problems](#)

## EIM common problems

### a. Domain names must match

The domain name that is configured with iSeries navigator must match the domain name of the machine.

If not, you will get an error on the client like: *"the specified target is not known or inaccessible"* (with tcp/ip error code -14)

Here is how to check it:

Step 1: to know what the real domain name is, do the following using a command prompt on the **client machine**:

Enter "nslookup", then type the name of the iseries, like follows:

```
Default server : domain_controller.domain-name.com
Address: 194.206.160.4

> my_iseries
Server : domain_controller.domain-name.com
Address: 194.206.160.4

Name : my_iseries.domain-name.com
Address: 194.206.160.112
```

So here the correct domain name is **domain-name.com**

Step 2: check that exported keytab contains the correct domain name.

Do do this, use iseries navigator, and go to "security", and then "Network authentication service". Right-click and select "Manage keytab". Click on the "Details" button.

You should see a line with:

Principal Type: i5/OS

Principal Name: krbsvr400/my\_iseries.domain-name.com@DOMAIN-NAME.COM



Where DOMAIN-NAME.COM is your i5/OS realm.

If this is not correct, you need to modify configuration and re-export keytab, or you need to check your DNS to have matching domain names.

**b. DES encryption must be enabled on the DC accounts created from keytab.**

If not, you will get an error "*Encryption or checksum type is not supported.*"

To enable it, you need to connect to the domain controller machine, and run the Active Directory application. Then, select "Users", and choose a user named:

my\_iseries\_1\_krbsvr400

(There also can be others: my\_iseries\_2\_krbsvr400, ...)

On the properties of that user, choose "Account", and check "use DES encryption".

**c. Error on connect: "Not authorized to access key table".**

The keytab file must be accessible from the i5/OS account that is used for EASYCOMD, typically QTCP.

You need to know the location of the keytab file. iseries navigator, and go to "security", and then "Network authentication service". Right-click and select "Manage keytab". Follow the wizard until the last step (you can cancel it if you already done the wizard). The keytab file path is specified in that window.

The typical location is:

/QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab

To grant access to QTCP you need to do the following command:

```
CHGAUT OBJ('/QIBM/UserData/OS400/NetworkAuthentication/keytab/krb5.keytab')
USER(QTCP) DTAAUT(*R)
```

**d. The time of all machines must be synchronized.**

If you get errors like '*ticket not yet valid*' or '*ticket is expired*', this is probably due to wrong time synchronization.

Check QTIMZON and QTIME system values using WRKSYSVAL. Also check the time clock and time zone for the domain controller and end-users' machines.

See also

[EIM Installation on AS/400](#)

[EIM with Easycom](#)

## SSL

### SSL connection - prerequisites

Easycom connection can use SSL encryption.

The main prerequisites for using this feature are:

- EASYCOM version must be 4.60.10 or above
- EASYCOMD version must be 3.0.3 or above
- OS/400 version must be V5R3 or above, with i5/OS Host Servers (57xx-SS1 Option 12), Qshell Interpreter (57xx-SS1 Option 30)
- An application ID named 'EASYCOM' must be created in the OS/400, using DCM. A certificate must be assigned to the application.
- System i™ Access for Windows® (57xx-XE1)
- The Easycom server must allow SSL connections using CFGEAC
- The client must support SSL and have the certificate of authority (CA) installed (the CA from which is issued the certificate assigned to the 'Easycom' application).

SSL client support depends on the product versions and on the platforms that are used. You need to check the documentation of the client products.

### SSL connection - server configuration

To enable SSL in Easycom you need to create an **application** and assign a **certificate to it**. The application ID must be equal to Easycom. The certificate must have been issued by a CA that will be accepted by the client.



To create the application, you will need to use the **Digital Certificate Manager (DCM)** of the AS/400. The exact same configuration is required to enable SSL connection with Telnet (apart for client part).

Here are the required steps for the server configuration:

- First, connect to the DCM using a web browser, with [http://my\\_iSeries:2001](http://my_iSeries:2001) and then click on "Digital Certificate manager" (a tip says that it is for creating and managing digital certificates).

If this doesn't work, you will need to enable it using iSeries navigator.

- Then, click on "**Select a Certificate Store**", and select "**\*SYSTEM**", then click "**continue**". This will prompt you to enter the password for the certificate store.
- Then select "**manage applications**" on the left menu and click on "**Add application**". Then select "**Server**", and click "**continue**".

Enter "**EASYCOM**" for the application ID. This is the key that will be used by Easycom. Enter a description and validate.

- Now we need to **assign a certificate to the application**. This is a required step: the certificate is used to ensure that the server can be trusted and also for encryption. There are two options for it:
  1. You can generate the certificate using the AS/400 CA (Certificate of Authority). In this case the CA certificate will need to be installed on the client (first, export the CA certificate using the export menu).
  2. You can request a certificate from a trust 3<sup>rd</sup> party CA. In this case you will need to import it into the \*SYSTEM certificate store using the "import" menu.

To assign the certificate, click on "**Manage Application**", and then "**Update certificate assignment**". Choose "**Server**" and click "**continue**". You will see the current assignment ("*none assigned*") for the application.

Select the 'Easycom' entry that you have created and click on '**Update Certificate Assignment**'. Select the appropriate certificate and click on 'Assign New Certificate'.

Now click on "**Validate**": this will check that the certificate is valid for the system.

- Finally, configure Easycom server to use SSL using [CFGGEAC](#):

```
CHGCURLIB EASYCOM
CFGGEAC LIB(EASYCOM) SSL(*ON)
```

- Then you need to restart EASYCOMD with the following command:

```
STREACD PORT(*JOBID) RESTART(*YES)
```

- then try a connection from a client using SSL. You can use the [Easycom Configuration](#) tool for that.
- You can check the options using the following command:

```
DSPMSG EASYCOM/EACMSGQ
```

This will show:

```
EASYCOMD:Starting from library EASYCOM, Version 3.00.03, (Nov 10 2008
11:15:49/OS530).
EASYCOMD:EASYCOM - (c)AURA Equipments - http://www.easycom-aura.com
----- Lib=
;Pwd=SSL support
EASYCOMD:Configuration used for Library EASYCOM is Dq=
SSL=On
```

In case of problem, the errors will appear here. Note: this does not ensure that the connection is actually in SSL, but only that SSL will be accepted.

To know if SSL is used during a connection, use [FACLOG002](#) exit program. You usually also can check it in the client application.

[Easycom Configuration](#) tool is showing SSL status on the connection test page.

To check it for an active job, look at the call stack of the job. To do this, use WRKACTJOB command, then option 5, and then option 11. If you see "SSL\_Read" in the stack, this means that the connection is using SSL.



## SSL connection - client certificate

Easycom can accept client certificates for two purposes:

- Additional security of the network. The server can give access only to clients that have a valid certificate.
- Use the client certificate to assign the OS/400 user to use. The client certificate subject can be used to define the OS/400 username, or the EIM database can be used for this.

The client certificate must be valid for the AS/400. The certificate is considered valid if it is issued by one of the CA (Certificate Authority) that are installed on the AS/400, in the \*SYSTEM certificate store.

So, the certificate can be issued by the AS/400; in this case the CA is the Local CA.

### Create a X.509 registry in EIM, and configure LDAP location (optional)

This step is required if you want to use the EIM database to map the certificate to the OS/400 user.

In this case the supplied username must be "\*\*SSL".

Using system i access, go to "Network"/"Enterprise Identity Mapping"/"Domain Management"/"<your domain>"/"User Registries", and click "Add a new system registry".

Choose a name, and "X.509" registry type.

Under "configuration", select properties, and select the X.509 registry just created.

Now we need configuring the LDAP location for the \*SYSTEM store. This will make the user certificates creation process linked to the EIM.

Use Digital Certificate manager. Connection is at: [http://my\\_iSeries:2001](http://my_iSeries:2001). Select "Digital Certificate Manager" (on V6R1 select "i5/OS management" and then "Internet configuration" first. Logon as QSECOFR when prompted).

Select "Manage LDAP location", and enter:

LDAP server: fully defined host name: my\_series.mydomain.com

Directory distinguished name (DN): dc=

Use Secure Sockets Layer (SSL): No

Port Number: 389

Login distinguished name (DN): cn=

Password: xxxx (password for LDAP used by EIM).

### Create a user certificate

Go to [https://my\\_iSeries:2010/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0](https://my_iSeries:2010/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0) using the user login for which you want to create the certificate.

Then select "Create Certificate". The login name will be the user under you connected to the web site.

Then click on "install certificate". This will install the certificate into the web browser. Then you can export it into a portable format if needed.

If you created the X.509 registry and specified the LDAP location the DCM configuration, the EIM settings is automatically updated. Note: an EIM mapping MUST exist for this user before doing this (with an i5/OS target equal to that user).

### Install the user certificate on your local store

Use the web browser to transfer the user certificate locally.

### Enable the Easycom server part

CHGCURLIB EASYCOM

[CFGGECAUTH](#) LIB(EASYCOM) SSL(\*ON) SSLAUTH(\*ON) SSLROLE(\*EIM)

Use "SSLROLE(\*EIM)" if you use a X.509 registry or \*SUBJECT if you use the certificate Distinguish name for username.

EIM must be configured with CFGEACEIM as well.

You can try connections with "\*\*SSL" userprofile and no password if EIM is activated, or with a regular user and password if not.

Now type DSPMSG EASYCOM/EACMSGQ. You should see:

```
EASYCOMD:Starting from library EASYCOM, Version 3.00.05, (Jun 23 2009 16:29:38/OS530).
```



```
EASYCOMD:Eim connection OK - X.509 registry is 'p520 certificates'
EASYCOMD:EASYCOM - (c)AURA Equipments -
http://www.easycom-aura.com
=====
EASYCOMD-V.3.00.05 (EASYCOM/EASYCOMD); Lib=EASYCOM; PJ=Off; SSO=Off;
Eim=On; Pwd=2; Port=6077; IPv6; SSL
EASYCOMD:Configuration used for Library EASYCOM is Dq=EASYCOM, Vers=
KerbAuth=Off, SSL=On, SSLAuth=On *EIM
```

This shows the the X.509 (certificates) registry is detected and named 'p520 certificates'.

This also confirms SSL capability for EASYCOMD.

This also shows (from first connection attempt) that the EASYCOM library is with SSL activated, and SSL authentication activated with \*EIM role.

If there is a problem with authentication a message will appear here.

### *EASYCOM jobs on AS/400*

#### EASYCOM jobs on AS/400

When a client application connects to Easycom Server on System I – AS/400, an Easycom Client job is submitted in Easycom Subsystem.

This job run under the authority of the connected user. It can "adopt the authority" of another user on request of the client application.

If exit program EACTCP003 exists in Easycom library, it can submit the client job according to its own rules and descriptions.

If job description EACJOB exists in Easycom library, the job is submitted according to it. Otherwise, it is submitted according to the Users job description.

In any case, user initial library list will be added to the Easycom Client job.

Priority of Easycom Client job is defined by class object EACCLS in Easycom library. This priority can be adjusted with CHGCLS command.

System can use [prestarts jobs](#).

#### Jobs creation and properties

The job alternatively can be created by the safety program [EACTCP003](#) (see below),

If EACTCP003 does not exist or the job does not start, it is created according to:

- EACJOB, if it exists,
- The JOB associated to the user profile that is authenticated, if EACJOB is not present.
- The JOB associated to the user profile that is authenticated for the LIBL management (see [Default LIBL](#))

EASYCOM on AS/400 works using a subsystem and a demon. That daemon handles the connection requests from client applications. When the application is launched, and a connection established with AS/400, a job is created on AS/400. There is an active job for each connected client application, using the appropriate authority and user rights. Each application can have its own file openings, locks, current positions, and transactions in progress.

#### EASYCOM job priority

Jobs are stored in EASYCOM subsystem. It uses **EACJOB** for its description and **EACCLS** for its priority class. Subsystem priority class can be modified with CHGCLS command.

#### EASYCOMD authority

EASYCOMD (\*PGM) is submitted in Easycom subsystem according to EASYCOMD (\*JOB) Job description.

EASYCOMD program has special authorities. Those authorities are necessary to handle security features and submit jobs (or work with prestart jobs) for other users.

To have those features EASYCOMD program is owned by QSECOFR, as "OWNER" user profile and is using 'adopt authority'. By default, EASYCOMD job is submitted under QTCP but using QSECOFR user rights because of those properties.

If EASYCOMD has wrong properties, you can restore them with the following commands:



CHGPGM PGM(EASYCOM/EASYCOMD) USRPRF(\*OWNER) USEADPAUT(\*YES)  
 CHGOBJOWN OBJ(EASYCOM/EASYCOMD) OBJTYPE(\*PGM) NEWOWN(QSECOFR)  
 GRTOBJAUT OBJ(EASYCOM/EASYCOMD) OBJTYPE(\*PGM) USER(QTCP) AUT(\*USE)

## EASYCOM Server configuration commands

### CFGEAC (Configure Easycom)

**CFGEAC** command allows configuring EASYCOM server properties on iSeries - AS/400 system.

```

EASYCOM SERVER CONFIGURATION (CFGEAC)
Type choices, press Enter
Easycom server library name . . . > EASYCOM           Alpha value
Easycom job priority . . . . . *DFT                 1-99, *SAME, *DFT
TCP/IP Keep Alive frequency . . 120                 seconds
Delay before asking again pwd . *NONE                 seconds
Delay before automatic SIGNOFF *NONE                 seconds
Easycom Log File level . . . . *NONE                 Number, *SAME, *NONE
Print the clock in Log File . . *NO                   *SAME, *YES, *NO
Automatic Keep Alive start . . *YES                  Number, *YES, *NO, *SAME
Detailed Job Log . . . . . *NO                   *SAME, *YES, *NO
Lock Easycom host . . . . . *NO                   *SAME, *YES, *NO
Time Out on Ext Pgm Call . . . *NONE                 Number, *SAME, *NONE
Character Set ID . . . . . *USRPRF                 -2-
65535, *USRPRF, *SYSVAL..
Sort sequence table . . . . . *NONE                 Name, *USRPRF, *SYSVAL...
Library . . . . . Name, *LIBL
Convert CONCAT field to A type *NO                   *SAME, *YES, *NO
SSL enable . . . . . *OFF                   *SAME, *OFF, *ON, *ONLY
End
F3=Exit F4= F5=efresh F12=Cancel F13=How to use this display
F24=More keys
    
```

#### EASYCOM server library name (LIB)

Enter the library name in which the EASYCOM server is installed.

#### EASYCOM server job priority (PTY)

This parameter is used to override JOBDD job priority setting. If set to 0, JOBDD determines job priority. JOBDD used with EASYCOM is EACJOBDD.

#### TCP/IP Keep Alive frequency (TCPTOUT)

This parameter is used to set the 'keep alive' interval value. Default value is 120 seconds. When 'keep alive' is on, a TCP/IP message is sent from PC to AS/400 every n seconds. This is useful to keep a remote line up, and to have automatic shutdown of jobs that are no longer linked to a client application, even in case of client crash.

If the AS/400 EASYCOM job does not receive the message in n+10 seconds, it automatically shutdowns.

These TCP/IP messages are only sent when the communication is idle for that delay.

This value can be set to 0 to disable it. This is useful when debugging, as some debuggers avoids Easycom to send the TCP/IP message when the process is stopped.

This parameter can also be set using 'Easycom configuration' tool on the PC.

#### Delay before asking again pwd (RESIGN)

This parameter is used to make end-user sign-on again after a given idle time. (Default is disabled).

*Not currently supported.*

#### Delay before automatic SIGNOFF (CONNECTION)

This parameter is used to close a connection after a given idle time.

#### EASYCOM Log File level (LOGLEV)



Use this to enable an AS/400 log file. Valid values are 1 to 4.  
It will create a EASYCOM/LOGFILE(MEMBER) file, EASYCOM is the Easycom installation library, MEMBER is Easycom job name.  
Be careful with that:

- Log file member is always cleared when a new connection is made
  - If two jobs with same job's name are run, the second cannot have log file and will be locked for 1 minute at start-up.

#### **Print clock in Log File (LOGCLOCK)**

Allows getting time information in log file: command processing starting and ending time, CPU consuming.

#### **Automatic Keep Alive start (HBEAT)**

If this value is \*YES, the 'keep alive' message (see above) will be generated unless the PC is configured to refuse it.  
If this value is \*NO, 'keep alive' will not start unless the PC is configured to enable it.

#### **Detailed Job Log (JOBLOG)**

This option is used to run an automatic job login. This can be changed with Easycom JOBID (EACJOBID).

#### **Lock EASYCOM host (LOCKED)**

Easycom is default locked if this option is used. This means that the Easycom connection is accepted, but no file neither program access will work until the 'unlock' password arrives. See our documentation about EACP003 entry program for more information.

#### **Time Out on Ext Pgm Call (PGMTOUT)**

Defines a timeout for program execution. This avoids program call taking too much time. When the timeout is reached, the call will abort and Easycom will return an error.

#### **Character Set ID (CCSID)**

Indicates character set used which EASYCOM. Default character set is \*HEX (65535).

A good idea can be to set it up to \*USRPRF.

#### **Sort Sequence table (SRTSEQ)**

Indicates the sort of file to be used for comparison and sorting. Possible values are those suitable for SRTSEQ parameter in the system CHGJOB command. \*LANGIDUNQ is a value that allows "natural" sorting for the current CCSID. However, there is a need to be careful to have the indexes or logical files with a compatible sort of sequence.

#### **Convert fields CONCAT to type A (CONCATF)**

Indicates if CONCAT operations resulting fields must always be considered as alphanumeric type fields. Possible values are:

\*YES: CONCAT result fields will be processed as a single alphanumeric type field.

\*NO: CONCAT result fields keep their original type.

#### **SSL enable (SSL)**

Specifies how SSL encryption can be used with Easycom.

Use DSPMSG EACMSGQ to know if SSL init worked.

Note: Modifying this option requires EASYCOMD job restart. You can perform it using STRSBS/ENDSBS system commands or the STREACD command.

Possible values are:

\*YES: Both SSL encrypted, and clear connections are accepted.

\*NO: SSL is not used on this library. SSL connection attempts will fail.

\*ONLY: SSL usage is mandatory. The connection will fail if the client does not support SSL, or if SSL negotiation failed.



## CFGFACTCP (Configure Easycom TCP/IP)

**OBSOLETE: use CFGEACSBS**

This command is automatically called when automatic installation is performed.  
It creates the subsystem and all the related objects, and it sets the TCP/IP port number used by Easycom service.

Objects created: SBSDB, JOBD, JOBQ, CLS.

Object name	Object type	Description
EASYCOM (default)	*SBSDB	Subsystem in which service is running, and client jobs are submitted.
EACJOBDB	*JOBDB	Descriptions for client jobs.
EACJOBQ	*JOBQ	Client job queue.
EACCLS	*CLS	Class for client jobs.
EASYCOMD	*JOBDB	Job description for the EASYCOMD job, which must always be active in the subsystem.

Note: this command creates the required objects to have a subsystem running and started it. But it **doesn't store any of the parameters**.

### EASYCOM library (LIB)

Enter EASYCOM server objects library name, where new sub-system will be created, with all related objects.  
If the objects already exist in the library, they will be replaced.

### System library in LIBL (SYSLIB)

This parameter is no used.

### EASYCOM sub-system name (SBS)

Enter the subsystem name to be created in the library.

The subsystem name must be unique on the system. If you have more than one Easycom Server running on the system, each server must have its own library and subsystem. See [Installing an additional Easycom Server](#).

When the subsystem will be active, job associated with each connection will run in this subsystem.

### EASYCOM service port (PORT)

Enter the TCP port number to be assigned to the EASYCOM server. If multiple EASYCOM servers will run on the same machine, a different port number must be assigned to each one.

Possible values are:

**\*DFT:** If a service named easycom exists in the port services table, the associated port will be used. See WRKSRVTBLE system command to manage the services table. If Easycom service does not exist, **default port 6077** is used.

**Number:** port number to be allocated to new EASYCOM server.

If port number is changed, it must be changed in the client configuration, using "Easycom configuration utility", or by changing the connection properties in your client application.

When a non default port number is configured, port number must be added at the end of the server's name or address in the client application, separated by a colon (:).

Example: my\_server:6090

### Authorize pre started jobs (PJ)

Possible values are:

**\*OFF:** "Prestart" jobs are not used when client session requires a connection, even if they are configured in the subsystem.

**\*ON / \*AUTO:** To use pre-started jobs on the subsystem, if they are configured and active.





This option only authorizes Easycom to use pre started jobs at connection time if they are active.

You need to configure the pre started jobs manually (available on V4R4 and above).

To configure pre started jobs, after running CFGEACSBS command, you need to proceed as Follow:

**Stop EASYCOM subsystem.**

Example: ENSSBS EASYCOM \*IMMED

**Run ADDPJE command.**

Example:

```
ADDPJE SBS (EASYCOM/EASYCOM)
PGM (EASYCOM/EASYCOM)
USER (QUSER) INLJOBS (4)
JOB (PJEASYCOM) JOBD (EASYCOM/EACJOB)
CLS (EASYCOM/EACCLS)
```

**Start subsystem again:**

STRSBS EASYCOM/EASYCOM

**Comments:**

Command CFGEACSBS starts the Easycom subsystem.

TCP/IP must be on when subsystem is started!

Starting Easycom subsystem, automatically starts EASYCOMD job.

If EASYCOMD is not started, check EASYCOMD \*JOB, and see message queue EACMSGQ in Easycom library:

DSPMSG EACMSGQ

## ***CFGEACSBS (Configure Easycom Subsystem)***

This command configures the EASYCOM subsystem, and the way Client jobs are started in the subsystem.

**EASYCOM Subsystem name (SBS)**

Enter the name of the Easycom subsystem to configure, and its library name.

**Server Job Default Start Mode (MODE)**

How Easycom Client jobs are started in the subsystem when Client connects.

Possible values are:

**\*SAME:** Do not change actual mode.

**\*BCH:** On Client connection, a new batch job is submitted.

**\*BCI:** On Client connection, a new batch immediate job is started.

Connection time is shorter using \*BCI mode.

**\*PJ:** On Client connection, Easycom tries to connect to a pre-started job.

If no pre-start job is available, a BCI is started.

**Easycom service port number (PORT)**

The TCP/IP port number you want to allocate to the Easycom host server.

If you want to run multiple Easycom servers on the same IBM i machine, you need to allocate a different port number to each one.

Possible values are:

**\*SAME:** Do not change actual port setting.

**\*DFT:** If a service easycom exists in the service table, the port number associated with that service will be used.

See system command WRKSRVTBLE to manage the service table.

If service Easycom doesn't exist, the default port number 6077 is used.

**Number:** A port number to allocate to the new Easycom host server.



**Server Job Description (JOBDD)**

Description used when Easycom jobs are submitted, when MODE is set to \*BCH.

Possible values are:

**\*SAME:** Do not change actual JOBDD setting.

**\*DFT:** Use Default EACJOBDD Job description.

**\*USRPRF:** Use Job Description from user profile.

**JobDD Name:** Set the name and library of the Job Description to use when Easycom jobs are submitted.

**Initial Prestart Jobs Number (INLJOBS)**

Specifies the initial number of prestart jobs that are started when the subsystem is started.

Possible values are:

**\*SAME:** Do not change actual setting.

**\*NONE:** Do not start prestart jobs when subsystem is started.

**1-9999:** Specify the number of prestart jobs that are started when the subsystem is started.

**Prestart Jobs Threshold (THRESHOLD)**

Specifies when additional prestart jobs are started.

When the pool of available jobs is reduced below this number, more jobs are started and added to the available pool.

Possible values are:

**\*SAME:** Do not change actual setting.

**1-9999:** Specify the minimum number of prestart jobs that must be available before additional prestart jobs are started.

**Additional Prestart Jobs (ADLJOBS)**

Specifies the additional number of prestart jobs that are started when the number of prestart jobs drops below the value

specified on the Threshold parameter.

Possible values are:

**\*SAME:** Do not change actual setting.

**1-9999:** Specify the number of additional prestart jobs to start.

**Prestart Jobs User Profile (PJUSER)**

Specifies the name of the user profile under which the prestart job is initiated

Possible values are:

**\*SAME:** Do not change actual setting.

**QUSER:** The IBM-supplied QUSER user profile is used.

**name:** Specify the name of the user profile used for the prestart job.

**Name of Prestart Jobs (PJJOB)**

Specifies the name of the prestart job that is started.

Possible values are:

**\*SAME:** Do not change actual setting.

**name:** Specify the name of the prestart job.

**Prestart Job JOBDD (PJJOBDD)**

Specifies the qualified name of the job description used for the prestart job.

Possible values are:

**\*SAME:** Do not change actual setting.

**\*DFT:** Default job description EACJOBDD is used to start prestart jobs.



**name:** Specify the name of the job description being used for this prestart job, and its library name.

#### **Start Prestart Jobs with Sbs (PJSTART)**

Specifies whether the prestart jobs should be started at the time the subsystem is started.

Possible values are:

- \***SAME:** Do not change actual setting.
- \***YES:** The prestart jobs are started at the time the subsystem is started.
- \***NO:** The prestart jobs are not started at the time the subsystem is started.

#### ***STREACD (EASYCOM service start)***

**STREACD** command starts EASYCOM service. EASYCOMD program is started in the subsystem to allow connection of clients' stations.

Remark: this command **doesn't store any of the parameters**.

#### **EASYCOM Library (LIB)**

Enter EASYCOM server library name, where subsystem description was created.

#### **EASYCOM service port (PORT)**

Enter the TCP port number assigned to EASYCOM server. If several EASYCOM servers will run on the same machine, a different port number must be assigned to each one. Possible values are:

\***DFT:** If a service named Easycom exists on the port services table, the associated will be used. See WRKSRVTBLE system command to manage the services table. If Easycom service does not exist, **default port 6077** is used.

\***JOB:** The service is started according to EASYCOMD job description in the library.

Number: Port number to be allocated to new EASYCOM server.

#### **Authorised pre-starts jobs (Pre-starts jobs - PJ)**

This parameter is used only if PORT parameter is different from \*JOB. Use pre-starts jobs in the subsystem.

Possible values are:

\***OFF (default):** "Pre-start" jobs are not used when client session requires a connection, even if they are configured in the subsystem.

\***ON / \*AUTO:** To use pre-started jobs on the subsystem if those are configured and active.

#### **EASYCOMD restart (RESTART)**

Stop and Start again EASYCOMD job if it is already running in the subsystem.

Possible values are:

\***NO:** If EASYCOMD job is already active, it remains unchanged.

\***YES:** If EASYCOMD is running, it is stopped, then re-started with new parameters.

EASYCOM subsystem must be active.

EASYCOMD job (demon) runs permanently in the EASYCOM subsystem. It starts automatically when subsystem is started.

EASYCOMD uses TCP/IP port 6077 (default) to accept connection requests from client stations.

If a safety system or another application prohibits using this port, it can be modified with CFGEACSBS command.

#### **Comments:**

TCP/IP must be on when EASYCOMD is submitted.

If EASYCOMD is not started, check EASYCOMD \*JOB, and see message queue EACMSGQ in Easycom library:

DSPMSG EACMSGQ



### EACINSTALL (Easycom Install)

This command is the final setup command. This command updates Easycom objects to have the best possible match according to the current running OS/400 release.

This command changes the default SQL interface, and EASYCOMD program to support EIM.

```

EASYCOM INSTALLATION (EACINSTALL)

Type choices, press Enter.

Easycom Library . . . . . EASYCOM Lib. of product EASYCOM
OS VERSION FOR ADJ. . . . . *AUTO          MINIMUM OS VERSION FOR ADJ
LEVEL OF SQL INTERFACE TO USE . . . *AUTO          *CISC, *EMBED, *CLI, *AUTO
    
```

You can change the default SQL INTERFACE from \*CLI to \*EMBED. This will use the embedded SQL interface in replacement of \*CLI.

The \*CLI interface is more powerful but using \*EMBED can help solving issues that are encountered by \*CLI interface. The \*CISC interface is obsolete and is no longer included in latest versions of Easycom.

The \*EMBED interface limitations are: cannot use LOB fields, or SQL procedures. However, in some cases it is faster than CLI.

In fact, the \*EMBED is the old - historical – interface, and \*CLI is the one. Only the \*CLI interface will have future improvements.

### CFGEACAUTH

This command configures the authentication methods and security options which are valid with Easycom.

```

Easycom Authentication config (CFGEACAUTH)

Type choices, press Enter.

Easycom server library name . . > EASYCOM Alpha value
Use SSL encryption . . . . . *OFF          *SAME, *OFF, *ON, *ONLY
Use SSL authentication . . . . . *OFF          *SAME, *OFF, *ON, *ONLY
SSL authentication role . . . . . *SAME
Use Kerberos authentication . . *ON          *SAME, *OFF, *ON, *ONLY
    
```

#### Use SSL encryption

This option defines if the SSL encryption is supported, or mandatory. Possible values are:

\*OFF: SSL is not used by the Easycom server.

\*ON: SSL is used if requested by the client

\*ONLY: SSL must be used. The connection will be rejected if the client doesn't support SSL or if the SSL negotiation fails.

#### Use SSL authentication

This option defines if SSL authentication is enabled. This option is valid is 'Use SSL encryption' is activated. Possible values are:

\*OFF: SSL authentication is not accepted.

\*ON: SSL authentication is valid. A valid certificate must be provided by the client.

\*ONLY: SSL authentication is mandatory. A valid certificate must be provided by the client. This SSL authentication can validate the OS/400 user or can only act as an additional security option (see 'SSL authentication role').

#### SSL authentication role

This option defines how the SSL authentication will imply an OS/400 user. Possible values are:

\*NONE: the SSL authentication won't define an OS/400 user. The client certificate will be checked by Easycom, but not used to define the OS/400 User. OS/400 User and password, or Kerberos authentication must be provided as well.

\*EIM: Easycom will search if the client certificate is found in the EIM database. If yes, the EIM will define which user to use. In this case EIM configuration must be valid.

\*SUBJECT: the certificate subject is equal to the OS/400 username. In this case the EIM configuration is not necessary. The SSL client certificate will be use for the whole authentication process.

#### Use Kerberos authentication



This option defines if the Kerberos authentication is valid. The EIM configuration must be valid to be able to map the Kerberos authentication (typically Windows credentials) to an OS/400 user.

### CFGEACEIM

This command is designed to configure the EIM connection for Easycom. It replaces the CFGEACSSO command, which is now obsolete.

The EIM system is used to define an OS/400 user from another authentication.

EIM can be seek the OS/400 user from different sources:

- from the Kerberos authentication. This allows single signon (SSO)
- from SSL client certificate authentication

The [CFGEACAUTH](#) command defines which kind of authentication are valid.

```

Easycom EIM Configuration (CFGEACEIM)

Type choices, press Enter.

Easycom server library name . . . > EASYCOM Valeur alpha
Use EIM in Easycom . . . . . *YES          *YES, *NO, *SAME
EIM valid from . . . . . *NONE          HHMM =
EIM valid to . . . . . *NONE          HHMM =
LDAP user for EIM . . . . . 'administrator'
LDAP password for EIM . . . . .

EIM logon is mandatory . . . . . *NO          *YES, *NO
LDAP dn for EIM . . . . .
LDAP service spn . . . . .
    
```

#### Use EIM in EASYCOM

This is the main option for enable EIM on Easycom or not. Must be \*YES to enable the other options.

#### SSO authorized from / SSO authorized to

EIM 'opening hours. EIM connections are forbidden outside of those hours.

#### LDAP user for EIM

Local LDAP user. This username is required during a connection attempt, to retrieve the "OS/400" username associated to the "Windows" user name.

This local username is the name used when configuring EIM with iSeries Navigator (when selecting NetWork/EIM Domain Mapping/Domain Management/<yourDomain>).

You need to only put the username, not "cn="

#### LDAP password for EIM

This is the password for the local LDAP connection.

#### EIM logon is mandatory

Configures EASYCOM to deny all non-EIM connections (with username/password).

#### LDAP dn for EIM

This is a alternate way for giving LDAP logon name, allowing specific syntax. So, this is valid only if user is left blank. A typical value is:

cn=

#### LDAP service spn

This allows a specific service principal name. If \*DFT is specified, Easycom calculates it using "krbsvr400" and the system name.

Example of valid values (with systemi5 name for the system, testdomain.com for the domain and TESTDOMAIN.COM for the realm):

krbsvr400/systemi5

krbsvr400/systemi5@TESTDOMAIN.COM

krbsvr400/systemi5.testdomain.com@TESTDOMAIN.COM (default if \*DFT is specified)



## Exit Programs

### Exit Programs

[EASYCOM](#) offers many programs called « Exit Programs ».

Those kinds of programs follow a given specification and must be implemented by the administrator of the AS/400, allowing a most advanced control and security of easycom connections and usage.

Some of them must be written in some configuration cases like the exit programs related to [Single Sign On](#) or to the Easycom lock ([Lock EASYCOM Host](#)).

Others are related to a specific configuration but are not mandatory, like with [Prestarts Jobs](#) use.

The others are not mandatory and are designed to have better security and control.

Sample sources are provided in **EACSYSSRC** source file in Easycom library.

### Easycom startup

#### Starting Client Job - EACSTART

If a program named EACSTART exists in the job libraries list (LIBL), it is called each time EASYCOM client job is submitted.

It is particularly useful to set properties or perform maintenance actions.

This program is called when the user is known. It can still modify attributes or parameters but cannot cancel the job, except by hardly killing it.

EACTCP003 is to be preferred to control user rights and eventually cancel the job.

#### Prestart job initialization - EACPJINI

If Pre-starts Jobs are activated in EASYCOM server configuration and if EACPJINI program exists in the job library list, it is called each time Pre-start Job is started by the system.

EACPJINI offers the possibility to define job properties when the job is created. At that time the connected user is unknown.

EACTCP002 will be called at connection time.

#### SQL initialization - EACSQLINI

This exit program is called when Easycom is using the SQL interface for the first time in the job (between SQL initialization and actual SQL usage, like SQL query prepare)

If using pre-start jobs, it is called **before the connection is made (SQL is initialized at this moment to reduce the connection delay)**, during the pre-start process; otherwise, it is called when the SQL for the first time (so it is never called if SQL is not used by the application).

This exit program can be used to check the environment at this point.

### Logon and access security

#### Connection control - EACTCPP01

This exit program is designed to control the connection before any authentication. This can deny connection before any password or ticket exchange is made.

This can also be used to control whenever the connection must or can be made using SSL.

```
PGM PARM(&LIB &TPNAME &RMTADDR &IPVERSION +
&SSLASK &SSLCNF &VALID)
DCL VAR(&LIB) TYPE(*CHAR) LEN(10)
DCL VAR(&TPNAME) TYPE(*CHAR) LEN(30)
DCL VAR(&RMTADDR) TYPE(*CHAR) LEN(50)
DCL VAR(&IPVERSION) TYPE(*CHAR) LEN(1)
DCL VAR(&SSLASK) TYPE(*CHAR) LEN(1)
DCL VAR(&SSLCNF) TYPE(*CHAR) LEN(1)
DCL VAR(&VALID) type(*CHAR) len(10)
```

**&LIB** is the library that when the Easycom program is. Usually, Easycom.

**&TPNAME** is the name of the Easycom program. By default, this is Easycom.



**&RMTADDR** is the TCP/IP address of the connection request. This can be in IPV4 or IPV6 form depending on &IPVERSION value.

**&IPVERSION** is equal to 4 or 6 depending on the IP version currently in use for the connection (if the AS/400 supports it, Easycom will accept both protocols by default)

**&SSLASK** informs if the client will try to negotiate an SSL connection. Possible values are:

- 'Y': the client supports SSL, and if the server accepts it, the connection will be made using SSL. In other words, the connection will maybe use SSL.
- 'N': the client is not supporting SSL or doesn't ask to use it. In other words, the connection won't use SSL in any case.

**&SSLCNF** informs if the SERVER will or supports SSL. Possible values are:

- 0: the server won't use SSL at all (even if supported)
- 1: the server may use SSL if SSLASK=Y. If SSL negotiation fails, the connection will remain valid.
- 3: the server will use SSL. If SSLASK=N or if the SSL negotiation fails, the connection will be aborted.

**&VALID** is used to tell EASYCOMD to grant or deny the connection. Possible values are:

- \*YES: the connection process can continue
- \*DENY: the connection is aborted immediately. An error message will be prompted on the client.

Note: only **&SSLCNF** and **&VALID** can be modified by the exit program.

## Logon control - EACLOG002

EACLOG002 is an exit program for general authentication process.

This program is called **after** the authentication made by Easycom.

This exit program is called on all authentication situations (normal, SSO, and EIM).

It can be used to audit the Easycom usage and/or deny connections from custom criteria.

EACLOG001 is the previous version of EACLOG002; it won't be called if EACLOG002 is implemented.

EACLOG002 has only two more parameters for IP version and SSL condition.

The prototype is:

```
PGM PARM(&LOGTYPE &RC &LOGUSER &LOGDOMAIN &USER
&IPADDR &STATION &IPVERSION &SSL)
```

```
DCL VAR(&LOGTYPE) TYPE(*CHAR) LEN(10)
DCL VAR(&RC) TYPE(*CHAR) LEN(10)
DCL VAR(&LOGUSER) TYPE(*CHAR) LEN(130)
DCL VAR(&LOGDOMAIN) TYPE(*CHAR) LEN(130)
DCL VAR(&USER) TYPE(*CHAR) LEN(10)
DCL VAR(&IPADDR) TYPE(*CHAR) LEN(130)
DCL VAR(&STATION) TYPE(*CHAR) LEN(130)
DCL VAR(&IPVERSION) TYPE(*CHAR) LEN(1)
DCL VAR(&SSL) TYPE(*CHAR) LEN(1)
```

**&LOGTYPE** is input, and tells which logon is being processed. The possible values are:

- \*STD: this is a standard login/password logon (&LOGUSER and &LOGDOMAIN are not available)
- \*EIM: this is an EIM logon. No password is available. &LOGUSER, &LOGDOMAIN and &USER are applicable.
- \*SSO: this is an Easycom kind SSO. All fields are available.

**&RC** is the result of the command. This can be used to deny the user or indicate that the OS/400 user was changed.

The possible values are:

- \*OK: the logon remains granted
- \*CHG: the &USER parameter is changed by the exit program. Note: the &USER user will not have a password validation.
- \*OUTOURS: the logon is rejected because of hours of work.
- \*DENY: the logon is denied.

**&LOGUSER** is the Windows username. This is filled only in \*EIM or \*SSO mode for &LOGTYPE.



**&LOGDOMAIN** is the Windows domain. This is filled only in \*EIM or \*SSO mode for &LOGTYPE.

**&USER** is the OS/400 user. This is the OS/400 user under which the Easycom job will run.

**&IPADDR** is the IP address of the client connection. This can be used to filter access or for auditing.

**&STATION** is a string that represents the station of the client connection. This can be the real machine name (the name that corresponds to the IP address) or the Terminal name, if the connection is made thru an RDP connection.

**&IPVERSION** is equal to 4 or 6 depending on the TCP/IP network version used for connection. (IPv4 or IPv6)

**&SSL** is equal to 'Y' if the connection is using SSL and 'N' if not. SSL negotiation is already made at this time.

### Security by restriction - EACTCP003

This exit program is designed for limiting EASYCOM use of to a user and/or a PC group.

If EACTCP003 program exists in EASYCOMD library list, it will be called at each connection attempt, excepted if EASYCOM is configured to use pre-starts Jobs (in this case [EACTCP002](#) can be used).

This program can allow or deny the connection from the client application.

If connection is accepted, it can submit by itself the client job or let Easycomd doing it.

**&JOBNAME** variable is used to determine what is decided:

- \*YES to accept the connection, but submit the job in the exit program.
- \*NO to refuse the connection
- Any value to let easycomd submit the job with that name.

Note: the initial value is equal to the jobname that is calculated during the connection, usually the name of the client pc if it is possible to use it as a jobname (or the jobname decided by the client application).

Program specification:

**PGM PARM(&TPPGM &TPLIB &USER &EAC\_PARM1 + &EAC\_PARM2 &RMT\_ADR &JOBNAME)**

**DCL VAR(&TPPGM) TYPE(\*CHAR) LEN(10)**

**DCL VAR(&TPLIB) TYPE(\*CHAR) LEN(10)**

**DCL VAR(&USER) TYPE(\*CHAR) LEN(10)**

**DCL VAR(&EAC\_PARM1) TYPE(\*CHAR) LEN(30)**

**DCL VAR(&EAC\_PARM2) TYPE(\*CHAR) LEN(30)**

**DCL VAR(&RMT\_ADR) TYPE(\*CHAR) LEN(50)**

**DCL VAR(&JOBNAME) TYPE(\*CHAR) LEN(10)**

Parameters:

**TPPGM:** Target Program Name

**TPLIB:** Library containing TPPGM program.

Parameters TPPGM and TPLIB will be used by the EACTCP003 program if it submits the client job by itself.

**USER:** Username

New client connection username (can be used to limit access to a user group).

**EAC\_PARM1:** Parameter 1 of TPPGM program

First parameter to pass to target program (TPPGM) if EACTCP003 submits the client job by itself.

**EAC\_PARM2:** Parameter 2 of TPPGM program

Second parameter to pass to target program (TPPGM) if EACTCP003 submits the client job by itself.



**RMT\_ADR:** TCP/IP client address  
 TCP/IP client address (may concern workstations set).

**JOBNAME:** SBMJOB job name (Input / Output).  
 Name of the job to be activated in EASYCOM subsystem. Default name is the client station name.  
 On return, set JOBNAME parameter to:  
 \*NO, to refuse the connection.  
 \*YES, if EACTCP003 has submitted the client job by itself.  
 A name, or leave it unchanged, to accept the connection and let Easycom submit the client job.

**Comments:**

This exit program can be used to check the validity of the user id or tcp/ip address.  
 It can also submit the job under the authority of a user different from the one requesting the connection.  
 Or it can also submit another program, different from TPPGM, in order to setup some environment properties before calling TPPGM.

**Prestart job control - EACTCP002**

EACTCP002 works the same way as as [EACTCP003](#) when Pre-starts Jobs are activated.  
 Since the job is already initialized, EACTCP002 does not create it but allows or refuses its start. It also permits controls or treatments prior initialization.  
 Note: this exit program is also called when not using prestart jobs.

**'Program Level' Security - EACP003**

In addition to basic safety, **programs level** safety can be used.  
 Only programs validated by data processing department can be used on AS/400.  
 Unauthorized EASYCOM programs may be connected to AS/400 but will be unable to make any operation (file opening, program calling or other).  
 Authorized program will send a special password to EASYCOM. A data processing department AS/400 program returns information telling if password is accepted. This password can be similar, for example, to EASYCOM program coding.

**To activate this mechanism:**

If '**Lock EASYCOM host**' entry is set to \*YES in CFGEAC, no file can be opened, no program can be called, no command can be sent to AS/400 by EASYCOM, until the client application frees it sending a password to it.  
 This option requires writing an EACP003 script. This script must be located in EASYCOM job LIBL.  
 Warning, if option is activated and script does not exist, EASYCOM will remain locked, and no job can be created.

Here is this script layout:

```
PGM PARM(&PASSW &RESULT)
DCL VAR(&PASSW) TYPE(*CHAR) LEN(100)
DCL VAR(&RESULT) TYPE(*CHAR) LEN(10)
...
/* IF PASSW HAS THE RIGHT VALUE */
CHGVAR VAR(&RESULT) VALUES(*YES)
...
/* IF PASSW DOES NOT HAVE THE RIGHT VALUE */
CHGVAR VAR(&RESULT) VALUES(*NO)
```

It receives a single-entry parameter (&PASSW applicative password different from profile password). It returns &RESULT parameter.

- \*YES value authorizes job starting and process to continue.
- \*NO value locks the job.

### Easycom mode single signon - EACSSO001

It's the Exit Program associated to Single Sign-On activation in easycom mode. This is not called in EIM mode.

Note: this is recommended to use the EIM mode single signon instead of the Easycom mode.

When Single Sign-on is configured and activated (see [CFGFACEIM](#)) and if program EACSSO001 exists in the job libraries list, it runs with various events:

- before memorizing a signature (simple connection or Windows session)
- when recording (simple connection or Windows session)

then with each connection request.

#### Parameters

EASYCOM calls the program, ,transfers various parameters to it and turns over.

**&OP - Operation:** program call origin

*\*BEFORE and \*WINBEFORE*

Before memorizing simple or session signature, the program can:

- modify username and/or password
- authorize or refuse memorizing

*\*SIGNON and \*WINSIGNON*

Signature memorizing, the program :

- can't modify user or password anymore,
- can authorize or refuse memorizing

*\*REQUEST*

Requires connection, the program :

- cannot modify user or password anymore,
- can erase storage and force user to be signed again.

**&RC - Return**

\*OK : accepts signature

\*DENY : refuses signature

\*EXPIRED : signature validity period is exceeded

\*OUTHOURS : request out of authorized hours,

\*CHG : user change

**&USER / &USERLEN - user name length**

**&PWD / &PWDLEN - password length**

&SOTIME – Time in HHMMSS format

&SODATE - Date in CYYMMDD format

&IDADR - IP client address

&STATION - workstation (different from &computer if TSE is used)

&COMPUTER – computer name

&LOGDOMAIN – Windows domain

&LOGUSER – Windows user



The fat variables (except &OP) can be modified with &RC program (to authorize or refuse signature or connection, change user, expiration or out of authorized domains), &USER and &PWD for a user change.

See EASYCOM library EACSSO001 file for an example and more detailed specifications.

### *Objects and programs security*

#### EACSOPEN - File open, SQL queries

EACSOPEN exit program is called, if it exists in the client job LIBL, each time a file open is requested by the client job, or a SQL statement is prepared or immediately executed.

The exit program can refuse the file operation, or it can change file name or SQL statement.

See source example in EACSYSSRC file, Easycom library.

#### EACSRCMD - Remote command

EACSRCMD exit program is called, if it exists in client job LIBL, each time a command is submitted by the client application, with Easycom function API.

Exit program can refuse execution of the command, or it can replace the command before returning.

See source example in EACSYSSRC file, Easycom library.

#### EACSCALL - Program Call

Exit Program EACSCALL will be called, if it exists in the client job library list, each time an external program or procedure is called by the client application, using Easycom API.

This exit program can refuse the program or procedure call by the client application.

It can also change the program name, library name or procedure name on return, so that the client application will call another program.

See source example in EACSYSSRC file, Easycom library.

#### EACSIFS - IFS access

This exit program is called on each IFS file open.

The parameters are the file path and open mode. The open mode is a numeric value that is a combination of the following constants (hexadecimal):

- \_EAC\_IFSOPEN\_READ=1 read access
- \_EAC\_IFSOPEN\_WRITE=2 write access
- \_EAC\_IFSOPEN\_CREAT=4 file will be created if not exist
- \_EAC\_IFSOPEN\_EXCL=8 file must not exist before open (create is mandatory)
- \_EAC\_IFSOPEN\_TRUNC=10 truncate file
- \_EAC\_IFSOPEN\_APPEND=20 append file
- \_EAC\_IFSOPEN\_BINARY=40 binary mode
- \_EAC\_IFSOPEN\_BIGFILE=80 big file. Allows to open > 2Gb files

Create mode:

- \_EAC\_IFSMODE\_RUSR 400 user can read (u+r)
- \_EAC\_IFSMODE\_WUSR 800 user can write (u+w)
- \_EAC\_IFSMODE\_XUSR 1000 user can execute (u+x)
- \_EAC\_IFSMODE\_RGRP 2000 group can read (g+r)
- \_EAC\_IFSMODE\_WGRP 4000 group can write (g+w)
- \_EAC\_IFSMODE\_XGRP 8000 group can execute (g+x)
- \_EAC\_IFSMODE\_ROTH 10000 others can read (o+r)
- \_EAC\_IFSMODE\_WOTH 20000 others can write (o+w)
- \_EAC\_IFSMODE\_XOTH 40000 others can execute (o+x)

Share mode:

- \_EAC\_IFSSHARE\_RDONLY 100 0000 read only share
- \_EAC\_IFSSHARE\_WRONLY 200 0000 write only share
- \_EAC\_IFSSHARE\_NONE 400 0000 no share (exclusive)



`_EAC_IFSSHARE_RDWR 300 0000 read/write share`

If you need to test the open mode, you need to use a bitwise AND with the flag to test and see if the result is equal to that flag.

Note: the exit program can only deny or accept the file open.

A source sample is available in the EACSYSSRC file in EASYCOM library.

## EASYCOM Configuration

This is the PC side centralized management tool for EASYCOM native access. The server part configuration is executed from a terminal or an emulator. All options are stored in `easycom.ini` file on Windows.

Unix versions (AIX, Linux or other) use the `/etc/easycom.conf` file, with the same syntax.

This configuration file can be general (in Windows directory, `C:\WINDOWS`) or be specific to an application (in the executable directory).

Using this file is optional for deployment. This is used only for convenience, allowing to avoid having connection parameters managed by the application program itself.

This tool contains the following tabs:

- Connection parameters
- EASYCOM Activation key
- Trace file
- Default settings
- Security
- Checking installation and versions

## Connection parameters

### *AS/400 name or IP address*

Machine name or TCP/IP address for the AS/400 is entered here.

Use of a name implies a DNS configuration or hosts file.

The **port number** can be specified with `":portnum"`, for example: `iseries:6078` to have 6078 port number.

The **easycom** service name is used to setup the default port; and if no service is defined, the 6077 port number will be used.

### *EASYCOM Server*

- **Default (EASYCOM/EASYCOM)**

Use the default server program: (EASYCOM/EASYCOM)

- **Other**

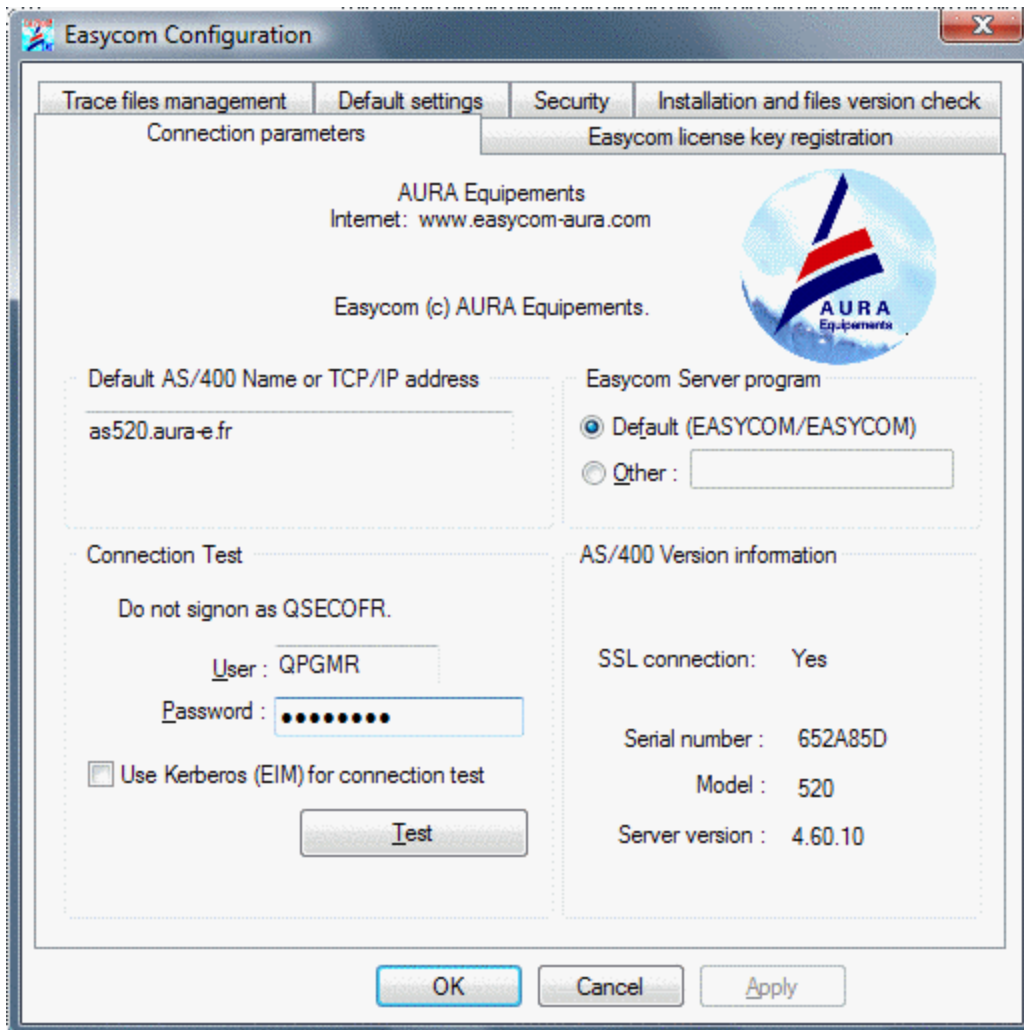
Select the server program to be activated (LIBRARY/PROGRAM) during connection. The server program is an AS/400 program started by the router or started by EASYCOMD job.

If no library is given, the library where EASYCOMD is running will be used.

### *Connection test*

These options are used only for connection test and are not saved in the configuration file.





Click on "Test" button.

If connection is successful, AS/400 version information's are displayed, as for example:

Serial number: 650643C

Model: 520

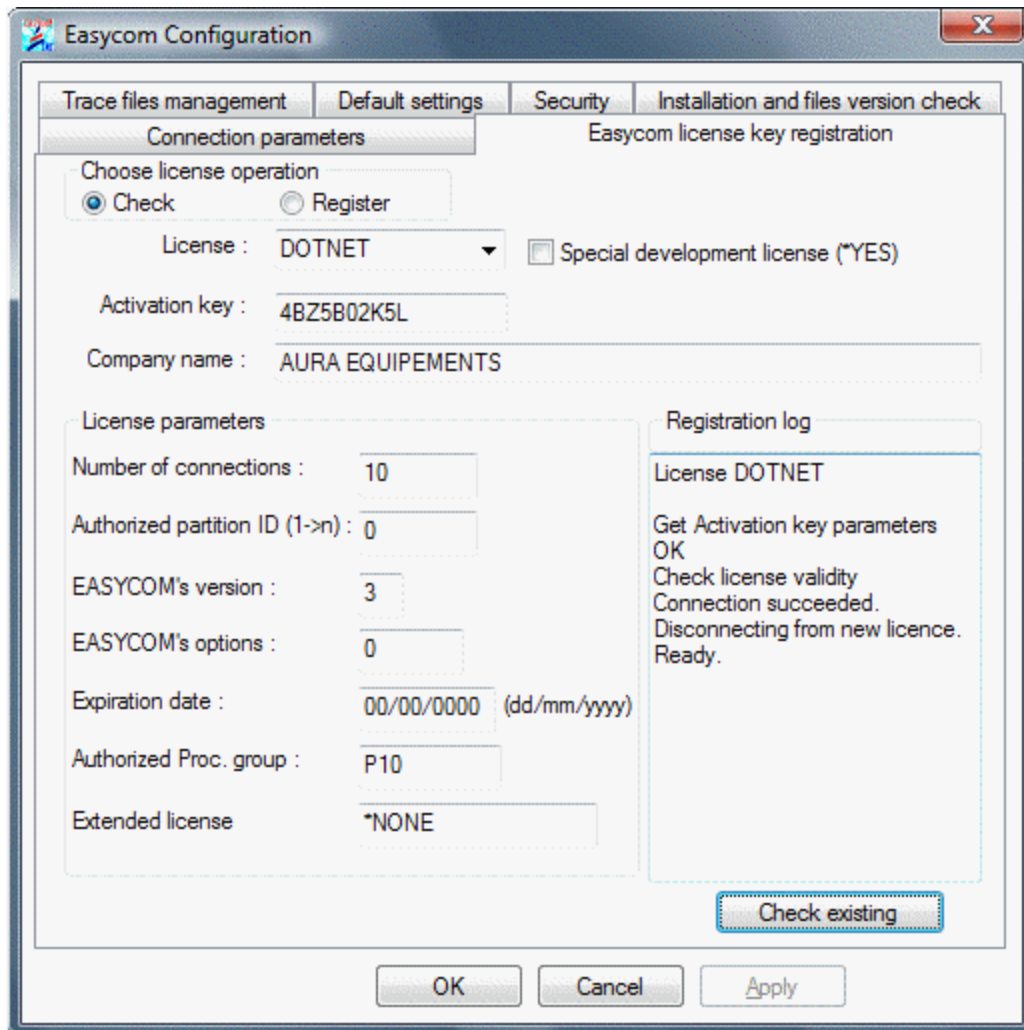
Server version: 4.60.10

If SSL connection was setup this will show if the connection was in SSL.

If Kerberos connection was selected the actual OS/400 username will be shown in the information box.

### Easycom license key registration

Activation key is provided by AURA Equipements. If this is a purchased product, the registration card will be claimed to obtain the activation key. For evaluation process, the activation key is automatically sent after having downloaded the product.



Below information must be the same as on the form received from AURA (Equivalent to EASYREG command).

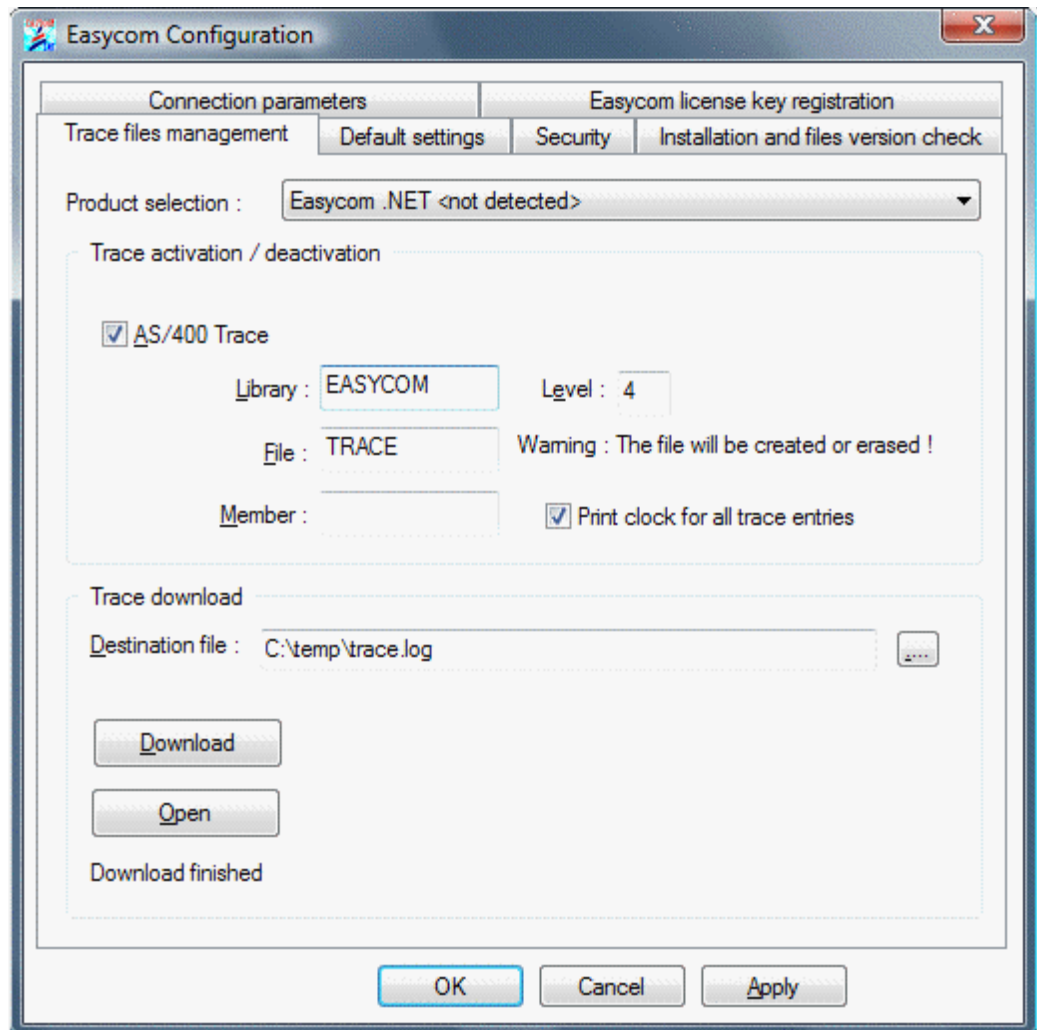
- **License** : Enter licence.
- **Special development licence** : Select this option for development licence.
- **Only used in development** : Select this option for a development licence which will never be used by an application.
- **Activation key** : Enter key (10 characters).
- **Compagny name** : Enter compagny name.
- **Number of connections** : Enter Connection(s) number.
- **Authorized partition ID** : 0 (default)
- **EASYCOM's version** : 3 (default).
- **EASYCOM's option** : 0 (default).
- **Expiration date** : Enter key the end date, in dd/mm/yyyy format
- **Authorized Proc. Group** : \* (default).
- **Extended license** : \*NONE (default).

Press '**Register**' button to submit the registration process to the iSeries. After integrity check it will store it into the iSeries and test the connection on that license.

### Trace file management

In the event of an error or to audit EASYCOM operations, it is interesting to keep traces of what program performs. AS/400 trace mode is devoted to this job.

**Traces mode reduced performances significantly, it must be strictly reserved for analysis purpose.**



*Trace Activation / Deactivation*

• **AS/400 trace :**

Select "AS/400 Trace" to activate or deactivate the EASYCOM generated trace on AS/400.

• **Library :**

Use of AS/400 trace requires to specify at least an AS/400 library and file names.

Warning, library name must be one with writing rights opened.

• **File :**

The file will be created if it does not exist and deleted later. Commands to AS/400 can detail if suitable.

• **Level :**

Low detail level is 1 (default value), highest is 9. Trace level 4 is usually sufficient. At this level, all fields' values sent or received are detailed.

• **Operation time printing :**

Operation time printing gives an idea of elapsed time between each request. Level 1 is enough in this case.

• **Member :**

Optional option.

*Trace download*

To download AS/400 generated trace, information related to AS/400 trace access is required. If AS/400 trace is already active, this information is already available.

• **Destination file :**

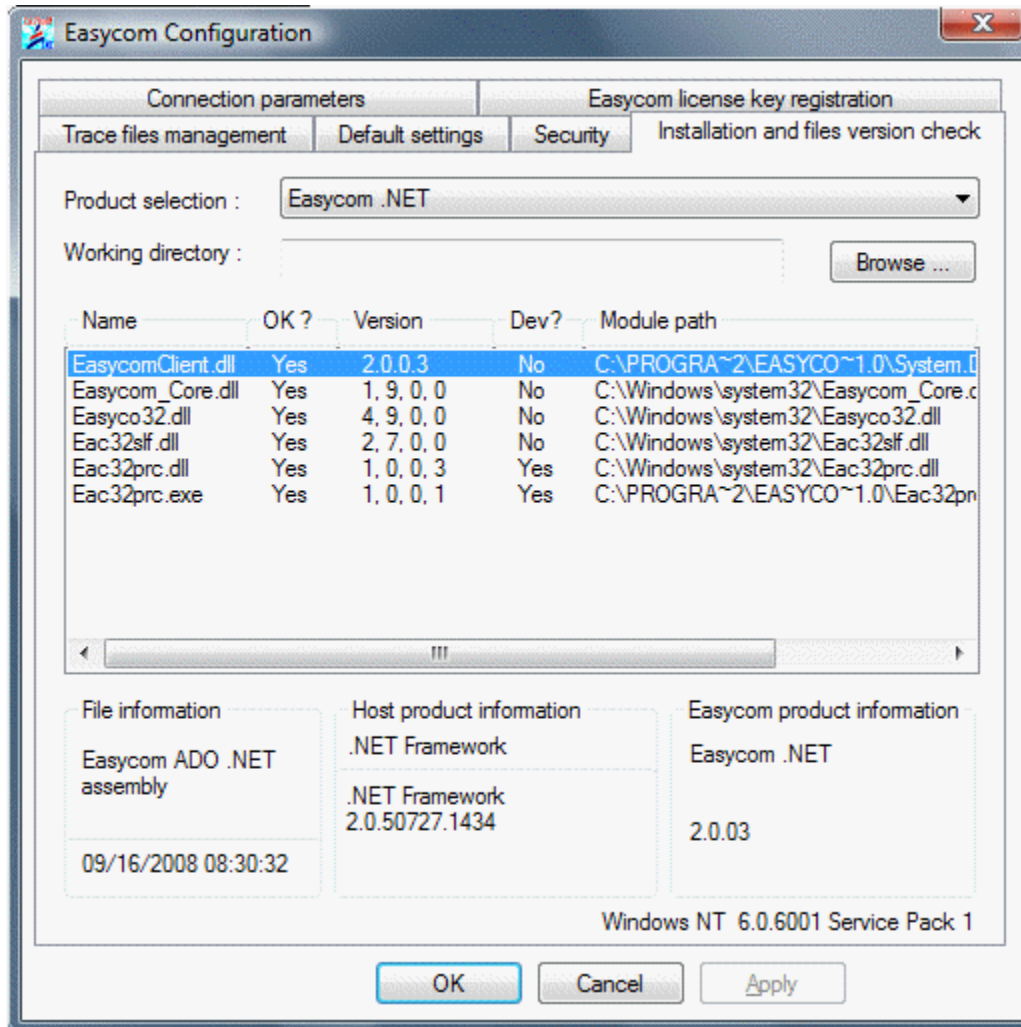
PC file should be specified, it will be generated by filling the entry box or choosing the file in the tree structure using the "browse" button.

NB: An username and password must be specified on "Connection parameters" bookmark.

• **To download :**

Click on "Downloading" to recover the trace.

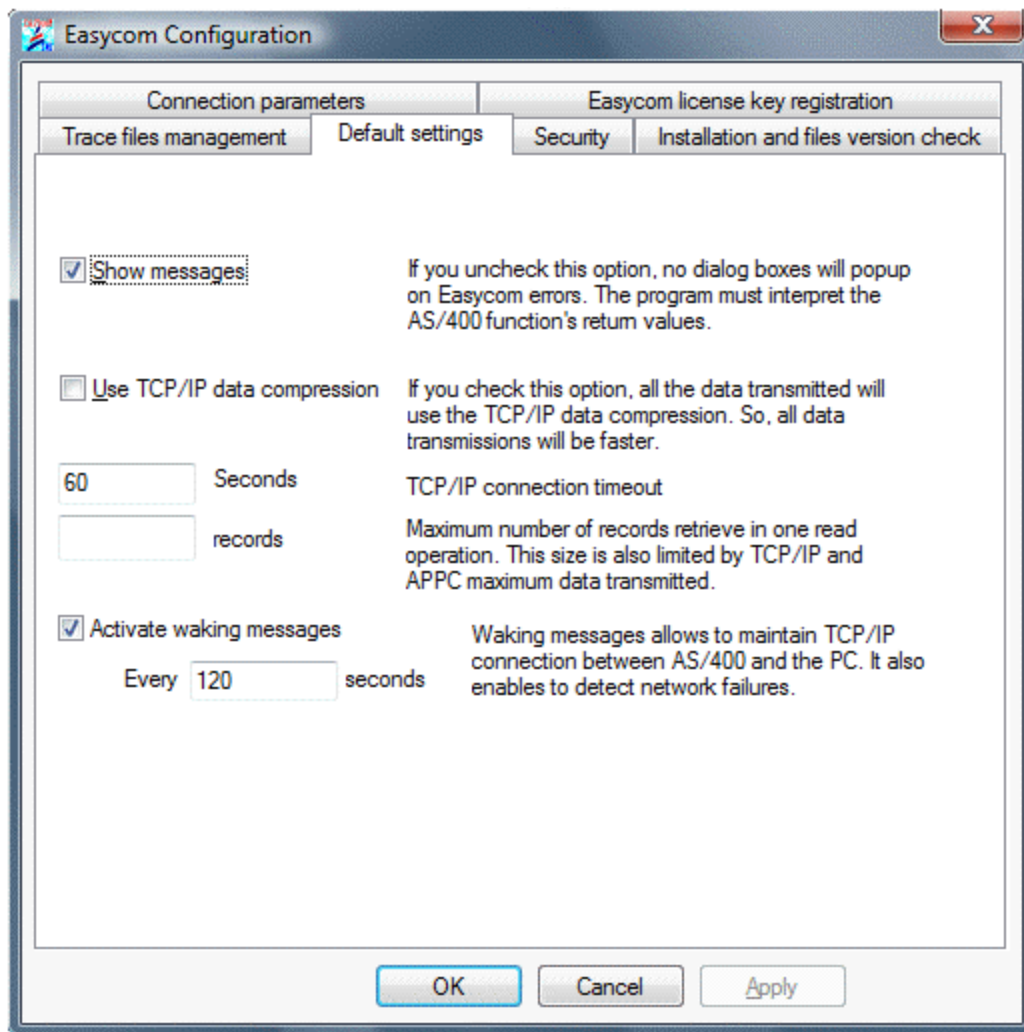
**Installation and checking modules versions (dlls)**



**Easycom default settings**

This part allows setting EASYCOM parameters to optimize network access times and reduce exchanges between AS/400 and application.





*Messages display*

If this option is unchecked, no dialog box will be displayed in case of EASYCOM error. Then, program will interpret the functions returned values in all cases (example: password error). This option is recommended with a PC program server type (Web or any program operating automatically).

*TCP/IP data compression*

This option allows to use data compression in order to reduce exchanged volumes between AS/400 and PC.

*TCP/IP connection maximum timeout*

Default: 60 seconds.

Timeout = "": Default value 60s

Timeout = 0: no timeout

*Retrieved recordings maximum number*

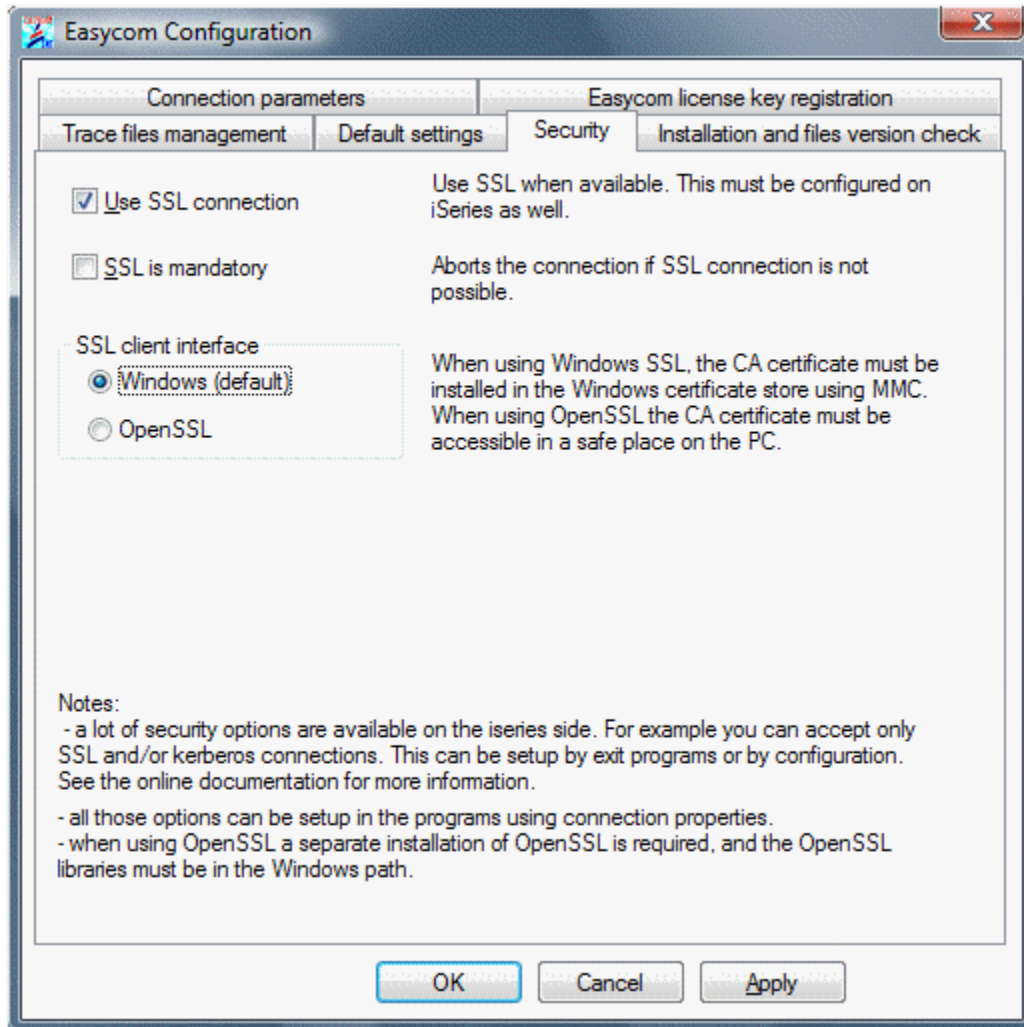
With this option maximum recordings number read in a block can be set. Default value is 32, and its limit is set by the block size parameter.

*Activating Keep Alive messages*

In the case of sustained applications uses without data exchanged, TCP/IP may close AS/400 to PC communication. To avoid that, regular intervals messages can be sent on established connection. That also allows an EASYCOM job automatic termination in case of PC prolonged silence.

**Easycom security**

This tab is for SSL default connectivity options. SSL connection settings can also be setup inside the client application.



If SSL is activated, SSL connection will be attempted. If the SSL negotiation fails or not supported by the server or client, the connection will continue without SSL (not encrypted).

If SSL is activated and mandatory, the client will successfully connect only if SSL negotiation succeeds.

Notes :

- if the client part is not up to date, the option may be ignored, and connection succeed without SSL
- if client part is up to date, but not server part, the connection will be aborted.
- The connection test tab shows if the connection test was successfully using SSL (yes, no or N/A for not supported on client). If this test succeeds, this does not mean that the application will use SSL, because client part is specific for each product (Delphi, WinDev, PHP ...).

This configuration screen shows two different interfaces:

- Windows (default). Use the Microsoft Windows integrated interface. You may be needed to install the certificate of the CA (certificate Authority) that issued the certificate of the SSL Easycom server ([see SSL connection - server configuration](#)).

To do this, use mmc (Microsoft Management console), and add the certificate store plugin into it. You can do this by clicking "start", "Run", and type "certmgr.msc" then enter. Then right-click on "Trusted Root Certification Authorities", then select "All Tasks", and the "Import". You need then to select the file that is containing the certificate.

- OpenSSL. Use OpenSSL interface. In this case the OpenSSL libraries must be available on the PC. You also need to have the CA certificate available. You can give the certificate path or name using Easycom configuration tool (or inside the application).

## Easycom.ini

The easycom.ini file contains parameters and comprehensive options (installation, optimization, trace, etc...) set including EASYCOM Configuration utility chosen parameters.

Several easycom.ini are possible. In this case it will be looked for first in the application repertory, then in the Windows repertory and finally in another path.

*Example : Easycom.ini file*

```
[INSTALL]
PCdir=C:\PROGRAM FILES\Easycom

[GENERAL]
Network=TCP
Msg=1 //Option 'Display messages'
NoWait=1
QryOptimize=a
Location=194.206.165.100 //AS/400 name or IP address

[TCP]
COMPRESSION=0
Timeout=5 //TCP/IP connection maximum time

[Buffers]
Record=9 //recordings maximum Number
//retrieved in a reading operation
Size=8000 //data maximum size in byte
//sent between AS/400 and PC
TimeOut=30 //data refreshing time

[WINDEV]
INITLIBL=PROD2005;STATS2005
```

## Annexes

### Copyright

The information contained in this document can be modified without prior notice and does not engage AURA Equipements. The Software described in this document is governed by a licensing or agreement of confidentiality. The software cannot be used, copied or reproduced on any support according to the terms of this licence or this agreement of confidentiality. No part of this handbook can be reproduced or transmitted by any maner, electronic or mechanical, including by photocopy or recording, without the express and written permission of AURA Equipements.



© 1986 - 2022 AURA Equipements. All rights reserved.

IBM, PC/AT, AS/400, iSeries, System i, i5/OS, power 5, PASE, AIX are trademarks of International Business Machines Corporation.

Windows, Word, Excel, Office 400 are trademarks of Microsoft Corporation.

EASYCOM and LAUNCHER 400 are trademarks of Aura Equipements..

All the quoted marks are trademarks by their authors.

## License contract

### CONTRAT DE LICENCE D'UTILISATION LOGICIEL " EASYCOM FOR WINDEV "

ATTENTION : Vous devez lire attentivement ce document avant d'utiliser le LOGICIEL dont vous venez d'acquérir un droit d'utilisation. En ouvrant l'emballage scellé contenant le CD Rom, vous vous engagez à respecter les termes de cette Licence. Si vous ou votre CLIENT, n'êtes pas d'accord avec les termes de cette LICENCE, vous devez retourner sous 24 Heures ce LOGICIEL complet à l'adresse où vous l'avez acheté pour obtenir son remboursement, SANS OUVRIR CETTE ENVELOPPE.

#### PROPRIETE DU LOGICIEL

1. Le LOGICIEL Easycom For WinDev (" LOGICIEL "), ainsi que la documentation produit (" Documentation ") sont et restent la propriété de AURA Equipements bénéficiant de la protection en matières de droit d'auteur et de droit des marques commerciales prévues par la loi française et les lois internationales en pareille matière.

#### DEFINITIONS

2. Le CLIENT: Le CLIENT représente la personne (physique ou morale ayant acquis la Licence ou utilisant une Version d'Evaluation).

3. Version d'évaluation: Vous utilisez une Version d'évaluation du LOGICIEL lorsque vous avez installé le LOGICIEL sans l'activation du numéro de série (clé d'activation). Après vingt minutes d'utilisation, un message s'affiche pour vous indiquer que vous utilisez une Version d'Evaluation. Il apparaîtra lors de l'utilisation du LOGICIEL toutes les minutes après une heure d'utilisation, et ceci dans la limite de 100 connexions pour une période de 15 jours après la date d'installation.

4. LOGICIEL: désigne le droit d'utilisation du LOGICIEL Easycom For WinDev acquis par le CLIENT.

(a) Installation: Le LOGICIEL Easycom For WinDev est composé d'une partie serveur qui s'installe sur une unité centrale AS/400 (Easycom For WinDev Serveur), et d'une partie cliente (Easycom For WinDev Client) qui s'installe sur les stations PC en liaison avec le Serveur.

(b) Le LOGICIEL serveur ne peut être installé qu'une seule fois sur une seule et même unité centrale AS/400 (N° de série de l'unité Centrale AS/400 unique). Si le LOGICIEL doit accéder à plusieurs sites différents, une licence du LOGICIEL doit être acquise pour chacun des sites.

(c) L'utilisation du LOGICIEL Client (licence de développement ou de déploiement) est fonction du nombre de sessions simultanées pour lequel vous avez acquis un droit de licence (la clé d'activation du LOGICIEL comprend le nombre de sessions AS/400).

(d) La Version d'Evaluation comprend un LOGICIEL Serveur et un LOGICIEL Client - 2 sessions.

4.1 Le LOGICIEL Easycom For WinDev de " DEVELOPPEMENT " permet le développement d'application. Le LOGICIEL comprend l'ensemble des fonctionnalités pour permettre le développement d'une application.

4.2 Le LOGICIEL Easycom For WinDev de " DEPLOIEMENT " permet l'exécution des développements effectués avec Easycom pour WinDev de " DEVELOPPEMENT ". Le LOGICIEL ne comprend pas l'ensemble des instructions nécessaires au développement.

4.3 La Version d'Evaluation est une version du LOGICIEL Easycom For WinDev de " DEVELOPPEMENT ".

5. Maintenance Mise à jour: comprend des corrections d'erreurs et ajout de fonctionnalités mineures ou fiabilisation de fonctions du LOGICIEL. Elle est identifiée par une modification du numéro à droite du point décimal (ex: identification de la nouvelle version du LOGICIEL " 10.1 " sera une Mise à jour de la Version 10 ou 10.0 du LOGICIEL).

6. Nouvelle Version: comprend des fonctions ou modules supplémentaires. Une nouvelle version est identifiée par une modification du numéro à gauche du point décimal (ex: une modification de " 10 " en " 11 ". AURA EQUIPEMENTS est le seul détenteur du droit de décider de la disponibilité des nouvelles Versions et/ou Mises à jour.

#### TERMES DE LA LICENCE



7. Licence pour Version d'évaluation: La version d'évaluation dans les limites et indications portées au paragraphe 3 est librement utilisable dans le seul but des tests du LOGICIEL en vue de l'acquisition d'une licence d'utilisation. La reproduction et distribution de la version d'évaluation (à l'exception expresse de la documentation) sont autorisées à condition de ne pas modifier les termes de cette licence, copyright, trademark, documentation et tout ou partie du LOGICIEL.

En aucun cas, le CLIENT ne pourra rechercher AURA EQUIPEMENTS en responsabilités pour des pertes ou dommages subis lors de la diffusion de la Version d'Evaluation.

8. Acquisition de la Licence d'utilisation: Si vous avez une Version d'Evaluation du LOGICIEL et que vous souhaitez bénéficier des droits d'utilisation du LOGICIEL décrits dans les sections 9,22 et 23, vous devez obtenir une clé d'activation, qui désinstalle les messages décrits dans la section 3. Vous devez acquérir une LICENCE d'utilisation pour obtenir une clé d'activation.

9. Licence: En vue de préserver et de protéger ses droits dans le cadre de la législation applicable, AURA EQUIPEMENTS ne vend pas de droit sur le LOGICIEL, mais octroie le droit d'exploiter ce LOGICIEL dans le cadre d'un accord de LICENCE et conserve expressément le droit de propriété de tous les LOGICIELS. Cette LICENCE d'utilisation non exclusive est concédée au CLIENT contre versement d'une redevance dépendant (i) du nombre de postes clients connectés à un seul serveur, et (ii) du critère de qualification du LOGICIEL soit " DEVELOPPEMENT " et/ou " DEPLOIEMENT ".

## UTILISATION

10. Le CLIENT ne peut copier le LOGICIEL ou la documentation, excepté (i) comme autorisé dans la section 7 et (ii) pour faire une copie de sauvegarde à des fins de sauvegarde ou d'archive.

11. Il est interdit de modifier ou tenter de modifier ce LOGICIEL, de le traduire ou tenter de le traduire, de décompiler ou tenter de décompiler, de créer ou tenter de créer des travaux dérivatifs qui s'en inspireraient, de désassembler ou tenter de désassembler ce LOGICIEL.

12. Le CLIENT ne pourra, en aucun cas, louer, prêter, vendre, donner, divulguer ou mettre à disposition d'un tiers le LOGICIEL. Par exception, il pourra dans le cadre de l'article 7 distribuer gratuitement la version d'évaluation.

13. Le CLIENT ne pourra diffuser par le réseau Internet tout ou partie de la documentation ou du LOGICIEL excepté comme autorisé dans la section 7. Il est interdit de modifier ou tenter de modifier les clés d'activation (i) du LOGICIEL acquis par Licence d'utilisation (ii) de la Version d'Evaluation.

14. Le CLIENT ne bénéficie pas des droits de reproduction ou de distribution de tout ou partie du code source du LOGICIEL. La duplication de la documentation, en totalité ou en partie est formellement interdite.

15. Le CLIENT gardera confidentiel son code d'accès au LOGICIEL. Il s'engage à n'autoriser l'accès au LOGICIEL qu'à ses seuls salariés pour lesquels un tel accès est nécessaire.

16. Il est interdit d'enlever ou de tenter d'enlever les mentions de copyright pouvant apparaître et/ou étant contenues dans le LOGICIEL.

## GARANTIE LIMITEE

17. Le LOGICIEL et la documentation qui l'accompagne sont vendus en l'état, sans aucune garantie d'aucune sorte.

18. AURA EQUIPEMENTS ne fournit aucune garantie expresse ou implicite, de quelque nature que ce soit sur le LOGICIEL et les services fournis. Notamment, le CLIENT ne saurait invoquer l'inaptitude du LOGICIEL et des services rendus à atteindre les objectifs qu'il se serait fixés.

19. Si le support physique LOGICIEL venait à être défectueux, le CLIENT se procurera auprès de AURA EQUIPEMENTS un nouveau support dans la limite d'une période de 30 jours à compter de la date d'acquisition.

20. Cette garantie ne sera pas appliquée si la détérioration provient d'un accident ou d'une mauvaise utilisation.

21. En aucun cas AURA EQUIPEMENTS ne pourra être tenu responsable de tout dommage direct, indirect, secondaire ou accessoires (y compris les dommages entraînés par la perte de bénéfices, l'interruption des activités ou la perte d'informations et autres) découlant de l'utilisation ou de l'impossibilité d'utilisation du produit, et ce même si AURA EQUIPEMENTS a été informé de la possibilité de dommages. La présente garantie limitée est régie par les lois en vigueur en France au bénéfice de AURA EQUIPEMENTS.

## SERVICES DE MAINTENANCE

22. Support Technique: Vous devez compléter et retourner la carte d'enregistrement auprès de AURA Equipements. L'assistance n'est effective qu'après retour de ce Bon de Garantie chez AURA Equipements. Si vous avez acquis une licence d'utilisation du LOGICIEL VERSION DEVELOPPEMENT, vous avez droit pendant 30 jours à partir de la date de livraison de la clé d'activation du LOGICIEL à l'accès au support technique de AURA Equipements : contacts avec le service du support technique de AURA Equipements par fax, e-mail, forum Internet. Ce support consiste à répondre aux questions générales d'installation du LOGICIEL. Après cette période le support technique concernant ce LOGICIEL est assuré uniquement par l'intermédiaire du service payant " ASSISTANCE PREMIUM ".

23. Mise à jour et Version: AURA EQUIPEMENTS met à la disposition du CLIENT les mises à jour du LOGICIEL et les nouvelles versions de ce dernier correspondantes uniquement aux évolutions technique de logiciel EASYCOM, dans le cadre d'un contrat de " Mise à Jour " annuel souscrit auprès de AURA EQUIPEMENTS, au tarif en vigueur.

SONT EXCLUS DE CE SERVICE LES CHANGEMENTS DE VERSION INDUITS PAR UNE NOUVELLE VERSION DU LOGICIEL WINDEV LUI-MEME.

DIVERS

24. Validité : La licence d'utilisation du LOGICIEL démarre à la date de livraison de la Clé d'activation et continuera tant que le LOGICIEL sera protégé par copyright. S'il s'avère que le CLIENT ne respecte pas pour quelles raisons que ce soient les clauses du contrat, il devra immédiatement détruire l'ensemble du LOGICIEL et ses copies ainsi que la documentation et en apporter la preuve à AURA EQUIPEMENTS.

25. Juridiction : Ce contrat de LICENCE est régi par le droit français, tous litiges relatifs à la validité, à l'interprétation, l'exécution ou inexécution du présent contrat seront de la compétence exclusive des tribunaux du siège social de AURA EQUIPEMENTS.

AURA EQUIPEMENTS

129 Rue de l'Abbé Groult, 75015 PARIS, France

Tel: 33 (0)1 53 76 86 35

Web: [www.easycom-aura.com](http://www.easycom-aura.com)

### Technical support

For support requests you must create a ticket via the support website: <http://support.easycom-aura.com>. You must register the first time to receive a password.

### Assistance contract

AURA Equipements offers you several levels of technical support.

Contact us, we will send you the best commercial offer to meet your needs.

For general or commercial information: [info@easycom-aura.com](mailto:info@easycom-aura.com).

### List of AS400 objects

You will find below the complete list of AS400 objects :

*ALRTBL
*CSPMAP
*AUTL
*CSPTBL
*BNDDIR
*CTLD
*CFGL
*DEV D
*CHTFMT
*DOC
*CLD
*DTAARA
*CLS
*DTADCT
*CMD
*DTAQ
*CNNL
*EDTD



*COSD
*EXITRG
*CRG
*FCT
*CRQD
*FILE
*CSI
*FLR
*FNTRSC
*JRN
*FNTTBL
*JRNRCV
*FORMDF
*LIB
*FTR
*LIND
*GSS
*LOCALE
*IGDCT
*MEDDFN
*IGSRT
*MENU
*IGCTBL
*MGTCOL
*IMGCLG
*MODD
*IPXD
*MODULE
*JOB
*MSGF
*JOBQ
*MSGQ
*JOBSCD
*M36
*M36CFG
*PNLGRP
*NODL
*PRDAVL
*NTBD

*PRDDFN
*NWID
*PRDLOD
*NWSCFG
*PSFCFG
*NWSO
*QMFORM
*OUTQ
*QMQRV
*OVL
*QRYDFN
*PAGDFN
*SBSO
*PAGSEG
*SCHIDX
*PDFMAP
*SPADCT
*PDG
*SQLPKG
*PGM
*SQLUDT
*SQLXSR
*SRVPGM
*SSND
*S36
*TBL
*TIMZON
*USRIDX
*USRPRF
*USRQ
*USRSPC
*VLDL
*WSCST