OPTIMAL SYSTEMS

Software Documentation
enaio® server-api

Version 8.50

optimal-systems.de

Software für Macher.

11.04.2017

Version 8.50

Contents

Introduction

enaio® is a powerful document management, workflow, and archiving system. A special feature of the product line is that it is flexibly configurable and that, thanks to many interfaces, it allows to integrate other systems at different points.

The functions provided by enaio® Server API will be described in detail in this document. The document serves as a functional reference model. Further information about the structure of enaio® and the various components can be found in the corresponding documentation.

The enaio® server is a runtime environment for various engines in the archive, DMS, and workflow environments. Thus, the server fullfils different tasks concerning organizational information flow. These tasks include for example capturing, administrating and processing documents including their index data, full text search, workflow management, archiving and many other functions.

Engine

An engine – also called executor – is loaded by the application server and can therefore execute jobs. Every executor manages one or more namespaces in which server jobs are organized. The following engines are provided by default:

§ DMS Engine: for search and manipulation of index and document data

§ Workflow engine: Processing and managing workflow processes and models

§ Standard engine: collection of various functions for archiving, and file and document transfer

§ Full text engine: full text engine queries (Microsoft SQL full text server, Convera RetrievalWare)

§ OCR engine: optical character recognition that converts images or scanned receipts

§ Subscription engine: notification functions for changes to the document inventory

§ Core services: initialization, licensing, and session management

§ Data piping allows accessing the database through the server interface with an internal format or as an Active Directory object

Server Job

Jobs are tasks which are executed by the server and which represent a specific collection, manipulation, or control function. Therefore, jobs can be compared to features. A server job has the following general structure:

§ Job name: the name consists of a namespace – in which the job is implemented – and the job description. (e.g. dms.XMLInsert)

§ Input parameters

§ Input file list: if files have to be transferred to the job, the absolute path and name of the file are transferred here.

§ Output parameters

§ Output file list: if the job returns files, the absolute path and name of the file are transferred here.

§ Return value: Every job generates a return value. In case of success it is always '0'. Otherwise a return value describing the nature of the possible error will be returned.

To execute a job, a session must be created between the querying instance - also called client - and the server. It consists of a connection via a technical protocol (e.g. TCP) and different information about this connection, e.g. authentication, station, and license data. The XML-RPC protocol was placed on top of the technical protocol for building search requests and processing results. This protocol dictates how search requests, parameters and results are presented. The communication process between client and server follows the following schema:

§ Establishing a TCP connection with the enaio® server

§ Initializing a session with respective parameters

§ Job calls and receipt of response data

§ Termination of the connection

The client executes a job by wrapping the job name and the respective parameters in XML and sending them to the server. The client will then wait for a response. The server kernel interprets the received data. It is determined which engine can execute the job and data is transferred to the job's queue mechanism for processing. The result is then transferred by the engine to the client through the server kernel. Jobs which are provided by the engines are described below.

The XML-RPC format is designed to transfer all parameters within an XML structure with the respective types. For file transfer (i.e. binary data) it is therefore necessary to convert the files with MIME encoding into strings. This deviation from the standard is due to performance and memory usage. Files are transferred separately as TCP streams after sending the job parameters. Some parameters, XML structures in particular, are converted into the MIME format for transfer, to remove control characters, if necessary.

Interface Libraries

To make the internal communication independent from technical conditions, several ways can be used to use the protocol without extensive knowledge of the communication sequence. For this purpose classes and libraries exist which are provided by OPTIMAL SYSTEMS.

These are:

§ COM – interface of the communication library oxsvrspt for multi-thread environments

§ Java interface for communication with the enaio® server

§ COM – interface of the communication library oxmljsc

§ Other libraries which encapsule server calls

The structure for using the interfaces is principally the same. After initializing and connecting to the application server, job objects are created which receive input parameters and input files. After the execution, output parameters and output files can be read from the object. Furthermore an error stack is provided which can contain possible technical error messages. Input and output lists are depicted as hash lists, arrays or other objects, depending on the used programming language.

Primarily the oxsvrspt.dll library should be used. Detailed information about using libraries can be found in the section 'OxSvrSpt.'

Java Interface

The Java interface libraries provided by OPTIMAL SYSTEMS support the creation of client applications that communicate with the enaio® server. In total there are three libraries:

§ Application server proxy

§ Java DRT Layer (JDL)

§ Java Object Layer

The three interface libraries can be used in different application areas and represent different abstraction layers regarding the OS system. While the application server proxy and the JDL are two dependent layers in the communication with the enaio® server, the Java Object Layer permits an object-oriented way of working with input and output data of the jobs. Complete documentation of the Java interface can be obtained separately.

Realization of Archive Integration

General

The goal of archive integration for various applications is to file documents, which contain structure and index information, in the DMS, to search them in the data record, to download and to delete them. These scenarios will be illustrated in the following examples. As already mentioned, functions are carried out by calling server jobs which have been implemented in the DMS engine and in the standard DMS engine.

A brief introduction to the structure of the system. Further information can be found in the enaio® administration manuals. enaio® uses the following DMS objects to depict information structures:

§ **Cabinet:**

The term 'cabinet' was chosen as a counterpart to a file cabinet. A cabinet contains folders, registers and documents. It is the top level in the DMS. All other DMS objects can only exist as sub-objects of a cabinet. A cabinet's only attribute is its name.

§ **Folder:**

Folders are described by index data and are containers for underlying objects. They can be compared to folders at root level in the file system. There is only one folder type for each cabinet.

§ **Register:**

Registers are used to create a more detailed information structure. Multiple register types which differ according to the structure of the index data can coexist in each cabinet.

§ **Documents:**

Documents not only have index data but are also linked to document files. Every document type has the property of a main type which indicates whether the linked files are images, Windows source documents or other file types. The enaio® client behaves differently while capturing and displaying documents depending on the main type.

The classes of folders, registers and documents are called object types. A customer folder or an invoice type are for example an object type. All instances of a class, i.e. the individual folders or receipts have the same indexing fields. For documents the object type also determines the main type. The structure of the index data is dictated by the object model. For each object type, a number of fields can be defined which are used for indexing and searching.

In the subsequent part, documents, registers and folders are called objects, unless a more detailed specification is necessary. Every object is indicated by index data and so-called basic parameters. Basic parameters area automatically assigned by the system and are used for internal management of objects. They contain information on the creator of an object, its object ID, modification data etc.

Every object in the DMS is uniquely described by its object type and object ID. These are two numbers which are uniquely combined. For most jobs, object type and object ID are expected as input parameters.

A search which is based on search parameters such as index data and basic parameters will determine objects for further processing. Further queries can be carried out with the retrieved object IDs and object types or the document files can be downloaded.

During an import, the called application dictates index data and document files and new objects are thereupon created in the system.

Scenarios and their Jobs

Below you can find a short description of the jobs that are necessary for archive integration.

§ **Create session:** krn.SessionAttach

§ **User login:** krn.SessionLogin

§ **Read object definition:** dms.GetObjDef

§ **Search by folders, registers, documents:** dms.GetResultList

§ **Download document:** std.StoreInCacheByID

§ **Insert objects:** dms.XMLInsert

§ **Delete object:** dms.XMLDelete

§ **Close session:** krn.SessionLogout

Test Options

Test Application axlabjobs.exe

For testing individual jobs of the enaio® server and its engines, OPTIMAL SYSTEMS provides the application `axlabjobs.exe`. Jobs can be executed with any parameter an unlimited number of times. Use a Connect string to specify to which server the TestLab will connect. It is possible to work with a great number of TestLabs on one server to run performance tests. The program is installed by default to the server directory. The reference to each job contains information about which parameters need to be specified for the test program.

You can use enaio® enterprise-manager to monitor job calls. There, you can specify computers as well as jobs which have to be monitored. Files which are sent along with jobs can also be accessed through a temporary directory.

The functions for monitoring jobs can be found in enaio® enterprise-manager in the 'Extended Administration/Monitoring' area.

Glossary

This section offers explanations for some terms used in the documentation of server jobs.

Cache directory – directory below the server path (..\server\CACHE), which is used to store archived documents read by archiving media for further access, so that the next user does not have to spend a lot of time accessing archiving media again.

Flag – is a parameter which enables or indicates a specific property.

OSTEMP directory – is a directory in the environment variable OSTEMP and is used by some jobs for interim storage or the creation of temporary files.

Work directory – is a directory below the server path (..\server\WORK), which is used to file all documents which have not been archived. Due to performance reasons, documents are filed in this directory and not in the database.

Parameters and return values which are enclosed in square brackets '[Parameter]' are optional parameters. If they are not needed, these parameters do not have to be indicated when a job is called.

enaio® Server API Engine Directory

Engine	Description	Areas
Subscription (abn)	Setup and control of subscriptions used to notify a change to DMS objects	
ADO database (ado)	Access to the database	
Convert (cnv)	Conversion and access to image files	
DMS (dms)	Searching and editing of index data, DMS objects, relations and portfolios	XML import XML export (search) Security system Relations and relation texts Portfolios User-related data Other jobs
Medicine (med)	Access to medical information	
Users/groups (mng)	Access to groups and users in enaio®	
OCR (ocr)	Optical character recognition	
Standard (std)	Work, cache, file and archive management.	Work, cache, and archive management File administration Internal jobs Other jobs
Full-text (vtx)	Processing full text queries of enaio® client	
Workflow (wfm)	Processing and managing workflow processes and models	Organizational structure Workflow model Workflow process and process step Workflow form, event, and script Administration and history administration Administration History administration Other jobs

		Server-internal jobs
Core services (following engines are implemented directly in the server core)		
Administration (adm)	System file management	
Kernel (krn)	Batch management, server monitoring, registry administration, and administration of loaded engines at runtime	Registry administration Batch administration Server administration Session administration Engine administration Other jobs
License (lic)	License management for the entire enaio® system	
Data Transfer Services (Namespace dtr)	Server-side execution of the data transfer server	

enaio® Server API Engine Documentation

Subscription Engine (Namespace abn)

Functions used to set up and control subscriptions are implemented in the subscription engine. These inform the user about a change to DMS objects.

In multi-server systems, a server can only inform clients that are connected to the server.

- abn.Add
- abn.CheckOsrevisit
- abn.GetAboGrpList
- abn.GetDocList
- abn.GetGroupList
- abn.GetRequestList
- abn.GetUserList
- abn.NotifyRequestAbo
- abn.NotifyAbonnement
- abn.Remove
- abn.RemoveAboIdent
- abn.UpdateReqAboGrp
- abn.RemoveAllObjNotifyFromUser
- abn.RemoveObjNotifyFromUser
- abn.ConfirmAboRead
- abn.RemoveObjRevisitNotifyFromUser
- abn.SetObjRevisitClosed
- abn.SetObjRevisitOpen
- abn.ChangeRevisitUser
- abn.AddRevisit
- abn.UpdateRevisit
- abn.GetSubscriptions
- abn.GetRevisits
- abn.SetOsInformed
- abn.ResetOsInformed
- abn.GetRecentObjects
- ado.ExecuteSQL

abn.Add

Description:

This job creates a subscription for the specified object.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object for which notification is intended

ObjectType (INT): Object type

ActionFlags (INT): action executed with the document, which leads to a notification

§ 2 = notification if a document was created (only for subscribed search requests)

§ 3 = notification if index data were changed (only for subscribed documents)

§ 4 = notification if the document was changed

§ 27 = notification if the document was deleted

§ 39 = notification if a location was added

Channel (INT): notification channel (type of notification)

§ 0 - notification via internal channel (oxmljsc)

§ 1 - notification via e-mail

AboGrpID (STRING): this ID relates to all actions of a subscription

[User] (STRING): Name of the user receiving a notification

[Alias] (STRING): Subscription info text (max. 255 characters)

[Product] (STRING): String describing the program instance, e.g. 'ax.exe'

[Confirm] (INT):

§ 1 = if user/group was notified, the message must be marked as read before it can be deleted;

§ 2 = before the message can be marked as read, the user's system password is queried; otherwise 0

[Station] (STRING): Name of the station exclusively intended to receive the message

[Mail] (STRING): E-mail address (max. 255 characters) for notification (multiple addresses are separated by a semicolon)

[AboType] (INT): Distinguishes between search request subscription and document subscription

§ 0 = document subscription

§ 1 = search request subscription

[RequestFormat] (STRING): request format Default is 'ABN' for the native SQL format. In this case, the search request is evaluated in the 'AboRequest' parameter. If 'XML' is specified, a DMSQuery request is expected in the 'XmlRequest' parameter.

[AboRequest] (STRING): SQL statement for search request subscriptions

The request must consist of lower-case letters and begin with 'select count (distinct d.id)' or with 'select distinct d.id', otherwise it will be rejected.

Depending on the main type of the requested object, the identifier 'o.id' must be used for folders, 'r.id' for registers or 'd.id' for documents.

[XmlRequest] (STRING/Base64): Abo request in DMSQuery XML format

[GroupID] (STRING): GUID of the group intended for notification

[UserID] (STRING): User GUID of the subscription administrator if system subscriptions were created

Return:

(INT): 0 = job successful, otherwise error code

See also:

abn.Remove

abn.CheckOsrevisit

Description:

This job checks the follow-ups (table 'osrevisit') and sends a notification to users for which follow-ups are set up if this is required by the respective time stamps. For notifying the concerned users, this job uses the job 'abn.Osrevisit'. This job is periodically called internally. Therefore a batch must be set up in the registry.

Parameter:

Flags (INT): not currently supported-> transfer 0

Return:

(INT): 0 = job successful, otherwise error code

See also:

Batch administration

abn.GetAboGrpList

Description:

This job provides information on the subscription via the AboGrp-ID.

Parameter:

Flags (INT): not currently supported-> transfer 0

AboGrpID (STRING): this ID relates to all defined actions of the subscription

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Abo[1..n] (STRING): information on the subscription separated by semicolon

§ ID of the document

§ Type of the document

§ Info Text

§ Subscription ID

§ Confirm: 1 = if user/group were notified, the message must be marked as read before it can be deleted; otherwise 0

§ UserID: User GUID of the subscription administrator if system subscriptions were created

- GroupID: ID of the group intended for notification
- Subscription type: 0 = document subscription; 1 = search request subscription
- Action which causes a notification
  - 2 = notification if a document was created (only for subscribed search requests)
  - 3 = notification if index data were changed (only for subscribed documents)
  - 4 = notification if the document was changed
  - 27 = notification if the document was deleted
  - 39 = notification if a location was added
- Notification type
  - 0 = notification via internal channel (oxmljsc)
  - 1 = notification via e-mail; e-mail address
- Name of the user receiving a notification
- SQL statement for the search request subscription

abn.GetDocList

Description:

This job returns all objects the specified user has subscribed to.

Parameter:

Flags (INT): controls the extent of the output

- 0 = documents for the specified user name are determined
- 1 = in addition, documents are determined that were created for user groups of which the user is currently part

User (STRING): User name

[Product] (STRING): Search is restricted for the program instance

[Station] (STRING): Search is restricted for the name of the user station

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Document[1..n] (STRING): Information on the subscription separated by semicolon

- Document ID
- Type of the document
- Info Text
- Subscription ID
- Confirm: 1 = if user/group were notified, the message must be marked as read before it can be deleted; otherwise 0
- UserID: User GUID of the subscription administrator if system subscriptions were created
- GroupID: ID of the group intended for notification

- Subscription type: 0 = document subscription; 1 = search request subscription
- Action which causes a notification
  - 2 = notification if a document was created (only for subscribed search requests)
  - 3 = notification if index data were changed (only for subscribed documents)
  - 4 = notification if the document was changed
  - 27 = notification if the document was deleted
  - 39 = notification if a location was added
- Notification type
  - 0 = notification via internal channel (oxmljsc)
  - 1 = notification via e-mail; e-mail address
- AboGrp: combines all data records of the different actions to a subscription

abn.GetUnreadAboCount

Description:

This job calculates the number of read and unread subscription notifications of a user.

Parameter:

Flags (INT): not currently supported-> transfer 0

UserID (INT): User ID of the user whose subscription notifications are to be calculated.

Mode (INT): Specifies which subscription notifications are to be calculated (optional):

- 0: All subscription notifications are calculated (default).
- 1: Notifications from subscriptions which the user created him or herself will not be taken into account.
- 2: For every subscription, only the last subscription notification will be determined in each case.
- 3: Notifications from subscriptions which the user created him or herself will not be taken into account and for each subscription, only the last subscription notification will be calculated (combination of mode 1 and mode 2).

Return:

Read (INT): Number of read subscription notifications

Unread (INT): Number of unread subscription notifications

abn.GetGroupList

Description:

This job determines the user groups which have subscribed to the specified object.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object to be checked

ObjectType (INT): Object type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Group[1..n] (STRING): information on group and subscription separated by semicolon

- Group ID
- Group name
- Group GUID
- Group description
- Subscription info text
- User GUID of the subscription administrator if system subscriptions were created
- AboGrp: combines all data records of the different actions to a subscription
- Action which causes a notification
  - 2 = notification if a document was created (only for subscribed search requests)
  - 3 = notification if index data were changed (only for subscribed documents)
  - 4 = notification if the document was changed
  - 27 = notification if the document was deleted
  - 39 = notification if a location was added
- Notification type
  - 0 = notification via internal channel (oxmljsc)
  - 1 = notification via e-mail; e-mail address
- Subscription ID

abn.GetRequestList

Description:

This job returns all search request subscriptions for the specified user.

Parameter:

Flags (INT): not currently supported-> transfer 0

User (STRING): User name

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Request[1..n] (STRING): information on search request separated by semicolon

- Subscription info text
- Object type
- Subscription ID
- 1 = request confirmation, otherwise 0
- User GUID of the subscription administrator if system subscriptions were created
- GUID of the group intended for notification

- Subscription type: 0 = document subscription; 1 = search request subscription
- Action which causes a notification
  - 2 = notification if a document was created (only for subscribed search requests)
  - 3 = notification if index data were changed (only for subscribed documents)
  - 4 = notification if the document was changed
  - 27 = notification if the document was deleted
  - 39 = notification if a location was added
- Notification type
  - 0 = notification via internal channel (oxmljsc)
  - 1 = notification via e-mail; e-mail address
- AboGrp: combines all data records of the different actions to a subscription
- SQL string for the search request subscription

abn.GetUserList

Description:

The job returns a list of all subscription owners (search requests/documents) for the specified object.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the subscribed document or 0 = subscribed search requests for which a user list will be created

ObjectType (INT): Object type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

User[1..n] (STRING): list of all received subscription users

- Name of the user receiving a notification
- Subscription info text
- User GUID of the subscription administrator if system subscriptions were created
- AboGrp: combines all data records of the different actions to a subscription
- Action which causes a notification
  - 2 = notification if a document was created (only for subscribed search requests)
  - 3 = notification if index data were changed (only for subscribed documents)
  - 4 = notification if the document was changed
  - 27 = notification if the document was deleted
  - 39 = notification if a location was added
- Notification type
  - 0 = notification via internal channel (oxmljsc)

  - 1 = notification via e-mail; e-mail address
- Subscription ID

abn.NotifyRequestAbo

Description:

This job is executed by the server kernel and checks if users need to be notified of new subscriptions. To have the kernel execute this job, a batch called 'RegAbo' must be set up.

Parameter:

Flags (INT): not currently supported-> transfer 0

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.Batch administration

abn.NotifyAbonnement

Description:

This job sends a notification to a user according to the specified parameters. If action = 1 (the document has been deleted) the notification and the subscription for this document are deleted from the database.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the document that was modified or deleted

ObjectType (INT): Object type

Action (INT): 1, if the document was deleted; 0, if the document was modified

User (STRING): name of the user who has modified or deleted the document

UserID (INT): ID of the user who has modified or deleted the document

Return:

(INT): 0 = job successful, otherwise error code

abn.Remove

Description:

This job deletes subscriptions for objects from the database. These parameters can be used in different combinations. If e.g. the parameter 'Station' is used, only the entries for this station are deleted.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the document for which you want to remove the subscription

ObjectType (INT): Object type

[User] (STRING): Name of the user receiving a notification

[UserID] (STRING): User GUID of the subscription administrator if system subscriptions were created

[GroupID] (STRING): ID of the group for which the subscription was defined

[Product] (STRING): to refine the search. String which describes the program instance of the subscription which is supposed to be removed.

[Station] (STRING): to refine the search. String which describes the user station of the subscription which is supposed to be removed.

[Alias] (STRING): Document alias whose subscription is to be removed (max. 255 characters)

Return:

(INT): 0 = job successful, otherwise error code

abn.RemoveAboIdent

Description:

This job deletes a subscription entry from the table 'osabonnement' according to the Abo-ID or the AboGrp-ID.

Parameter:

Flags (INT): not currently supported-> transfer 0

[AboID] (INT): Subscription ID

[AboGrpID] (STRING): ID of the subscription group

Return:

(INT): 0 = job successful, otherwise error code

abn.UpdateReqAboGrp

Description:

This job changes the SQL statement and the object type of search request subscriptions for the subscription group.

Parameter:

Flags (INT): not currently supported-> transfer 0

AboGrpID (STRING): Subscription group

AboRequest (STRING): SQL statement for search request subscriptions

The request must consist of lower-case letters and begin with 'select count (distinct d.id)' or with 'select distinct d.id', otherwise it will be rejected.

Depending on the main type of the requested object, the identifier 'o.id' must be used for folders, 'r.id' for registers or 'd.id' for documents.

ObjectType (INT): Object type

Return:

(INT): 0 = job successful, otherwise error code

abn.RemoveAllObjAboNotifyFromUser

Description:

This job removes all subscription notifications assigned to the user.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object

ObjectType (INT): Object type

UserID (INT): User ID

Return:

(INT): 0 = job successful, otherwise error code

abn.RemoveObjAboNotifyFromUser

Description:

This job removes a specific subscription notification assigned to the user.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object

ObjectType (INT): Object type

UserID (INT): User ID

AboSetTime (INT): Notification time

SetUserID (INT): ID of the user who triggered the notification

Return:

(INT): 0 = job successful, otherwise error code

abn.ConfirmAboRead

Description:

This job sets the subscription notification for which a confirmation was requested to 'confirmed'.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object

ObjectType (INT): Object type

UserID (INT): ID of the user who is intended to receive the notification

AboSetTime (INT): time when the user received notification

Return:

(INT): 0 = job successful, otherwise error code

abn.RemoveObjRevisitNotifyFromUser

Description:

This job removes a specific follow-up notification assigned to the user.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object

ObjectType (INT): Object type

UserID (INT): User ID

RevisitTime (INT): Notification time

Return:

(INT): 0 = job successful, otherwise error code

abn.SetObjRevisitClosed

Description:

This job sets a specific follow-up notification assigned to the user to 'edited'.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object

ObjectType (INT): Object type

UserID (INT): User ID

RevisitTime (INT): time when the notification was created

Return:

(INT): 0 = job successful, otherwise error code

abn.SetObjRevisitOpen

Description:

This job removes the status 'edited' for a specific follow-up notification assigned to the user.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object

ObjectType (INT): Object type

UserID (INT): User ID

RevisitTime (INT): time when the notification was created

Return:

(INT): 0 = job successful, otherwise error code

abn.ChangeRevisitUser

Description:

This job assigns all follow-up notifications of a user to another user.

Parameter:

Flags (INT): not currently supported-> transfer 0

NewUserID (INT): ID of the new user

OldUserID (INT): ID of the user to date

Return:

(INT): 0 = job successful, otherwise error code

abn.AddRevisit

Description:

This job sets an object for a specific user to 'follow-up'.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object

ObjectType (INT): Object type

UserID (INT): ID of the user who is intended to receive the notification

RevisitTime (INT): time when the user is expected to receive the notification

InfoText (string): info text(max. 225 characters)

RevisitGUID (string): GUID to identify follow-ups belonging together (32 characters)

EMail (string): optional e-mail addresses to be specified, separated by a semicolon (max. 255 characters)

Confirm (INT): optional request to be specified for a password-protected confirmation

If 'Confirm' is not set or set to 0, no password will be requested.

Return:

(INT): 0 = job successful, otherwise error code

abn.UpdateRevisit

Description:

This job sets a new info text and a new follow-up time for an existing follow-up. Optionally, a new e-mail address can be specified.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object

ObjectType (INT): Object type

UserID (INT): ID of the user who receives the existing notification

SetUserID (INT): ID of the user who created the notification

OldRevisitTime (INT): time when the user should receive the notification

SetRevisitTime (INT): time when the existing notification was created

NewRevisitTime (INT): time when the user is expected to receive the notification

NewInfoText (string): new info text (max. 225 characters)

EMail (string): optional e-mail addresses to be specified, separated by a semicolon (max. 255 characters)

Confirm (INT): optional request to be specified for a password-protected confirmation

If 'Confirm' is not specified, this setting will not change.

Return:

(INT): 0 = job successful, otherwise error code

abn.GetUnreadRevisitCount

Description:

This job calculates the number of read and unread follow-ups of a user.

Parameter:

Flags (INT): not currently supported-> transfer 0

UserID (INT): User ID of the user whose follow-ups are to be calculated.

StartTime (INT): Specifies for which time period the follow-ups are to be calculated (optional):

- 0: All follow-ups until the current time (default).
- >0: Specification of time in seconds as of 1970/01/01 01:00:00 (CET) until which follow-ups are to be calculated.

Note:

When a time is specified, 86399 seconds (almost 24 hours) are always added. This is done to simplify specification of a time. To calculate the follow-ups after 2018/01/01, the seconds up to 2018/01/01 00:00:00 are calculated and then you receive all the follow-ups due until 2018/01/01 23:59:59.

Return:

Read (INT): Number of read follow-ups

Unread (INT): Number of unread follow-ups

abn.GetSubscriptions

Description:

This job returns the subscriptions for the logged-on user in the format DMS Content.

Parameter:

Flags (INT): options for this job (currently none available)

[XML (BASE64)]: Request in XML format (see Detailed Description). Thus a selection of the subscribed objects can be made. The possibilities for search requests are hereby limited to the ones of linear search requests.

In addition, the following parameters can be set for formatting the returned XML document: RequestType, OutputFormat, OutputLanguage, Baseparams, Offset, Pagesize, MaxHits, Rights, DateFormat, Variants, FileInfo, Baseparams. The description of these parameters can be found in the description of the job dms.GetResultList

Return values:

[Count] (INT): subscription count

[XML] (BASE64): subscriptions in the DMSContent XML format

The following fields are returned:

Object status fields:

Number of links

Main type (only for documents)

Number of pages (only for documents)

Archiving status (only for documents)

Check-out status (only for documents)

Properties of subscriptions

Time when the follow-up was created (dbname="firstvisit")

Subscription type: (dbname="osabotype"). This field can have the following values

OBJECT_CREATED (@value="2")

DATA_CHANGED (@value="3")

DOCUMENT_CHANGED (@value="4")

DOCMOVEDFROMTRAY (@value="38") is treated like OBJECT_CREATED.

CREATEREFERENCE (@value="39")

Confirmation status (dbname="osconfirm"). Values:

0=does not have to be confirmed

1=has to be confirmed

2=was confirmed

3=has to be confirmed with a password request

ID of the user who triggered the action, which has set the subscription notification for the user (dbname="set_user_id")

GUID of the user who set up the subscription (dbname="set_user_id"). This value will only be set if another user has set up the subscription

Remark concerning the subscription (dbname="infotext")

Index data of the object

Example:

```
<Rowset>
<Columns>
<Column object="Photos" type="DOCUMENT" name="links" system="1" datatype="INTEGER"
dbname="links" ostype="9" size="10">OBJECT_LINKS</Column>
<Column object="Photos" type="DOCUMENT" name="count" system="1" datatype="INTEGER"
dbname="count" ostype="9" size="10">OBJECT_COUNT</Column>
<Column object="Photos" type="DOCUMENT" name="flags" system="1" datatype="INTEGER"
dbname="flags" ostype="9" size="10">OBJECT_FLAGS</Column>
<Column object="Photos" type="DOCUMENT" name="lockuser" system="1"
datatype="INTEGER" dbname="lockuser" ostype="9" size="10">OBJECT_LOCKUSER</Column>
<Column object="Photos" type="DOCUMENT" name="main type" system="1"
datatype="INTEGER" dbname="main type" ostype="9" size="10">OBJECT_MAIN</Column>
```

```xml
<Column object="Revisit" type="REVISIT" name="firstvisit" system="1" datatype="DATETIME"
dbname="firstvisit" ostype="9" size="10">REV_FIRSTVISIT</Column>
<Column object="Revisit" type="REVISIT" name="osabotype" system="1" datatype="INTEGER"
dbname="osabotype" ostype="9" size="10">REV_OSABOTYPE</Column>
<Column object="Revisit" type="REVISIT" name="osconfirm" system="1" datatype="INTEGER"
dbname="osconfirm" ostype="9" size="10">REV_OSCONFIRM</Column>
<Column object="Revisit" type="REVISIT" name="set_user_id" system="1" datatype="INTEGER"
dbname="set_user_id" ostype="9" size="10">REV_SET_USER_ID</Column>
<Column object="Revisit" type="REVISIT" name="osuserid" system="1" datatype="TEXT"
dbname="osuserid" ostype="X" size="32">REV_OSUSERID</Column>
<Column object="Revisit" type="REVISIT" name="infotext" system="1" datatype="TEXT"
dbname="infotext" ostype="X" size="225">REV_INFOTEXT</Column>
<Column object="Photos" type="DOCUMENT" name="Comments" datatype="TEXT"
dbname="field1" ostype="X" size="248">Comments</Column>
</Columns>
<Rows>
<Row id="73543">
<Value>0</Value>
<Value>1</Value>
<Value value="2">NOT_ARCHIVABLE</Value>
<Value value="0">UNLOCKED</Value>
<Value value="3">COLOR</Value>
<Value value="1089804789">2004/07/14 13:33:09</Value>
<Value value="2">OBJECT_CREATED</Value>
<Value>1</Value>
<Value value="53">LOVE</Value>
<Value  value="" />
<Value>New photos</Value>
<Value>Front view</Value>
</Row>
</Rows>
</Rowset>
```

abn.GetRevisits

Description:

This job returns the follow-ups for the logged-on user in the DMS Content format.

Note:

Currently this is not available for portfolios

Parameter:

Flags (INT): options for this job

- 4096 = the XML document is encoded as UTF-8, otherwise UTF-16

StartTime (STRING): optional time stamp for the time by when the follow-ups
are to be returned. Format: YYYY/MM/DD HH:MM.SS, the time does not have to be specified.

Special value 0: (default) all currently available and unchecked follow-ups are returned

In addition, the following parameters can be set for formatting the returned XML document: RequestType, OutputFormat, Baseparams, Offset, Pagesize, MaxHits, Rights, DateFormat, Variants, FileInfo, Baseparams. The description of these parameters can be found in the description of the job dms.GetResultList

Return values:

[Count] (INT): Follow-up count

[XML] (BASE64): Follow-ups in the DMSContent XML format

The following fields are returned:

- Object status fields:
  - Number of links
  - Main type (only for documents)
  - Number of pages (only for documents)
  - Archiving status (only for documents)
  - Check-out status (only for documents)
- Properties of the follow-up:
  - Time when the follow-up was created (dbname="set_time")
  - Time as of when the follow-up is displayed to the user (dbname="firstvisit")
  - Time at which the user has acknowledged the follow-up (dbname="lastvisit"). See also 'abn.SetObjRevisitClosed' or 'abn.SetObjRevisitOpen'
  - Comment on template (dbname="infotext")
  - Confirmation of the follow-up (dbname="osconfirm")
  - (0=no confirmation with a password expected, 1 = confirmation only possible with password input)
  - The user who set up the follow-up (dbname="set_user_id")
- Index data of the object

Example:

```
<Rowset>
<Columns>
<Column object="Grayscale image" type="DOCUMENT" name="links" system="1"
datatype="INTEGER" dbname="links" ostype="9" size="10">OBJECT_LINKS</Column>
<Column object="Grayscale image" type="DOCUMENT" name="count" system="1"
datatype="INTEGER" dbname="count" ostype="9" size="10">OBJECT_COUNT</Column>
<Column object="Grayscale image" type="DOCUMENT" name="flags" system="1"
datatype="INTEGER" dbname="flags" ostype="9" size="10">OBJECT_FLAGS</Column>
<Column object="Grayscale image" type="DOCUMENT" name="lockuser" system="1"
datatype="INTEGER" dbname="lockuser" ostype="9" size="10">OBJECT_LOCKUSER</Column>
<Column object="Grayscale image" type="DOCUMENT" name="main type" system="1"
datatype="INTEGER" dbname="main type" ostype="9" size="10">OBJECT_MAIN</Column>
<Column object="Revisit" type="REVISIT" name="set_time" system="1" datatype="DATETIME"
dbname="set_time" ostype="9" size="10">REV_SET_TIME</Column>
<Column object="Revisit" type="REVISIT" name="lastvisit" system="1" datatype="DATETIME"
dbname="lastvisit" ostype="9" size="10">REV_LASTVISIT</Column>
<Column object="Revisit" type="REVISIT" name="firstvisit" system="1" datatype="DATETIME"
dbname="firstvisit" ostype="9" size="10">REV_FIRSTVISIT</Column>
<Column object="Revisit" type="REVISIT" name="osconfirm" system="1" datatype="INTEGER"
dbname="osconfirm" ostype="9" size="10">REV_OSCONFIRM</Column>
<Column object="Revisit" type="REVISIT" name="set_user_id" system="1" datatype="INTEGER"
dbname="set_user_id" ostype="9" size="10">REV_SET_USER_ID</Column>
<Column object="Revisit" type="REVISIT" name="infotext" system="1" datatype="TEXT"
dbname="infotext" ostype="X" size="225">REV_INFOTEXT</Column>
<Column object="Grayscale image" type="DOCUMENT" name="Document type"
datatype="TEXT" dbname="field1" ostype="X" size="30">Document type</Column>
<Column object="Grayscale image" type="DOCUMENT" name="Date" data type="DATE"
dbname="date1" ostype="D" size="10">Date</Column>
<Column object="Grayscale image" type="DOCUMENT" name="Author" data type="TEXT"
dbname="field2" ostype="X" size="50">Author</Column>
<Column object="Grayscale image" type="DOCUMENT" name="Source" data type="TEXT"
dbname="field3" ostype="X" size="150">Source</Column>
<Column object="Grayscale image" type="DOCUMENT" name="Content" data type="TEXT"
dbname="field4" ostype="X" size="150">Content</Column>
</Columns>
<Rows>
<Row id="415">
<Value>0</Value>
<Value>1</Value>
<Value value="2">NOT_ARCHIVABLE</Value>
<Value value="">UNLOCKED</Value>
<Value value="1">GRAYSCALE</Value>
<Value value="1089723507">2004/07/13 14:58:27</Value>
<Value  value="0" />
<Value value="1089723600">2004/07/13 15:00:00</Value>
<Value />
<Value value="53">LOVE</Value>
<Value>Please view</Value>
<Value>Drawing</Value>
<Value>2002/09/04</Value>
<Value>Love</Value>
<Value>Cave</Value>
<Value>Cattle</Value>
</Row>
<Row id="416">
<Value>0</Value>
<Value>0</Value>
<Value value="8">NO_PAGES</Value>
<Value value="">UNLOCKED</Value>
<Value value="1">GRAYSCALE</Value>
```

```
<Value value="1089727995">2004/07/13 16:13:15</Value>
<Value  value="0" />
<Value value="1089728100">2004/07/13 16:15:00</Value>
<Value />
<Value value="53">LOVE</Value>
<Value>Deliver image subsequently</Value>
<Value>Annoyances</Value>
<Value>2002/09/05</Value>
<Value>Admin</Value>
<Value>Unknown</Value>
<Value>Bat</Value>
</Row>
</Rows>
</Rowset>
```

abn.SetOsInformed

Description:

This job sets an object in the subscription list to 'read'

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object

ObjectType (INT): Object type

UserID (INT): ID of the user who receives the existing notification

AboSetTime (INT): time stamp when the subscription was created

Return:

(INT): 0 = job successful, otherwise error code

abn.ResetOsInformed

Description:

This job sets an object in the subscription list to 'unread'

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the object

ObjectType (INT): Object type

UserID (INT): ID of the user who receives the existing notification

AboSetTime (INT): time stamp when the subscription was created

Return:

(INT): 0 = job successful, otherwise error code

abn.GetRecentObjects

Description:

This job determines the dms objects recently edited by the logged in user.

Parameter:

Flags (INT): not currently supported-> transfer 0

Count (INT): number of DMS objects to be determined

HistActIDList (String): comma-separated list of HistActID

Return:

Recent (String): Comma and semicolon-separated list of the determined DMS objects.

Parameter (In):

Count (Required)	DWORD	Number of recently edited dms objects. The value's range is limited from 5 to 99. When indicating a value beyond the range limits, either 5 or 99 is used as value, depending on which range limit the indicated value is close to the most.
		E.g.
		Count=0 -> Count=5
		Count=1000 -> Count=99

HistActIDList (Optional)	String	A comma-separated list of osHistAct actions, which are to be used to determine the most recently edited dms objects. If this parameter is not indicated, the actions "2,3,4" will be used. 2 - Object created: the specified object was created by functions of the client or by an import. 3 - Index data modified: the index data of the object or its status was modified by functions of the client or by an update by means of the import. 4 - Document changed: the document was edited by functions of the client or by an update by means of the import.

Parameter (Out):

Recent	String	Comma and semicolon separated list of determined DMS objects. Form: ObjId1,ObjType1,Action1,Time1;…; ObjId(n),ObjType(n),Action(n),Time(n) E.g. 12,0,2,12389147391; 13,1,2,12389137474; 14,0,3,12389127897 The order is sorted by time in descending order. I.e. the first list item is the most recently edited dms object.

ADO Database Engine (Namespace ado)

The ADO database engine makes it possible to access the database within the three-tier architecture. This is done as follows. The client sends an SQL query to the application server which executes the query and sends the result as ADO (Active Data Object) record set in XML representation to the client.

These features are used especially for SQL queries, to formulate queries that go beyond the normal search options of the client.

With this interface, data manipulations (INSERT, UPDATE, DELETE) can then be carried out as well, if it is allowed in the registry of the application server.

ado.ExecuteSQL

Description:

This job executes an SQL statement in the database. The result is stored in an XML file and saved in the ostemp directory.

Parameter:

Flags (LONG): currently not supported transfer-> 0 passed

CursorType (int): -1 = Cursor type as specified in the registry (default); 0,1,2,3 = in accordance with the ADO constants for cursor types. Others values lead to error messages.

Command (STRING): SQL command for execution

Return:

(INT): 0 = job successful, otherwise error code

Return values:

File list: path and name of the result file in XML format

Example:

Returned XML file for 'SELECT * FROM osorganizations'

```
<xml xmlns:s="uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882
xmlns:dt="uuid:C2F41010-65B3-11d1-A29F-00AA00C14882"
xmlns:rs="urn:schemas-microsoft-com:rowset" xmlns:z="#RowsetSchema">
<s:Schema id="RowsetSchema">
<s:ElementType name="row" content="eltOnly" rs:CommandTimeout="30"
rs:updatable="true">
<s:AttributeType name="id" rs:number="1" rs:nullable="true"
rs:writeunknown="true" rs:basecatalog="as_test"
rs:basetable="osorganizations" rs:basecolumn="id">
<s:datatype dt:type="string" rs:dbtype="str" dt:maxLength="32"/>
</s:AttributeType>
<s:AttributeType name="name" rs:number="2" rs:nullable="true"
rs:writeunknown="true" rs:basecatalog="as_test"
rs:basetable="osorganizations" rs:basecolumn="name">
<s:datatype dt:type="string" rs:dbtype="str" dt:maxLength="255"/>
</s:AttributeType>
<s:AttributeType name="layout" rs:number="3" rs:nullable="true"
rs:maydefer="true" rs:writeunknown="true"
rs:basecatalog="as_test" rs:basetable="osorganizations"
rs:basecolumn="layout">
<s:datatype dt:type="bin.hex" dt:maxLength="2147483647"
rs:long="true"/>
</s:AttributeType>
<s:AttributeType name="active" rs:number="4" rs:nullable="true"
rs:writeunknown="true" rs:basecatalog="as_test"
rs:basetable="osorganizations" rs:basecolumn="active">
<s:datatype dt:type="int" dt:maxLength="4" rs:precision="10"
rs:fixedlength="true"/>
</s:AttributeType>
<s:extends type="rs:rowbase"/>
</s:ElementType>
</s:Schema>
<rs:data>
<z:row id="45808CE977334AB88C5A8EFF467689A8" name="Test" active="1"/>
```

```
</rs:data>
</xml>
```

Convert Engine (Namespace cnv)

This engine provides jobs for the conversion of image files.

- cnv.ConvertDocument
- cnv.CreateSlide
- cnv.AddAnnotations
- cnv.GetIcons
- cnv.GetExifData
- cnv.GetPageCount
- cnv.GetPictureInfos
- cnv.GetRendition

cnv.ConvertDocument

Description:

This job converts one or more document files of the given format into one or more PDF or TIFF files. Depending on the used configuration, any number of input and output formats are available.

For example, the following classes of transformation are possible:

Bitmap format (JPG, TIF)-> PDF

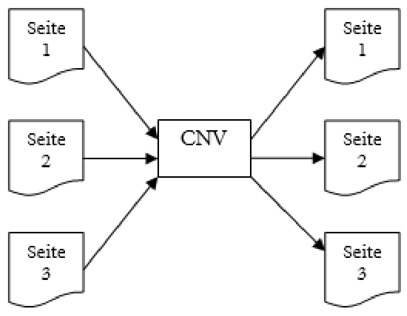
Single page TIFF -> Multi page TIFF

ASCII-COLD (asc)-> TIFF / PDF

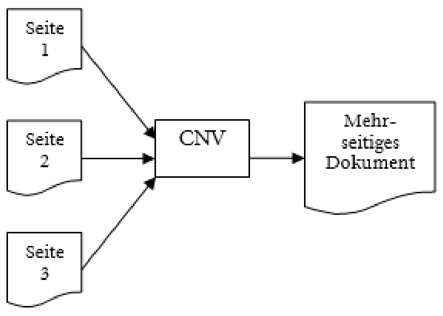
XSL:FO (.fo, .xml) -> PDF

Office documents (`ps, doc, xls, ppt, txt, rtf, pdf, ima`)-> PDF

For bitmap formats and ASCII COLD documents, a flag can be set to control whether an output file is created for each input file (A) or the individual pages are merged in one document (B):



A



B

Parameter:

Flags (INT): Options for the job

- 0 = an output file is created for each input file
- 1 = all input files are written into an output file
- 2 = an output file is created for each input file, temporary files created by the server are not deleted

§ 3 = all input files are written into an output file, temporary files created by the server are not deleted

SourceFormat (STRING): output file format

DestinationFormat (STRING): destination conversion format (see above).

- Note: The format specifications are expected in lower case ('pdf' instead of PDF).

Timeout (INT): (optional) maximum time in milliseconds for a conversion using external programs. Taken into account for XSL:FO and Office format conversions.

AddAnnotations (INT): 1 = Public layers are burnt in.

ConvertEqualFormat (INT): (Optional, default is 0) if set to 1, conversion is attempted even if source and target format are identical, e.g. to convert a PDF into a PDF/A.

Watermark (INT): (optional, default is 0) if set to 1, header and footer lines are added to a created PDF document. Design and content are configured in enaio® administrator. See enaio® administrator handbook -> print labeling tab.

An additional possibility is to use the extended watermarks function. To do so, request the documentation for the oxsvrspt.dll library.

dwObjectID (INT): (optional) document object ID for use in watermarks. Only required if the Watermark option was set to 1 and the print watermarks are configured to include the document ID at output.

ProtectPDF (INT): (optional, default is 0) a created PDF document is protected. It is then no longer possible to print the document or to copy text from it.

ObjectID (INT): (required parameter) document ID

ObjectType (INT): (required parameter) object type

or

File list: path and name of the files to be converted.

If ObjectID and ObjectType are passed, OS RenditionPlus (___ren.bat) is used for conversion.

Digest (STRING): (optional) hash value of the document

Optionally, 'SLIDE' can be passed as target format for preview generation. With 'Height' and 'Width' you can specify the size of the preview to be generated.

Thus, 'TXT' instead of 'SLIDE' is allowed as target format also for text recognition.

Return values:

File list: Path and name of the converted file(s)

Return:

(INT): 0 = job successful, otherwise error code

See also:

cnv.CreateSlide

Use of XSL:FO

§ The Apache FOP processor must be installed for conversion of FO (formatting objects) files. It requires Java runtime environment version 1.6 or higher. The path to the FOP batch file (Windows) or to the FOP shell script (Linux) must be saved in the registry using e.g. the Enterprise

Manager. Furthermore, a time (timeout) can be defined after which the conversion process will be aborted.

§ If an XML document is transferred to the FOP, an XSLT document has to be additionally transferred to the input file list as a **second file**, with which the XML can be transferred to an XSL:FO document.

§ If images are to be integrated into the resulting PDF document, they have to be referenced in the XSL:FO file with a URL accessible by the application server.

§ Only one FO or XML document can be processed per job.

Example of a cnv.ConvertDocument call:

SourceFormat: XML
DestinationFormat: PDF
Flags=0

1. Input file: XML file with any input information.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="example.fo"?>
<persons>
<person age="42">John Doe</person>
<person age="37">Jane Doe</person>
</persons>
```

2. Input file: XSL file for conversion into an XML:FO file.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format"  version="1.0">
<xsl:template match="persons">
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
<fo:simple-page-master page-master-name="one" page-height="29cm" page-width="21cm"
margin-left="2cm" margin-right="2cm">
<fo:region-body margin-top="50pt" margin-bottom="50pt"/>
</fo:simple-page-master>
</fo:layout-master-set>
<fo:block text-align="center" font-size="24pt" font-weight="bold" line-
height="28pt" space-after="10mm">
<xsl:apply-templates/>
</fo:block>
</fo:root>
</xsl:template>
<xsl:template match="person">
...
</xsl:template>
</xsl:stylesheet>
```

cnv.CreateSlide

Description:

This job converts files and documents of the formats JPG or TIF into a slide file with the same format of the input file. Slide files are highly compressed and help to preview the document.

Parameter:

Flags (INT): Options for the job

§ 0 = a slide output file is created

§ 2 = a slide output file is created, temporary files created by the server are not deleted

(required parameter)

ObjectID (INT): ID of the document

ObjectType (INT): Object type

or

File list: Path and name of the file in JPG or TIF format to be converted.

Page (STRING): (optional) page number of the document (file list) for which data is to be determined. Default '1', 'All' for all pages.

Return values:

File list: path and name of the converted slide file(s) with the same format as the input file(s)

Return:

(INT): 0 = job successful, otherwise error code

See also:

cnv.ConvertDocument

cnv.AddAnnotations

Description:

This job is used to burn annotations (transparencies) on one or more image files.

Note:

The job takes into account access rights to the annotations. Thus only the personal transparency of the user who is logged on to the system will be displayed.

Parameter:

Flags (INT):

§ 0 = the input files are deleted. A temporary copy of unaltered input files is created.

§ 2 = Input files are not deleted. Unaltered input files are written back to the output list.

AnnotationID (INT): (optional) specific ID of an annotation to be burned on the input file(s).

dwObjectID (INT): (optional) ID of an object to be burned on the input file(s). AnnotationID can, dwObjectID has to be transferred for this.

File list: paths and names of the files to be annotated.

Return values:

nProcessed: number of image files that have been annotated.

sAnnotationIDs: comma-separated list of AnnotationIDs, which were burned on the files.

File list: path and name of the annotated image file if annotations were available for this input file. Otherwise the input file will be returned without any changes (refer to Flags). The output format corresponds to the format of the input file.

Return:

(INT): 0 = job successful, otherwise error code

cnv.GetIcons

Description:

This job returns icons in GIF format. This enables to read user-specific icons which are used in the archive area or in the hit lists.

Parameter:

Flags (INT): not currently supported-> transfer 0

sIconIds (String): comma-separated list of icon IDs.

Return values:

sIconIds (String): comma-separated list of icon IDs in the same order as the output files.

File list: path and name of the icons in GIF format.

Return:

(INT): 0 = job successful, otherwise error code

cnv.GetExifData

Description:

This job determines the EXIF data from the image files of a document.

Data (EXIF, Dicom and general data) are determined only for EXIF, JPEG, TIF, and Dicom files.

Parameter:

ObjectID (INT): ID of the document

ObjectType (INT): Object type

or

File list: paths and names of the files from which data will be determined.

Flags (INT): (required parameter) 1 = do not delete transferred file list, 0 = delete for client-side calls.

Page (STRING): (optional) page number of the document (file list) for which data is to be determined. Default '1', 'All' for all pages.

Return values:

Contained information.

The information is returned as 'lead tool name(page number) value'.

e.g.:

CMNT_SZMAKE(1)   "Panasonic"

CMNT_SZMODEL(1) "DMC-TZ10"

...

Return:

(INT): 0 = job successful, otherwise error code

cnv.GetPageCount

Description:

This job determines the total number of pages from the image files of a document.

This applies only to image and PDF documents. If the total number of pages could be determined from all files in the list or from the document, 'FileCount' and 'PageCount' are returned.

Parameter:

(required parameter)

ObjectID (INT): ID of the document

ObjectType (INT): Object type

or

File list: paths and names of the files whose total number of pages are to be determined.

Flags (INT): (required parameter) 1 = do not delete transferred file list, 0 = delete for client-side calls.

Return values:

FileCount (INT): number of files of the document (file list)

PageCount (INT): number of files of the document (file list)

Return:

(INT): 0 = job successful, otherwise error code

cnv.GetPictureInfos

Description:

This job determines image information from the files of a document.

This applies only to image and PDF documents.

Parameter:

(required parameter)

ObjectID (INT): ID of the document

ObjectType (INT): Object type

or

File list: paths and names of the files whose total number of pages are to be determined.

Flags (INT): (required parameter) 1 = do not delete transferred file list, 0 = delete for client-side calls.

Page (STRING): (optional) page number of the document (file list) for which data is to be determined. Default '1', 'All' for all pages.

Return values:

Image information; the information is returned as 'info name(page number) value'.

The following information (INT) is returned for image documents:

"InfoBits(1)"

"InfoCompress(1)"

"InfoFormat(1)"

"InfoIFD(1)"

"InfoLayers(1)"

"InfoSizeDisk(1)"

"InfoSizeMem(1)"

"InfoViewPerspective(1)"

"InfoHeight(1)"

"InfoWidth(1)"

"InfoFileType(1)"

"InfoFileSuffix(1)"

The page number is indicated by the respective value in brackets.

For PDF files, the following information is returned:

"InfoFormat(1)"

"InfoFileType(1)"

"InfoFileSuffix(1)"

Return:

(INT): 0 = job successful, otherwise error code

cnv.GetRendition

Description:

This job executes a rendition of a document using a call via the ___ren.bat file.

Parameter:

(required parameter)

ObjectID (INT): ID of the document

ObjectType (INT): Object type

Flags (INT): (required parameter) 1 = do not delete transferred file list, 0 = delete for client-side calls.

DestinationFormat (STRING): format into which the file will be converted. See also cnv.ConvertDocument

Optionally, 'SLIDE' can be passed as target format for preview generation. With 'Height' and 'Width' you can specify the size of the preview to be generated.

'TXT' instead of 'SLIDE' is allowed also as target format for text recognition.

Digest (STRING): (optional) hash value of the document

Watermark (INT): (optional, default is 0) if set to 1, header and footer are added to a created PDF document. Design and content are configured in enaio® administrator. See enaio® administrator handbook, chapter: 'Print Labeling' Tab

AddAnnotations (INT): 1 = Public layers are burnt in.

ProtectPDF (INT): (optional, default is 0) a generated PDF document is protected. Thus, the document cannot be printed and it is not possible to copy paragraphs.

SynchData (INT): (optional, default is 0)

Return values:

File list: Path and name of the converted file(s)

Return:

(INT): 0 = job successful, otherwise error code

DMS Engine (Namespace dms)

The DMS Executor includes jobs to request and edit index data, DMS objects, relations, and portfolios while taking account of the security system. In addition, there are jobs to administer the security system on the object level.

Areas

- § XML import
- § XML export (search)
- § Security system
- § Relations and relation texts
- § Portfolios
- § User-related data
- § Other jobs

XML Import

- § DMS.CheckInDocument
- § DMS.CheckOutDocument
- § DMS.GetXMLJobOptions
- § DMS.UndoCheckOutDocument
- § DMS.XMLDelete
- § DMS.XMLInsert
- § DMS.XMLMove
- § DMS.XMLCopy
- § DMS.XMLUpdate
- § DMS.XMLImport
- § DMS.XMLUnknownToKnown

Detailed Description

All XML import jobs have the same schema and can therefore usually be treated with a standard procedure. They all have the same input parameters and basically the same output parameters. There is for example the XMLInsert job which has the transfer parameters 'flags', 'options' and 'XML'.

The general behavior of the job (e.g. error list generation or XML validation) is controlled by the 'Flags'. The 'Options' parameter can be used for switching certain checks on or off (e.g. key field check). The parameter 'XML' contains the data describing the object to be inserted. The parameter 'JobUserGUID' can be used to modify the user context for this job.

Example:

The example shows an XML file which inserts a document with the document type: 'document type name' from the cabinet: 'cabinet name' with the XMLInsert job. The field with the name 'field name' and the value 'my value' are inserted into the new document's index data.

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive name="cabinet name">
<ObjectType name="document type name" type="DOCUMENT">
<Object>
<Fields>
<Field name="Field Name">My Value</Field>
</Fields>
</Object>
</ObjectType>
</Archive>
</DMSData>
```

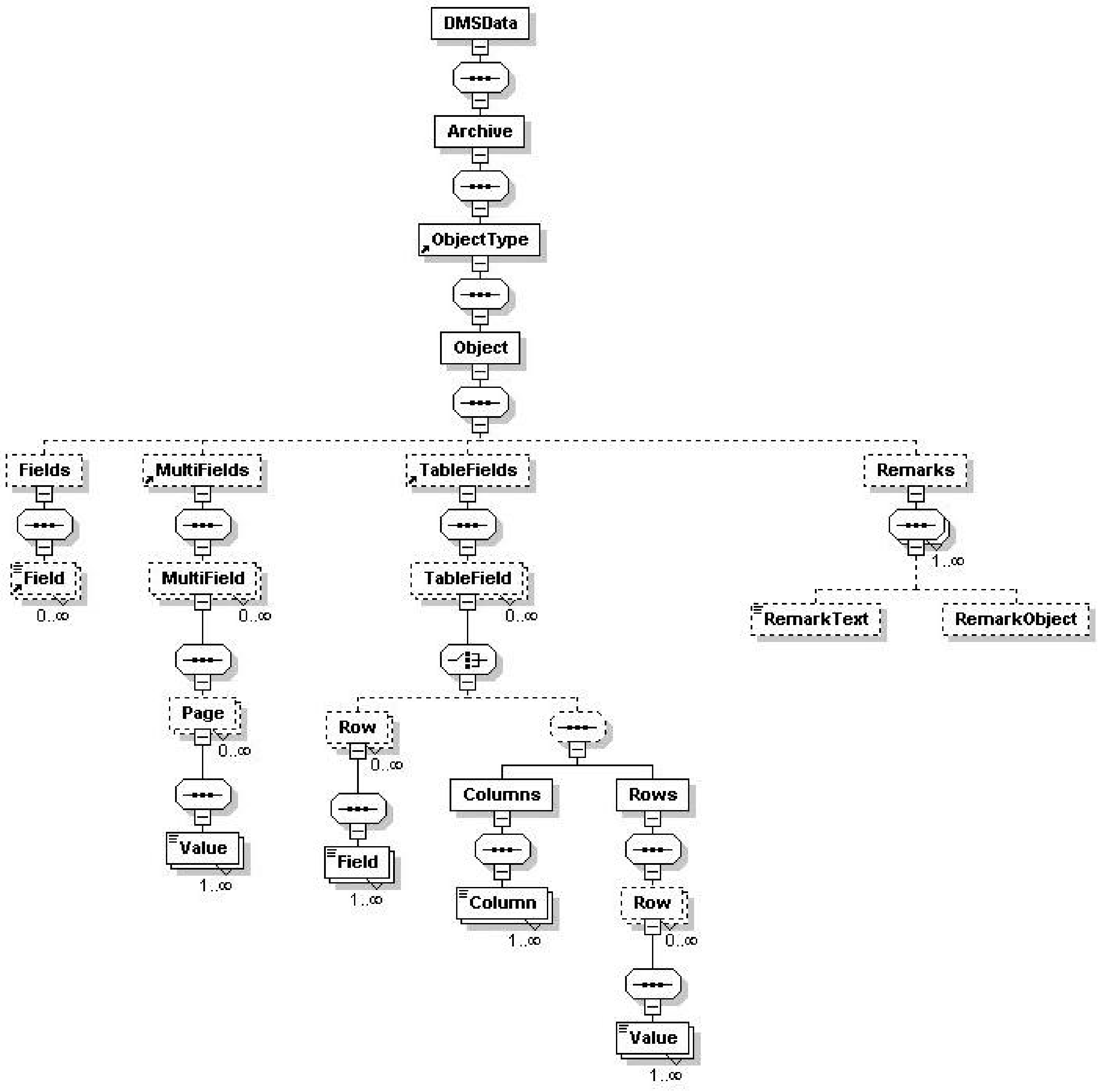
Return parameters are the object ID of the newly inserted object and optionally (see Flags) an XML text describing the errors which occurred. Every job has a return value beyond the parameters. Upon successful completion, the return value will be 0. Otherwise a value describing the nature of the possible error will be returned.

The 'XML' Parameter

With the 'XML' parameter, the object description will be transferred in XML format. XML has to be Base64-encoded as the server otherwise has problems to transfer special characters. Also UTF-16/UCS-4 formats might not be transferred in such a case. A schema exists for the XML structure against which the XML can be validated. Job 'GetXMLSchema' exists in order to view this schema. Enter the 'DMSData' value in the 'Schema' parameter and get the import schema in the return file list. If you enter attributes, which you do not want to be evaluated by the job, text attributes have to be initialized with the empty string '' and numeric attributes have to be initialized with the value '1.'

The Import Schema

The hierarchical XML structure is shown in the figure below.



Description of the XML Schema

DMSData: this tag represents the root tag of the XML. With the optional 'query language' attribute you can specify the language for the DMS identifier. The relevant language code is expected as a value, e.g. '7' for German or '9' for English. If this attribute is not specified or set to '0', the request language will be the default language.

Archive: The attributes of these tags are used to identify the cabinet in which the object type to be handled is located or is to be inserted. If no potential attributes are specified, the system tries to find the right cabinet according to the object type which will be imported. However, this is done at the expense of the performance and should be avoided whenever possible. The cabinet can be identified based on the attributes: ID (ID of the object type), name (name of the cabinet), internal_name (internal cabinet name) or osguid (GUID of the cabinet).

ObjectType: The object type of the object to be handled is defined within these tags. In the attributes, it has to be specified whether the type of the involved object is a folder, register or document (possible values are FOLDER, REGISTER and DOCUMENT). The object type can be determined by analyzing the attributes: name, internal_name, object type ID, table name, osguid or a valid combination of maintype and cotype.

Object: This tag contains information specifically concerning the object to be processed. If an object, which already exists on the server, has to be specified by the job (e.g. XMLUpdate), the object_id (ID of the folder, register or document) has to be set as attribute. If a location has to be specified (e.g. for XMLInsert()), it is described here as a combination of register_id, register_type, and/or folder_id. In addition, it is also possible to determine the maintype for documents. Variants of W-documents can be specified by setting the 'variantparent_id' attribute (document ID to which the variant will be added; see also Options: Variantsamelevel, Variantsetactive). The sub elements of these tags have to be indicated in the prescribed order in the XML. It is possible to check whether this object has been edited by another user since the last query by indicating the 'concurrency_timestamp' attribute in connection with the timestamp of the 'MODIFYTIME' field which has to be queried before. An update is refused in this case. With the 'sourceparent_id' attribute, you can determine which location will be changed if a document with several locations is moved or deleted.

Warning: if the main type of a document is changed and a document log is used at the same time, only the document will be restored in case of a recovery, however, not the modified main type. This may cause inconsistencies when a document is edited!

Fields: this tag is allowed only once for each object and represents a list of all fields which are to be edited.

Field: this tag describes the field which will be inserted or modified. The field can be identified on the basis of the attributes: name (name of the field on the data sheet), internal_name (internal field name), dbname (field name in the database), sortpos (tab position on the data sheet) or osguid (GUID). The element's text contains the value which will be assigned to the field. It is also possible to set a value to 'blank' by indicating the 'field_function' attribute (see below). If the field is a system field, this has to be indicated by entering the value '1' for the system attribute (system). Available system fields are:

Database name	**Internal name**	**Description**
OSOWNER	OBJECT_USERGUID	Owner of the object
FOREIGNID	OBJECT_FOREIGNID	Foreign ID of a reference
SYSTEMID	OBJECT_SYSTEMID	System ID of an object

Warning: if there is a reference to a document with variant administration by means of the system ID (=0) and a corresponding foreign ID, this may lead to inconsistency if variants are activated with main types which are not the same main types as those of the reference!

Furthermore, a 'field_function' attribute can be set optionally. The following values are allowed:

Attribute value	**Description**
NULL	The field value is blank (as DBNull)
OBJECT_ID	The field's value is equivalent to the unique internal ID of the object.

USER	Name of the logged-out user

MultiFields: This tag is allowed only once for each object and represents a list of all multi fields which will be processed. Multi fields are only allowed for documents.

Multifield: This element is used to identify a specific multi field of a document. The multi field can be identified by the attributes: name (name of the multi field), internal_name, dbname (name of the field in the database) or osguid (GUID).

Page: This element identifies a certain page whose page number has to be indicated in the 'ID' attribute.

Value: Values which correspond to the specified pages. The value is part of the element's text.

TableFields: This tag is allowed only once for each object and represents a list of all table fields of the object to be processed.

TableField: A specific table field (table control) of an object is identified by this element. A table field can be identified based on the attributes: name, internal_name, dbname (name in the database) or osguid (GUID).

Row: This tag describes a table field row. Thus, it is a summary of all columns of a certain row. (As an alternative to this specification of table field entries, they can also be determined by a (LOL) specification of rows and columns. See below.)

Field: This tag is a value of a specific column in a row of the table field. It can only be identified by the attributes: name, internal_name or dbname (name in the database). The element text contains the value which will be added.
Furthermore, a 'field_function' attribute can be optionally set (see above for details).

OR

Columns: This tag is a collection of column tags.

Column: This tag describes a field within a table control. The field can be identified by its attributes: name, internal_name or dbname (name in the database). In the following tags in rows, the field values are listed in the same order as the columns.

Rows: This tag describes a list of row tags and thus a collection of table control rows.

Row: A row, which corresponds to a row in the table control, is described here.

Value: This tag describes a field value of a row. The assigned field also must have been indicated in the 'Column' tag at the according position. The element's text is equivalent to a value of the field.
Furthermore, a 'field_function' attribute can be optionally set (see above for details).

Remarks: This tag describes a list of remarks which will be assigned to the object.

RemarkText: This tag describes a remark text for the specified object. The element's text is equivalent to the note's text.

RemarkObject: This tag describes an object, which is to be linked to the specified object via the remarks. The object which has to be linked with the object_id attribute and the object_type attribute has to be selected.

The 'Flags' Parameter

Every job has a 'Flags' parameter. Flags allow to control the general behavior of the job. Flags can also be combined. If, for example, an error list has to be returned in the data format (flags = 1) and UTF-16 encoded (flags = 16), the 'Flags' parameter has to be set to the value '17' (1 + 16). The following flags exist:

Flag	Description
1	The error list is returned as a file.
2	The job does not return an error list.
4	The job does not perform a validation with the XSD file
8	The provided files are not deleted on the server.
16	The returned error list is UTF-16 encoded. (Default value UTF-8)

The 'Options' Parameter

The 'Options' parameter allows you to determine which checks are carried out. This can partly lead to great improvements in performance. The parameter value is a semicolon-separated list with the format: OPTION1=1; OPTION2=0; ....

Option	Description	Default	Usable for the job
APPENDFILESTOFRONT	For updates or variant generation whether files are appended before (1) or after (0) the existing one	(0) Option not active	XMLInsert XMLUpdate
ARCHIVABLE	Specifies whether the document's status is set to archivable (1) or not archivable (0)	XMLInsert: (0) Option not active XMLUpdate: Option not defined	XMLInsert XMLUpdate XMLCopy
CHECKACCESS	User permissions for the object are checked (1) or not checked (0)	(1) Option active	XMLInsert XMLUpdate XMLMove XMLDelete XMLCopy
CHECKCATALOGUE	Checks whether all transferred catalog entries are also contained in the original catalog (1) or not checked (0)	(1) Option active	XMLInsert XMLUpdate
CHECKEXISTENCE	Checks whether the	(1) Option	XMLInsert

	specified object exists at the given location (1) or is not checked (0)	active	XMLUpdate XMLMove XMLDelete XMLCopy
CHECKKEYFIELDS	Checks whether all key fields are unambiguous (1) or not checked (0)	(1) Option active	XMLInsert XMLUpdate
CHECKOBLIGATION	Checks whether all mandatory fields have been completed (1) or not checked (0)	(1) Option active	XMLInsert XMLUpdate
CHECKPOSITION	Checks whether specified (target) objects (e.g. folders) exist (1) or not checked (0)	(1) Option active	XMLMove XMLCopy
CHECKREADONLY	Checks whether any fields were changed without write access (1) or not checked (0)	(1) Option active	XMLUpdate
DELETECASCADING	Specifies whether cascading deletion is carried out for objects even if they contain sub objects (1) or not (0)	(0) Option not active	XMLDelete
COPYCASCADING	Specifies whether objects are copied in a cascading way if they contain sub objects (1) or not (0)	(0) Option not active	XMLCopy
FULLTEXTFILEATTACHED	Specifies whether full-text data for the document will be attached to the last transferred file (1) or not (0)	(0) Option not active	XMLInsert XMLUpdate
HARDDELETE	Specifies whether an object is deleted permanently and therefore not moved to the trash can (1) or not (0)	(0) Option not active	XMLDelete
INITFIELDS	Specifies whether all uncompleted fields will be filled out with default values (1) or not (0)	(1) Option active	XMLInsert XMLUpdate
INUSERTRAY	Specifies whether the object is inserted in the user's filing tray (1) or not (0)	(0) Option not active	XMLMove XMLInsert
INWFTRAY	Specifies whether the object is inserted in the user's	(0) Option not	XMLMove

	workflow tray (1) or not (0)	active	XMLInsert
REPLACEFILES	Specifies whether files which have already been saved are replaced by the transferred files (1) or whether the transferred files are appended (0)	XMLInsert (variants): (1) Option active, XMLUpdate: (0) Option not active	XMLUpdate XMLInsert
REPLACEMULTIFIELDS	Specifies whether the transferred multi-fields replace the originals (1) or whether they are appended (0)	(0) Option not active	XMLUpdate
REPLACEREMARKS	Specifies whether the transferred notes replace the originals (1) or whether they are appended (0)	(0) Option not active	XMLUpdate
REPLACETABLEFIELDS	Specifies whether the transferred table fields replace the originals (1) or whether they are appended (0)	(0) Option not active	XMLUpdate
TRUNCATEVALUES	Specifies whether transferred strings will be truncated if their character number exceeds the defined value (1) or not (0)	(0) Option not active	XMLInsert XMLUpdate
TYPELESS	Specifies whether a typeless object is inserted in the filing tray (1) or not (0)	(0) Option not active	XMLInsert
UPDATEALLFIELDS	Specifies whether fields which have not been specified are set to blank (1) or not (0)	(0) Option not active	XMLInsert XMLUpdate
VARIANTSAMELEVEL	Specifies whether the variant is inserted on the same layer (1) or as a "sub-variant" (0)	(0) Option not active	XMLInsert
VARIANTSETACTIVE	Specifies whether the new variant is set as "Active" in the same step (1) or not (0)	(0) Option not active	XMLInsert
VARIANTTRANSFERRETENTION	Specifies whether the original retention time is assigned to the new variant (1) or not (0)	(0) Option not active	XMLInsert

LINKDOCUMENT	Specifies whether a document is allowed to receive only one location (1) or not (0)	(0) Option not active	XMLCopy
WFTOUSERTRAY	Specifies whether, when a document is moved, it is moved from the workflow tray to the user tray (1) or not (0)	(0) Option not active	XMLMove
KEEPLINKWHENEXISTS	Specifies whether an existing link should be assessed as an error (1) or not (0) (Applies for XMLCopy only in connection with the option LINKDOCUMENT)	(0) Option not active	XMLCopy XMLMove
DELETEVARIANTMODE	Specifies whether the deletion of a given inactive variant leads to the deletion of the whole variant tree (1) or not (0)	(0) Option not active	XMLDelete
COPYINDEXONLY	Specifies whether only the index data is copied (1) or not (0)	(0) Option not active	XMLCopy
COPYCREATEHISTORY	Specifies whether information for copying is entered in the history (1) or not (0)	(1) Option active	XMLCopy

The 'JobUserGUID' Parameter

The 'JobUserGuid' parameter allows the user to change the context of the XML job. This is true for all XMLImport jobs, i.e. all 'XML...' jobs which modify objects in the DMS. If the user's GUID is specified here, all checks (e.g. access rights), filing trays or similar are used only by the specified user.

Warning: This option is only available if the corresponding jobs are called from the server by other jobs. A client, however, will not be allowed to use this option!

The 'File_N' Parameter

Every job which analyzes transferred files can instead be given a number of parameters. In an ascending order, the parameters are called, File_0, File_1, etc. The parameters have the 'string' type and include a file path, the file therefore does not have to be transferred to the file list. This is for cases in which the caller is logged on the server's processor. The effort of transferring a file through the network adapter is avoided. In this case, read or write access to the server path (whether local or with UNC notation) has to be guaranteed. Either a list of files can be transferred or a number of file parameters. It is not allowed to mix both.

The Return Value

Every job generates a return value. In case of success it is always '0'. Otherwise a return value describing the nature of the possible error will be returned. Exact error messages, however, are necessary for a detailed analysis. The following return values exist:

Error code	Description
0	The job has been successfully executed.
-1	A general error has occurred. (In this case the error cannot be specified)
-2	No cabinet has been specified.
-3	The specified cabinet is unknown
-4	No register type has been specified.
-5	The specified register type is unknown.
-6	No document type has been specified.
-7	The specified document type is unknown.
-8	The specified register does not exist in the specified folder.
-9	The specified document type is not allowed in the specified cabinet.
-10	The required folder identifier is missing.
-11	The required document identifier is missing.
-12	The required register identifier is missing.
-13	The required register identifier is unknown.
-14	The required folder identifier is unknown.
-15	The required document identifier is unknown.
-16	Updating folder failed.
-17	Updating folder failed.
-18	Updating register failed.
-21	The document has already been archived.
-22	The specified/required object is unknown.
-23	The ID of the specified register does not exist on the archive server.
-24	The field name could not be resolved.
-28	Invalid field value for field.
-29	The specified objectID is invalid.
-30	The required fields have not been filled out.
-31	The specified value does not match the type on the archive server.
-32	The specified/required file does not exist.

-40	A transfer parameter contains errors or is missing.
-47	The user does not have the appropriate rights on the archive server.
-51	The document contains no pages.
-65	No index could be obtained from the server.
-68	It is not allowed to move folders.
-89	Invalid relation between document and register.
-90	Reference documents cannot be moved without indicating a location.
-94	No document pages are allowed.
-1001	Unable to find the specified value in the corresponding catalogue.
-1002	The key field is ambiguous.
-1003	It has been tried to set a read-only field.
-1004	No document list was specified.
-1005	The specified object is connected to a workflow process.
-1006	The requested function is not yet implemented in the present version.
-1007	An error occurred while the object definitions were read by ObjDefReader.
-1008	A requested file could not be accessed.
-1009	The object is in the trash can.
-1010	The document is in a portfolio.
-1011	Recurrence depth is too large, the action was cancelled.
-1012	The target register is a child register of the register which will be moved.
-1013	A system ID but no foreign ID have been specified.
-1014	No pages can be added to a reference document.
-1015	The document cannot reference to another document as it contains pages.
-1016	An error occurred while parsing the XML text.
-1017	An error occurred while validating the XML text.
-1018	The XML text is incomplete.
-1019	The specified owner could not be determined.
-1020	Invalid object type for this operation.
-1021	No cascading deletion allowed, as the object contains sub objects.
-1022	The reference could not be found.
-1023	The specified system field must not be manipulated by the user.
-1024	A job parameter is missing.
-1025	The specified parameter value is invalid.
-1026	Rights authorization cannot be bypassed by users.

-1027	More values than available columns were specified for the table field.
-1028	An error occurred while inserting note objects/texts.
-1029	An XML element is unknown.
-1030	An object type was not found on the DMS server.
-1031	No object field was found on the DMS server.
-1032	The job was cancelled by the user.
-1033	The specified condition is invalid.
-1034	An XML attribute is incorrect.
-1035	A required XML attribute is missing.
-1036	Only documents can be checked out.
-1037	The document has been checked in.
-1038	The document has been checked out.
-1039	The document has been checked out by another user.
-1040	The document has been checked out by another station.
-1041	The document cannot be checked out because it has no pages.
-1042	The document is not located in the workflow tray.
-1043	The document is located in the workflow tray.
-1044	The parent variant of the document could not be determined.
-1045	New variants can only be created for W-documents.
-1046	The document variant could not be determined.
-1047	The entered user data are ambiguous.
-1048	A competitive update of an object failed.
-1049	The search request was not found.
-1050	A copy of the object does already exist in the specified location.
-1051	The request format is not supported.
-1052	The cabinet name could not be read.
-1053	The document name could not be read.
-1054	The register name could not be read.
-1055	An expression has an invalid format.
-1056	The section name is invalid.
-1057	A general error occurred while reading the search request
-1058	Unknown request format
-1059	A new variant has been created but could not be set to 'active'.
-1060	The search request could not be processed.

-1061	No objects can be moved or linked within the system.
-1062	The location of the object could not be determined.
-1063	A document does already exist in the specified location.
-1064	It is not allowed to copy objects with key fields.
-1065	The entered password is incorrect.
-1066	New password is the same as the old password.
-1067	The full text search contains only words which are ignored.
-1068	There is insufficient memory.
-1069	Access to a system resource was denied.
-1070	The XMLImport job returns 'Error' instead the number of hits.

Cross Job Restrictions

The following descriptions are universal, they are therefore valid for all XML Import Jobs.

Date and Time Formats

Date fields can be imported in the formats listed below. This means that DD always stands for a day with two digits ('09' instead of '9'), MM represents a month with two digits and YY stands for a year with two digits while YYYY represents a year with four digits.

§ YYMMDD

§ YYYYMMDD

§ YYYY/MM/DD

Time fields can be imported in the format HH:MM:SS, with HH representing the hours with two-digits, MM for the minutes and SS for the seconds. Time stamps are basically imported in the format YYYY/MM/DD HH:MM:SS, with the date and time separated by blanks.

Special Field Types

§ A checkbox can only accept the values 0 or 1.

§ A radio button can be identified as field with the name of the first button (after the tab sequence). If the radio buttons are also correctly grouped with a group field, the field can also be identified with the name of the group field. Valid values for the corresponding buttons start with 0.

§ Field types like 'type of patient,' 'page,' 'gender,' and 'question' can contain the first letter as well as the value spelled out in full.

§ Decimal values can start with a plus or minus sign. The digits before and after the comma can be separated by a comma or a dot.

§ In text fields with several lines, single lines of the values can be separated using 'Carriage Return Line Feed'. In XML this represents the combination: &#13;&#10.

Additional Field Values

All fields can also get the value ZERO through the 'field_function' attribute. In this case, this means that the field is desired to be empty. Numeric fields and text fields can get the value of the object's internal ID through the same attribute. In this case, the field value will be replaced with the new or existing ID of the corresponding object.

The XML Error List

If an error occurred, a return value describing the nature of the possible error will be returned, however very often this is not a sufficient description of the exact error. Therefore all jobs generate an error list. The error list can be obtained with GetErrorList() after the job has been executed. This error list contains the exact description of the recently occurred errors. The errors described first are the most conclusive errors. The error list is in general also provided as XML in the 'DMSResult' return parameter. Alternatively, this XML can also be returned as a return file.

Example:

XML error list

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DMSData>
<Messages>
<Message Sourcename="oxjobdms">The field &gt;Table - Column2&lt; has
a catalog; the value &gt;7&lt; could not, however, be found
in the catalog</Message>
<Message Sourcename="oxjobdms">SAX error: Common exception</Message>
<Message Sourcename="oxjobdms">An error occurred during XML
import</Message>
</Messages>
</DMSData>
```

Besides, all errors which occurred on the server can be seen in the framework of the usual server logging.

DMS.CheckInDocument

Description:

This job checks in the specified document. The document is transferred to the server and deleted at its location (indicated in the File List parameter).

Parameter:

Flags (INT): 1 = the transferred document can be checked in by another station, otherwise 0

[ArchiveType] (INT): cabinet type

ObjectType (INT): Object type

ObjectID (INT): ID of the document

File list: path and name of the file to be checked in

Return values:

Info (STRING): if the document has been checked out by another user/station, the name will be returned

See also:

DMS.CheckOutDocument

DMS.CheckOutDocument

Description:

This job checks out the specified document. The document will be marked as checked out in the database. The actual document has to be fetched from the server with the job std.StoreInCache.

Parameter:

Flags (INT): 1 = the document will be checked out to 'external,' otherwise 0.

Number of hits (INT): cabinet type

ObjectType (INT): Object type

ObjectID (INT): ID of the document

Return values:

[Info] (STRING):if the document is already checked out, the name of the person checking out is returned

See also:

DMS.CheckInDocument, DMS.UndoCheckOutDocument

DMS.UndoCheckOutDocument

Description:

Use this job to undo the checkout of the specified document.

Parameter:

Flags (INT): 1 = checkout out can be undone by another station, otherwise 0.

Number of hits (INT) cabinet type

ObjectID (INT): Document ID

ObjectType (INT): Object type

Return values:

[Info] (STRING): if another user tries to undo the checkout, the name of the person checking out is returned (the same applies to the station)

DMS.GetXMLJobOptions

Description:

This job returns all options of the jobs XMLInsert, XMLUpdate, XMLMove and XMLDelete.

Return values:

JobOptions (BASE64): list of all job options in XML format

Example:

Structure of JobOptions

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DMSOptions>
<DMSOption optionstring="" description="" defaultvalue="" />
<DMSOption optionstring="" description="" defaultvalue="" />
.
.
.
</DMSOptions>
```

Note:

Detailed description of JobOptions

- optionstring: Name of the option
- description:description of the option
- defaultvalue: default value
  - SET: the option is set by default
  - NOT_SET: the option is not set by default
  - UNDEFINED: the option is not defined by default

See also:

The 'Options' Parameter

DMS.RestoreIndexdataVersion

Description:

This job restores the specified version from the index data history.

Parameter:

Flags (INT): not currently supported

ObjectID (INT): ID of the document whose index data is to be restored

Guid (STRING): GUID of the object version to be restored

DMS.XMLDelete

Description:

This job deletes the specified object. Folders, registers and documents can be deleted. The 'HARDDELETE' option offers to two ways to delete:

- 'Delete physically': the object is completely deleted, i.e. it is not moved to the trash can. If the object is a document all of its files will also be erased from the server.
- 'Delete to trash can': the object will not be physically erased but moved to the trash can from where it can be restored.

The object can contain sub objects (e.g. a folder contains documents). This object cannot be deleted without deleting all its sub objects. This is controlled by the 'DELETECASCADING' option.

It is not possible to delete objects which are assigned to a workflow process. If a document exists in several locations, the job usually will delete the document from all locations. However, it is also possible to delete the document from only one location. Therefore the parent attributes (register ID, register type or folder ID) have to be set in the object element. This specifies the location of the document which is to be deleted.

Parameter:

Flags (INT): general options for the job (see Flags)

Options (STRING): semicolon-separated job options (e.g. HARDDELETE=1 ;CHECKACCESS=0) (see Options)

XML (BASE64): contains object description in XML format (see The XML Parameter)

JobUserGUID (STRING): determines the user context (see The JobUserGUID Parameter)

Note:

The following XML examples always contain all tags and tag attributes which can be used for the respective actions. Of course, unnecessary tags and attributes can be dropped.

Example:

XML for deleting a folder

```xml
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType name="press archive" type="FOLDER" internal_name="" osguid=""
table="" id="-1">
<Object object_id="214"/>
</ObjectType>
</Archive>
</DMSData>
```

Example:

XML for deleting a register

```xml
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType type="REGISTER" name="Register" internal_name="" osguid=""
table="" id="-1">
<Object object_id="229"/>
</ObjectType>
</Archive>
</DMSData>
```

Example:

XML for deleting a document

```xml
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive name="press archive" id="-1" internal_name="" osguid="">
<ObjectType name="Word texts" maintype="4" cotype="0" type="DOCUMENT"
id="-1" internal_name="" osguid="" table="">
<Object object_id="214" variantparent_id="-1"/>
</ObjectType>
</Archive>
</DMSData>
```

DMS.XMLInsert

Description:

This job inserts an object into enaio®. A folder, register or document can be inserted. The return value 'ObjectID' is the ID of the inserted object or -1, if the job failed. If the object, which is inserted, is a document and if it contains files which have to be transferred to the archive server, the file list has to be completed with the corresponding file paths. Slides can also be passed. If more than one slide has to be passed, only the first slide from the list will be inserted into the archive server. In case of a document or register, the location of the new object has to be entered into the XML object element. If there has only been entered a register ID, the register type and the cabinet will be specified automatically. However, due to performance reasons it is recommended to make these specifications.

Parameter:

Flags (INT): general options for the job (see Flags)

Options (STRING): Semicolon-separated job options

(e.g. ARCHIVABLE=1;CHECKACCESS=0) (see Options)

XML (BASE64): contains object description in XML format (see The XML Parameter)

JobUserGUID (STRING): determines the user context (see The JobUserGUID Parameter)

[File list]: Path and name of the documents to be inserted

File_N: (STRING) file path n as an alternative to the file list

Return values:

ObjectID (INT): new object ID, if the job succeeds, otherwise -1

ObjectType (INT): object type, otherwise -1

[File list]: path and name of the XML file with the errors (see flags)

Note:

The following XML examples always contain all tags and tag attributes which can be used for the respective actions. Of course, unnecessary tags and attributes can be dropped.

Important:

If important attributes such as main type, register ID, register type or system are not used, they should be either completely omitted, or set to '0' or '-1' depending on the functionality.

Important:

The following example with the '<TableFields/>' tags only works if there is a table in the indexing mask of the cabinet.

Example:

XML for inserting a folder into a cabinet

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType table="" id="-1" name="press archive" internal_name=""
osguid="" type="FOLDER">
<Object>
<Fields>
<Field dbname="" name="specialist area" internal_name=""
osguid="">software development</Field>

<Field dbname="field2" system="" name="" internal_name=""
osguid="">test user</Field>
</Fields>
<TableFields>
<TableField dbname="table" system="" name="" internal_name="">
<Row>
<Field name="description" internal_name=""
dbname="">Documentation</Field>
</Row>
</TableField>
</TableFields>
<Remarks>
<RemarkText action="INSERT" color="WHITE">A note can be defined here.</RemarkText>
<RemarkObject action="INSERT" object_id="123" object_type="196616">Link
note</RemarkObject>
```

```
</Remarks>
</Object>
</ObjectType>
</Archive>
</DMSData>
```

Important:

The following example with the '<TableFields/>' tags only works if there is a table in the indexing mask of the register.

Example:

XML for inserting a register

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType table="" id="-1" name="year2004" internal_name="" osguid=""
type="REGISTER">
<Object folder_id="228">
<Fields>
<Field dbname="" system="" name="category" internal_name=""
osguid="">New development</Field>

<Field dbname="" system="" sortpos="" name="" internal_name=""
osguid="BDED8A3C99E64AD2A4ECBFDB586">Public</Field>

</Fields>
<TableFields>
<TableField dbname="" name="table" internal_name="">
<Row>
<Field name="topic" internal_name=""
dbname="">Document management</Field>
</Row>
</TableField>
</TableFields>
<Remarks>
<RemarkText action="INSERT" color="WHITE">A note on the register.</RemarkText>
<RemarkObject action="INSERT" object_id="234" object_type="196616">Link
note</RemarkObject>
</Remarks>
</Object>
</ObjectType>
</Archive>
</DMSData>
```

Important:

The following example with the '<TableFields/>' tags only works if there is a table in the indexing mask of the document.

Example:

XML for inserting a document

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType table="" id="-1" maintype="4" cotype="0" name="Word texts"
internal_name="" osguid="" type="DOCUMENT">
<Object object_id="-1" folder_id="" register_id="78" register_type="0"
variantparent_id="-1" maintype="0">
<Fields>
<Field dbname="" system="0" name="author" internal_name=""
```

```
osguid="">Test user</Field>
</Fields>
<MultiFields>
<MultiField dbname="" system="0" name="" internal_name=""
osguid="2AED8A3399EE778DS4ECBFDB582">
<Page id="1">
<Value>345</Value>
</Page>
<Page id="2">
<Value>123</Value>
</Page>
</MultiField>
</MultiFields>
<TableFields>
<TableField dbname="" name="table" internal_name="" osguid="">
<Row>
<Field name="Team" internal_name=""
dbname="">Development</Field>
<Field name="" internal_name="feld2" dbname="">
Status: released</Field>
</Row>
</TableField>
<TableField dbname="" name="" internal_name=""
osguid="AAED8A3C99EED78DS4ECBFDB586">
<Row>
<Field name="" internal_name="" dbname="fd1">Year 2004</Field>
</Row>
</TableField>
</TableFields>
<Remarks>
<RemarkText action="INSERT" color="BLUE">A note on the document.</RemarkText>
<RemarkObject action="INSERT" object_id="432" object_type="196616">Link
note</RemarkObject>
</Remarks>
</Object>
</ObjectType>
</Archive>
</DMSData>
```

DMS.XMLMove

Description:

This job moves an object. You can move a register or a document. Parent attributes (register ID, register type and folder ID) are attributes indicating the new location. The folder ID should be indicated even if the object's new location is a register.

If a register is moved, it will be moved to the new location with all its sub objects. In addition, the location of reference documents has to be indicated as 'sourceparent_id' attribute in the object tag.

Documents can be moved to the user tray. Therefore no locations have to be indicated in the XML object element, but the option 'WFTOUSERTRAY' has to be activated. The document which will be moved must have been in the workflow tray before, documents from folders and registers cannot be moved.

Documents can be moved from the user tray or the workflow tray. These documents can then be moved to a register or folder. The job automatically recognizes whether the object is located in a register, folder, workflow tray or user tray.

Parameter:

Flags (INT): general options for the job (see Flags)

Options (STRING): Semicolon-separated job options

(e.g. ARCHIVABLE=1;CHECKACCESS=0) (see Options)

XML (BASE64): contains object description in XML format (see The XML Parameter)

JobUserGUID (STRING): determines the user context (see The JobUserGUID Parameter)

Note:

The following XML examples always contain all tags and tag attributes which can be used for the respective actions. Of course, unnecessary tags and attributes can be dropped.

Important:

If important attributes such as ID are not required in the tags, either leave them out completely or set them to '0' or '-1' depending on the function.

Example:

XML for moving a register

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType type="REGISTER" name="Register" internal_name="" osguid=""
table="" id="-1">
<Object object_id="28" folder_id="58" register_id="-1"
register_type="-1"/>
</ObjectType>
</Archive>
</DMSData>
```

Example:

XML for moving a document

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType type="DOCUMENT" name="Word texts" maintype="4" cotype="0"
internal_name="" osguid="" table=""
id="-1">
<Object object_id="248" register_id="228" register_type="6488064"
folder_id="58"/>
</ObjectType>
</Archive>
</DMSData>
```

DMS.XMLCopy

Description:

This job copies objects; folders, registers and documents can be copied. If a register or folder is copied, you can use the 'COPYCASCADING' option to determine whether all contained objects are copied in a cascading way.

If a document is copied, it can be copied in two ways. Either a new document is generated or a reference to the original document is made at the new location. The document then has two locations, but only one index dataset. To link the document this way, the 'LINKDOCUMENT' job option has to be selected. Source and target location have to be different for such a reference copy. This option is also valid for all documents which are copied with the 'COPYCASCADING' option.

Parent attributes (register ID, register type and folder ID) are attributes indicating the new location. The folder ID should be indicated even if the object's new location is a register. The register type can be determined by the executor, however, due to performance reasons it is recommended to make these specifications.

Parameter:

Flags (INT): general options for the job (see Flags)

Options (STRING): Semicolon-separated job options

(e.g. LINKDOCUMENT=1;CHECKACCESS=0) (see Options)

XML (BASE64): contains object description in XML format (see The XML Parameter)

JobUserGUID (STRING): determines the user context (see The JobUserGUID Parameter)

File_N: (STRING) file path n as an alternative to the file list

Note:

The following XML examples always contain all tags and tag attributes which can be used for the respective actions. Of course, unnecessary tags and attributes can be dropped.

Example:

XML for creating a new document location

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType type="DOCUMENT" name="Word texts" internal_name="" osguid=""
table="" id="-1">
<Object object_id="28" folder_id="58" register_id="-1"
register_type="-1"/>
</ObjectType>
</Archive>
</DMSData>
```

DMS.XMLUpdate

Description:

This job modifies the specified object. A folder, register or document can be updated. File paths can be passed to the file list if the object which will be inserted is a document. The 'REPLACEFILES' option determines whether the files will replace existing files or whether they will be appended to them.

Documents can be changed into reference documents as long as they do not contain files. This is done by setting the 'Foreign ID' system field (and possibly the system ID). Similar rules are true for reference documents: they can be changed back to documents if the foreign ID (and possibly the system ID) are removed. This is done by setting the system fields using the attribute 'field_function' and the value 'NULL'.

The owner can be changed by specifying the 'Owner' system field. The owner cannot be queried with GUID; the value has to be a user name.

If the 'concurrency_timestamp' attribute is filled with a value in the object tag, it is checked whether the dataset has been changed since the last call, and an error is returned. Thus, it can be prevented that changes between the call and the desired change get lost.

By default, table control rows are appended to the existing data. If in the <Row> element, the 'line' attribute is set with a row number, the respective row will be updated. Numbering of the rows is 1-

based. If the REPLACETABLEFIELDS job option is activated, all existing rows will be completely replaced by the new data.

The job option REPLACEREMARKS specifies whether, when notes or links are created, all existing notes and all existing links are deleted or the notes and links are added (default). 'RemarkText' and 'RemarkObject' have the attribute 'action' with the following values: INSERT (default), DELETE, UPDATE, UPDATE_TEXT, UPDATE_COLOR. Existing notes are specified via the ID (remark_id), links via object ID and object type ID of the linked object.

Parameter:

Flags (INT): general options for the job (see Flags)

Options (STRING): Semicolon-separated job options

(e.g. ARCHIVABLE=1;CHECKACCESS=0) (see Options)

XML (BASE64): contains object description in XML format (see The XML Parameter)

JobUserGUID (STRING): determines the user context (see The JobUserGUID Parameter)

[File list]: path and name of the modified document

File_N: (STRING) file path n as an alternative to the file list

Note:

The following XML examples always contain all tags and tag attributes which can be used for the respective actions. Of course, unnecessary tags and attributes can be dropped.

Example:

XML for changing a folder

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType table="" id="-1" name="press archive" internal_name=""
osguid="" type="FOLDER">
<Object object_id="54">
<Fields>
<Field dbname="" system="0" name="specialist area"
internal_name="" osguid="">Software development</Field>
<Field dbname="field2" system="0" internal_name=""
osguid="">test user</Field>
</Fields>
<TableFields>
<TableField dbname="table" system="0" name="" internal_name="">
<Row>
<Field name="description" internal_name=""
dbname="">Documentation</Field>
</Row>
</TableField>
</TableFields>
<Remarks>
<RemarkText action="INSERT" color="WHITE">A note on the folder.</RemarkText>
<RemarkObject action="INSERT" object_id="123" object_type="196616">Link
text</RemarkObject>
</Remarks>
</Object>
</ObjectType>
</Archive>
</DMSData>
```

Example:

XML for changing a register

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType table="" id="-1" name="year2004" internal_name="" osguid=""
type="REGISTER">
<Object object_id ="312" folder_id="228" register_id="-1"
register_type="-1">
<Fields>
<Field dbname="" system="0" name="category"
osguid="">New development</Field>
<Field dbname="" system="0" name="" internal_name=""
osguid="BDED8A3C99E64AD2A4ECBFDB586">Public</Field>
</Fields>
<TableFields>
<TableField dbname="" name="table" internal_name="">
<Row>
<Field name="topic" internal_name=""
dbname="">Document management</Field>
</Row>
</TableField>
</TableFields>
<Remarks>
<RemarkText action="INSERT" color="WHITE">A note on the register.</RemarkText>
<RemarkObject action="INSERT" object_id="234" object_type="196616">Link
text</RemarkObject>
</Remarks>
</Object>
</ObjectType>
</Archive>
</DMSData>
```

Example:

XML for changing a document

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType table="" id="-1" maintype="4" cotype="0" name="Word texts"
internal_name="" osguid="" type="DOCUMENT">

<Object object_id="221" folder_id="" register_id="-1" register_type=""
variantparent_id="-1" maintype="0">
<Fields>
<Field dbname="" system="0" name="author" internal_name=""
osguid="">Test user</Field>
</Fields>
<MultiFields>
<MultiField dbname="" system="0" name="" internal_name=""
osguid="2AED8A3399EE778DS4ECBFDB582">
<Page id="1">
<Value>345</Value>
</Page>
<Page id="2">
<Value>123</Value>
</Page>
</MultiField>
</MultiFields>
<TableFields>
<TableField dbname="" name="table" internal_name="" osguid="">
<Row>
<Field name="Team" internal_name=""
dbname="">Development</Field>
```

```
<Field name="" internal_name="field2"
dbname="">Status: released</Field>
</Row>
</TableField>
<TableField dbname="" name="" internal_name=""
osguid="AAED8A3C99EED78DS4ECBFDB586">
<Row>
<Field name="" internal_name="" dbname="fd1">Year 2004</Field>
</Row>
</TableField>
</TableFields>
<Remarks>
<RemarkText action="INSERT" color="BLUE">A note on the document.</RemarkText>
<RemarkObject action="INSERT" object_id="432" object_type="196616">Link
text</RemarkObject>
</Remarks>
</Object>
</ObjectType>
</Archive>
</DMSData>
```

DMS.XMLImport

Description:

This job allows to insert or update an object depending on the result of the previous search.

Parameter:

Flags (INT): general options for the job (see Flags)

Options (STRING): Semicolon-separated job options

(e.g. ARCHIVABLE=1;CHECKACCESS=0) (see Options)

XML (BASE64): contains the object description in XML format (see The XML Parameter) with the <SearchFields> tag extension. Refer to the chart below.

JobUserGUID (STRING): determines the user context (see The JobUserGUID Parameter)

[Action0]: Action when no hit is returned (see action table)

[Action1]: Action when a hit is returned (see action table)

[ActionM]: Action when multiple hits are returned (see action table)

[File list]: path and name of the documents to be inserted

Return values:

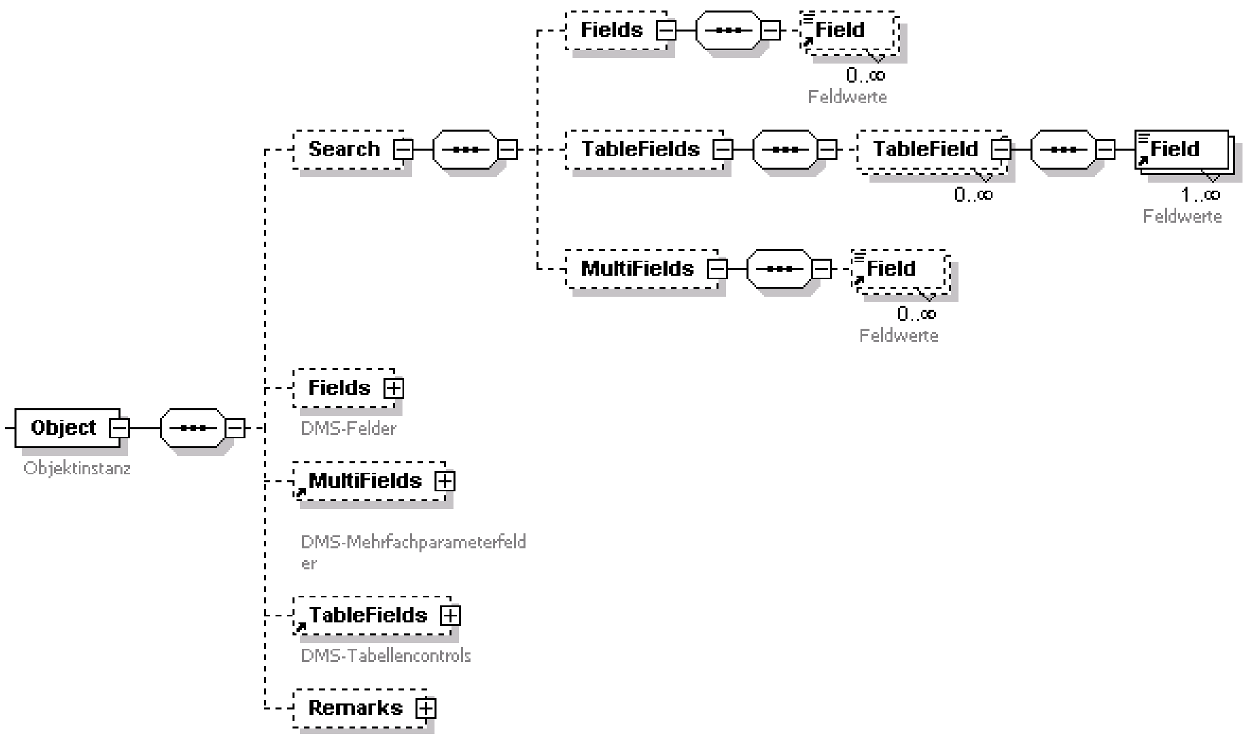
ObjectID (INT): new object ID, if the job succeeds, otherwise -1

ObjectType (INT): object type, otherwise -1

Hits (INT): number of search hits

Action: executed action. Possible values are UPDATE, INSERT, NONE, ERROR

[File list]: path and name of the XML file with the errors (see flags)



Extension of the XML format:

Action table

Search result [Hits]	**Parameter name**	**Possible parameter value**	**Explanation**
0	Action0	INSERT [Default] NONE ERROR	Insert. See DMS.XMLInsert Execute no action Create error message

1	Action1	NONE
		UPDATE [Default]
		INSERT
			Execute no action
			Update object [Default]. See DMS.XMLUpdate
			Insert at location of the found object. See DMS.XMLInsert

>1	ActionM	NONE
		UPDATE
		INSERT
		ERROR [DEFAULT]
			Execute no action
			Only update the first object. See DMS.XMLUpdate
			Insert at location of the first found object. See DMS.XMLInsert
			Generate error message

If the object's location is specified or limited, the location will be used for the search as well as for inserting. The search can also be used to determine the location. If no location is specified and no object is found, it is not possible to insert a register or document. In this case, an error message will be generated .

If no search fields are specified, the object will be inserted.

Example:

In the following example, within an address element, the telephone number of a contact person will be changed.

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive name="addresses">
<ObjectType name="addresses">
<Object>
<Search>
<Fields>
<Field name="contact person:">Schaumer</Field>
<Field name="first name:">Harald</Field>
</Fields>
</Search>
<Fields>
<Field name="Telephone:">0815-12345</Field>
</Fields>
</Object>
</ObjectType>
</Archive>
</DMSData>
```

DMS.XMLUnknownToKnown

Description:

This job changes a typeless document into a document with a type. Typeless documents are located either in the user tray or in the workflow tray. The 'INWFTRAY' option or the 'INUSERTRAY' option should be activated in the job, as should the 'ISTYPELESS' option. The type which is intended for the document is specified in the 'ObjectType' tag and the ID of the typeless document is the 'object_id' attribute in the 'Object' tag.

Parameter:

Flags (INT): general options for the job (see Flags)

Options (STRING): Semicolon-separated job options

(e.g. INWFTRAY=1;CHECKACCESS=0) (see Options)

XML (BASE64): contains object description in XML format (see The XML Parameter)

JobUserGUID (STRING): determines the user context (see The JobUserGUID Parameter)

Note:

The following XML examples always contain all tags and tag attributes which can be used for the respective actions. Of course, unnecessary tags and attributes can be dropped.

Example:

XML for typing an typeless document.

```
<?xml version="1.0" encoding="UTF-8"?>
<DMSData>
<Archive id="-1" name="press archive" internal_name="" osguid="">
<ObjectType table="" id="-1" name="press archive" internal_name=""
osguid="" type="FOLDER">
<Object object_id="54">
<Fields>
<Field dbname="" system="" name="specialist area"
internal_name="" osguid="">Software development</Field>
```

```
<Field dbname="field2" system="0" internal_name=""
osguid="">test user</Field>
</Fields>
<TableFields>
<TableField dbname="table" system="0" name="" internal_name="">
<Row>
<Field name="description" internal_name=""
dbname="">Documentation</Field>
</Row>
</TableField>
</TableFields>
<Remarks>
<RemarkText action="INSERT" color="WHITE">A note on the folder.</RemarkText>
<RemarkObject action="INSERT" object_id="123" object_type="196616">Link
note</RemarkObject>
</Remarks>
</Object>
</ObjectType>
</Archive>
</DMSData>
```

XML Export (Search)

Index data and DMS object information can be searched with the export function of the DMS Executor.

§ DMS.GetObjDef

§ DMS.GetObjectTypeByID

§ DMS.GetResultList

§ DMS.GetObjectDetails

§ DMS.GetDeletedObjects

§ DMS.GetLinkedObjects

§ DMS.SelectDistinctFieldValues

§ DMS.GetUserTrayObjects

§ DMS.ExecuteStoredQuery

§ DMS.GetStoredQuery

§ DMS.AddStoredQuery

§ DMS.UpdateStoredQuery

§ DMS.RemoveStoredQuery

§ DMS.ConvertQuery

§ DMS.GetObjectHistory

§ DMS.GetShadowData

§ DMS.GetObjectsByDigest

DMS.GetObjDef

Description:

This job returns all object definitions in XML format.

Parameter:

Flags (INT): options for this job

- § 1 =the object definition is returned by the sRet parameter or otherwise as XML file
- § 2 =visible cabinets and objects without field information are returned Visibility is checked based on the 'Security system' object flag.
- § 4 =images of the object definition are returned Base64-encoded in the XML return.
- § 4096 = the XML document is encoded as UTF-8, otherwise UTF-16

Return values:

[FileCount] (INT): 1 = Output file was successfully created, otherwise 0

[sRet] (BASE64): Object definition in XML format

[File list]: path and name of the XML file containing the object definition

Example:

Object definition in XML format

```
<?xml version="1.0" encoding="UTF-8"?>
<asobjdef created="" version="4.00">
<OS4x/>
<languages>
<language lang_id="" active="" name=""/>
</languages>
<cabinet cotype="" name="" internal="">
<object compressionflags="" historyflags="" maintype="" cotype=""
iconid="" name="" internal="" os_guid="" extablename="" tablename=""
no_dias="0" reference="0" fulltext="0" apply_security="0"
multidoc="0">
<names>
<name lang_id="" tooltip=""/>
</names>
<fields>
<field classstring="" name="" fieldname="" os_guid="" init=""
taborder="" tooltip="" prnalias="" internal="">
<names>
<name lang_id="" tooltip=""/>
</names>
<ids/>
<flags align="left" dt="A" flags="" flags1="" flags2=""
input_length="0" readonly="supervisor" multifield="0"
zeropad="0" control_type="edit" datatype="text"
constraints="none" required="0" crosscheck="0" color=""
case="0"/>
<field_pos bottom="0" left="0" right="0" top="0"/>
<field_pos bottom="0" left="0" right="0" top="0"/>
<init func="0" init_type="const"/>
<list addon32="" multiselection="0" order="0">
<rawdata/>
<extra/>
<row/>
</list>
<page/>
</field>
</fields>
<ids oid="" pid="" vid=""/>
<frame bottom="0" left="0" right="0" top="0"/>
<multiframe height="" width=""/>
</object>
</cabinet>
</asobjdef>
```

DMS.GetObjectTypeByID

Description:

This job can be used to determine the type of an object instance.

Parameter:

Flags (INT): reserved. Has to be 0.

ObjectID (INT): ID of the object instance

Return values:

ObjectType (INT): Object type

The object type generally consists of a main type (highword) and a sub type (lowword). The following table provides an overview over the supported object types:

Object type	Main type	Sub type
Folder	0	>=0
Register	99	>=0
Document	1 = Grayscale 2 = B/W module 3 = Color 4 = Windows 5 = Multimedia 6 = e-mail 7 = XML 8 = Container	>=0
Typeless document in the user tray	200	1-7 (see document main type)
Typless document in the workflow tray	300	1-7 (see document main type)
Portfolio	203	0
Note	32767	1-4

DMS.GetResultList

Description:

This job searches DMS objects which match the search request. DMS target objects and fields are specified in an XML request document. OS names, internal names, GUIDs and database names can be used in the XML request document.

General properties of the search request can be set using attributes. These attributes can be set as job parameters or they can be defined as XML attributes in the root element of an XML search request with XML attributes prioritized.

Parameter:

Flags (INT): flags to control the output format

§ 0x00000010 = XML result is returned as a file, otherwise as buffer

XML (BASE64): Search request in XML format (see Detailed Description)

[Encoding] (STRING):coding of the result document. Permitted values: UTF-8 and UTF-16. Default value is UTF-16.

[Offset] (INT): Offset for first returned dataset. Default=0 (see Browsing Hit Lists)

[PageSize] (INT): number of datasets to be returned; -1 = all (default) (see Browsing Hit Lists)

[MaxHits] (INT): maximum number of hits to be determined per object type; -1 = all (default), 0 = do not determine (A high value can impact performance.) (see Browsing Hit Lists)

[Sql] (INT): 1 = Sql statements are written to the result document (only LOL); default=0

[Rights] (INT): 1 = additional access rights (open/delete/write) are determined for each object instance (HOL) default=0 (see Rights)

[ObjectInserts] (INT): 1 = number of insertable objects for each object type. Default=0 (see Rights)

[DateFormat] (STRING): indicates how returned date information is to be formatted (see Date Formats)

[RegisterContext] (INT): 0 = When register clauses are specified, only objects in registers are requested (HOL) 1=Default (see General Search Behavior)

[ItemDelimiter] (STRING): Separator between the values for linear list as simple text file (see Output Formats)

[RequestType] (STRING): 'LOL', 'INF' or 'HOL' possible

[OutputFormat] (STRING): depending on the search request type 'LOL', 'INF', 'TXT' or 'HOL' possible. (see Output Formats)

[Baseparams] (INT): 1 = basic parameters are returned, in the case of HOL as own XML structure (see Basic Parameters)

[FileInfo] (INT): 1 = file size in bytes, file extension and MIME type of document files are determined (see File Information)

[FollowDocLink] (INT): 0 = if activated, the file information of the linked document (green arrow links) is returned. This option will be evaluated only if [FileInfo=1] and can affect the performance.

[Variants] (INT): 1 =In the case of a document, the variant tree is returned if several variants of this document exist (see Document Variants)

[Status] (INT): 1 = object status is delivered (including links and specially for documents: module type, check out and archiving status) (see Object Status)

[CheckParams] (INT): 0 = if no value for a search request parameter, conditions referencing this parameter are ignored. This is the default value. 1 = an error message will be returned if a referenced parameter was not defined

[FieldSchema] (STRING): 'MIN' (only object ID), 'ALL' (all index fields) or 'DEF' (user defined) possible. This setting can be overwritten within the XML request docent by the 'field_schema' attribute in the <Fields>, <ParentObjects> or <ChildObjects> elements.

[AutoStar] (INT): Specifies whether text fields are to be automatically appended or prepended in the case of conditions. 1=asterisk is prepended, 2=asterisk is appended, 3=asterisk is prepended and appended

[QueryLanguage] (INT): Code of the language used for the DMS names in the search request. Default language=0

[OutputLanguage] (INT): Code of the language to be used for the DMS names of the result document. Default language=0

[DisableSearchGroups] (INT): If this parameter is set to 1, search groups are not taken into account, i.e. search conditions are only set for defined field. If the parameter is set to 0 (default), the search conditions refer to all fields of the corresponding search group.

[JobUserGUID] (STRING): determines the user context (see The JobUserGUID Parameter)

[GarbageMode] (INT): 1=only objects from the trash can are taken into account. 0=objects from the trash can are not taken into account.

Return values:

Count (INT): number of returned datasets

TotalHits (INT): number of available hits

[FileCount] (INT): only one file is returned

[XML] (BASE64): hit list in XML format

[File list]: name and path of the XML file containing the hit list

Detailed Description

In principle, a search request consists of a specification of the DMS object, a selection of return fields and the specification of search conditions. There is also the possibility of requesting location information, carrying out full text or basic parameter searches, creating links to objects in other cabinets, or setting parameters for requests.

§ DMS – Search Request Types

§ DMS – Result Formats

§ General Query Behavior

§ Creating Queries

§ Browsing Hit Lists

§ DMS Reference

DMS Search Request Types

In the case of search requests, a basic distinction is made between linear and hierarchical requests. A detailed search request represents a special case.

Linear search requests/linear object lists (LOL)

Simple index data of objects of a certain type can be requested with linear search requests. Table control elements and multiple parameter fields are excluded. If documents or registers are searched, fields of the direct parent register and the folder can be searched as well.

Hierarchical search requests/hierarchical object lists (HOL)

Hierarchical search requests can return the contents of table control elements and multiple parameter fields as well as child elements, e.g. registers and documents within the requested folder. Object properties such as document variants, access rights, and notes can be requested additionally.

Next to the functional differences between a linear and a hierarchical search request there is an important technical difference. Hit lists for linear search requests can be obtained with a set of database queries. Hierarchical search requests require that in addition to every hit at least one extra database query is executed for every object instance. Every searched object property increases the number of database queries for each object instance.

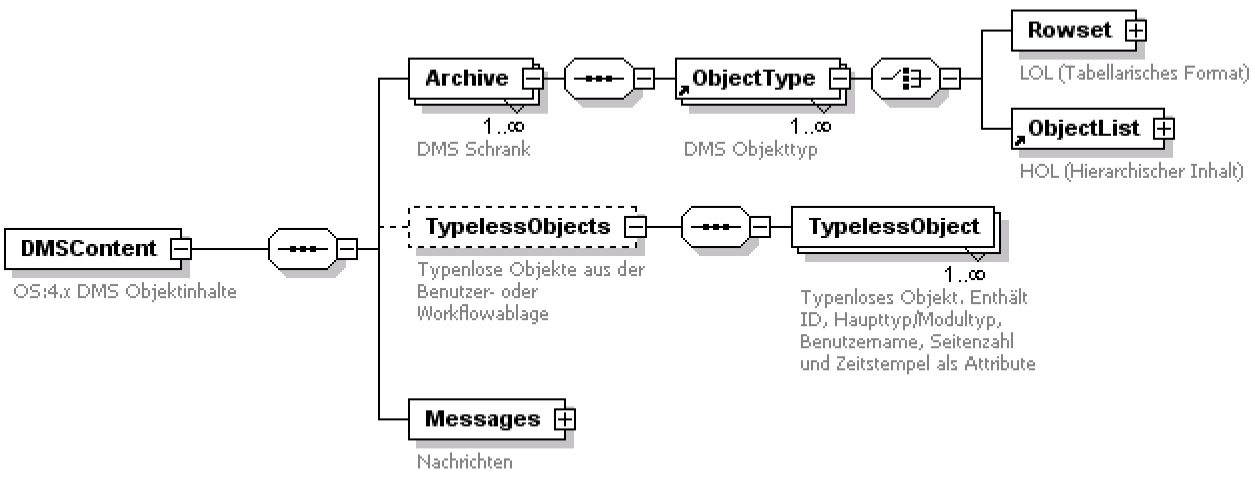
If no table control elements, multiple parameters, child elements or additional properties (document variants, access rights, notes) are necessary, due to performance reasons, it is recommended to carry out only linear search requests.

Mixed search requests (MIX)

Due to the above-mentioned substantial differences in performance, there is a third search type, which is called mixed search request. It is a mix of a linear and a hierarchical search request which can be used to query detailed information concerning folders, registers and information about direct child objects. A database query is made for each object type of child objects. The implementation scenario corresponds, for example, to opening a folder to obtain its complete index data and object properties and to get an overview of all registers and documents located in this folder.

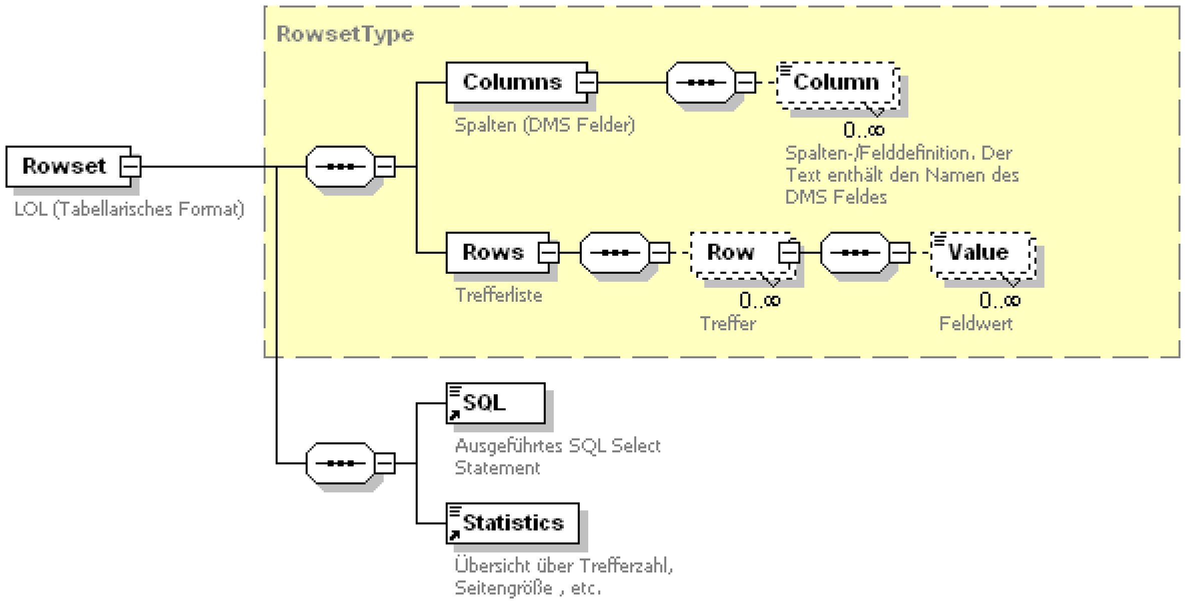
DMS - Result Formats

The XML format of the search results always starts with the <DMSContent> element. It contains a 'format' attribute which can have the values LOL for linear object list, HOL for hierarchical object list and MIXED for mixed hit lists.


DMSContent
OS:4.x DMS Objektinhalte
Archive
1..∞
DMS Schrank
ObjectType
1..∞
DMS Objekttyp
Rowset
LOL (Tabellarisches Format)
ObjectList
HOL (Hierarchischer Inhalt)
TypelessObjects
Typenlose Objekte aus der Benutzer- oder Workflowablage
TypelessObject
1..∞
Typenloses Objekt. Enthält ID, Haupttyp/Modultyp, Benutzername, Seitenzahl und Zeitstempel als Attribute
Messages
Nachrichten

Linear object list (LOL)

Linear object lists (LOL) They are either structured in a table form, i.e. the table header is described first (<Columns>) and all hit rows (<Rows>) are listed below.

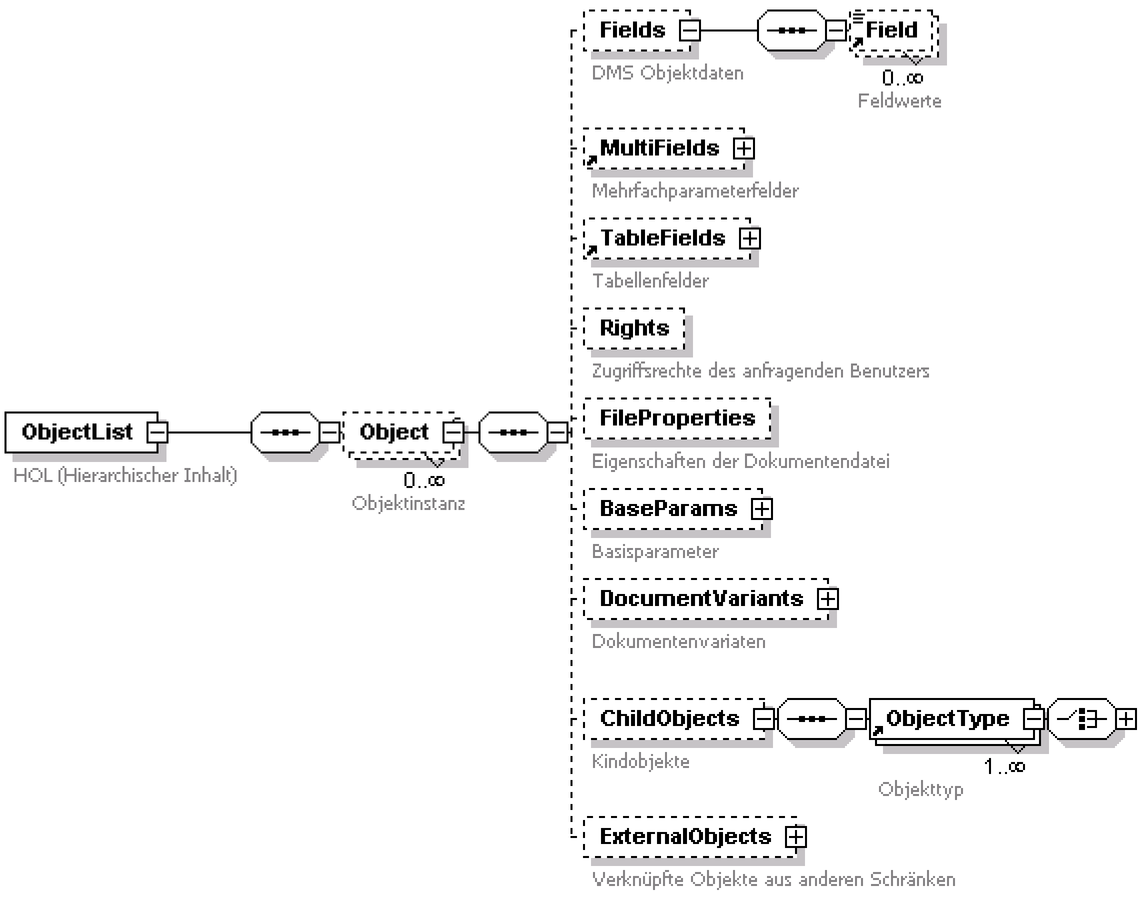

RowsetType
Rowset
LOL (Tabellarisches Format)
Columns
Spalten (DMS Felder)
Column
0..∞
Spalten-/Felddefinition. Der Text enthält den Namen des DMS Feldes
Rows
Trefferliste
Row
0..∞
Treffer
Value
0..∞
Feldwert
SQL
Ausgeführtes SQL Select Statement
Statistics
Übersicht über Trefferzahl, Seitengröße , etc.

Example:

```
<?xml version="1.0" encoding="UTF-16" standalone="yes" ?>
<DMSContent format="LOL" version="4.50.582.4137" timestamp="2004-05-
24T15:18:19" user="ROOT" station="OSEPA">
<Archive name="Patient" id="5" osguid="04C7E64C981C456A9D1F5B12D188A752">
<ObjectType name="Patient" id="5" osguid="04C7E64C981C456A9D1F5B12D188A752"
type="FOLDER" table="root6">
<Rowset>
<Columns>
<Column object="Patient" type="FOLDER" name="Name" datatype="TEXT"
dbname="field2" ostype="X" size="50">Name</Column>
<Column object="Patient" type="FOLDER" name="First name" datatype="TEXT"
dbname="field3" ostype="X" size="50">first name</Column>
<Column object="Patient" type="FOLDER" name="Place" datatype="TEXT"
dbname="field10" ostype="X" size="50">Place</Column>
</Columns>
<Rows>
D:\Testdata\VirtPC\Queries\HOL\tmp0002.xml - #      <Row id="38503">
<Value>Abold</Value>
<Value>Beate</Value>
<Value>Berlin</Value>
</Row>
<Row id="38003">
<Value>Abold</Value>
<Value>Christina</Value>
<Value>Berlin</Value>
</Row>
</Rows>
</Rowset>
<Statistics startpos="0" pagesize="2" total_hits="530" />
</ObjectType>
</Archive>
<Messages/>
</DMSContent>
```

Hierarchical object list (HOL)

Hierarchical hit lists are introduced by the <ObjectList> element. Field definition and field value are written for every hit object. Hierarchical hit lists can depict any level of hierarchy with the <ChildObjects> element.



Example:

Hierarchical object list

```
<DMSContent format="HOL" version="4.50.582.4137" timestamp="2004-05-
24T15:29:03" user="ROOT" station="OSEPA">
<Archive name="Patient" id="5"   osguid="04C7E64C981C456A1F5B12D188A752">
<!—Folder -->
<ObjectType name="Patient" id="5"
osguid="04C7E64C981C456A9D1F5B12D188A752" type="FOLDER" table="root6">
<!—Folder list -->
<ObjectList>
<Object id="37751">
<Fields>
<Field name="PatientID" datatype="TEXT" dbname="field1" ostype="X"
size="20">777777</Field>
<Field name="Name" datatype="TEXT" dbname="field2" ostype="X"
size="50">Sandmann</Field>
<Field name="First name" datatype="TEXT" dbname="field3" ostype="X"
size="50">Sandor</Field>
</Fields>
<!—Child objects of the folder à
<ChildObjects>
<!—Register -->
<ObjectType name="visit" id="6488064"
osguid="A90F043941488DB43B69CBDA" type="REGISTER" table="register1">
<ObjectList>
<Object id="3226">
<Fields>
<Field name="first name" datatype="TEXT" dbname="field62" ostype="X"
size="20">987654</Field>
<Field name="Begin" datatype="DATE" dbname="date1" ostype="D"
```

```
size="10">2003/10/01</Field>
<Field name="end" datatype="DATE" dbname="date2" ostype="D"
size="10" />
</Fields>
<-- Child objects of the register -->
<ChildObjects>
<!–Document type 'doctor's letter' -->
<ObjectType name="doctor's letter" id="262273" osguid="14799BCD39A920AF3"
type="DOCUMENT" module="WINDOWS"  table="object162">
<ObjectList>
<Object id="37744">
<Fields>
<Field name="begin" datatype="DATE" dbname="date1" ostype="D"
size="10">2003/11/12</Field>
<Field name="creat. doctor" datatype="TEXT" dbname="field1"
ostype="X" size="50">DEMO</Field>
<Field name="type" datatype="TEXT" dbname="field3" ostype="X"
size="20">doctor's letter operative</Field>
<Field name="status" datatype="INTEGER" dbname="count2"
ostype="9" size="1" />
</Fields>
</Object>
</ObjectList>
<Statistics startpos="0" pagesize="5" total_hits="1" />
</ObjectType>
</ChildObjects>
</Object>
</ObjectList>
<Statistics startpos="0" pagesize="5" total_hits="1" />
</ObjectType>
</ChildObjects>
</Object>
</ObjectList>
<Statistics startpos="0" pagesize="5" total_hits="1" />
</ObjectType>
</Archive>
<Messages />
</DMSContent>
```

Appearance of field values

Field values are generally displayed as XML element text. However, for some field types the value in the database is different from the displayed value, for example for:

§ Time

§ Timestamp

§ Radio buttons

§ System field owner

§ Trees

§ Lists

§ Special values like: direction and question etc.

For these field types, the database value is returned as 'value' attribute while the displayed text is contained in the XML element text.

LOL example:

```
<Value value="M">male</Value>
```

HOL example:

```
<Field value="M" name="gender" datatype="TEXT" dbname="field7"
ostype="X" size="1">male</Field>
```

The <Statistics> element

At the end of any hit list is a <Statistics> element which contains data about the number of results. This information is especially important for Browsing Hit Lists.

```
<Statistics startpos="20" pagesize="20" total_hits="50" />
```

The <Messages> element

Information on invalid searches is found in the <Message> elements.

Example:

```
<Messages>
<Message faultcode="-2116351928" sourcecode="475">the full text request could not
be carried out because a search text was not indicated.
</Message>
</Messages>
```

Combination of search types and output formats

The default output format for linear search requests is the above-mentioned LOL-XML format. It is also possible to query linear hit lists in a simple text format or in the HOL-XML format. This is done by using the 'OutputFormat' parameter.

With the text format, the values are separated by a delimiter which can be specified in the 'ItemDelimiter' parameter.

		Output format			
		TXT	LOL	MIX	HOL
Search request type	LOL	+	+	-	+
	MIX		-	+	+
	HOL		-	-	+

General Search Behavior

For a search, search conditions can be laid down for one or multiple object types. For example for a document search request, conditions can be set for a register and a folder.

A special search behavior has been set for the enaio® client. If search requests contain register conditions – set by the user or the rights system – the search behavior for search requests in enaio® looks as follows:

Rule for documents:

Search for all documents contained in a register which fulfills the search criteria and for all documents which cannot be found in a register.

Rule for folders:

Search for all folders contained in a register which fulfills the search criteria and for all folders which do not contain a register.

In the 'GetResultList' job, the behavior can be controlled with the 'RegisterContext' job parameter and the 'registercontext' XML attribute respectively. If the parameter is not indicated or if it has the value 1, the search behavior will be active, if the value is 0, it will be deactivated.

Create Search Requests

To create a search requests, the DMS objects and fields to be searched will have to be defined at first. DMS objects and fields can be identified using

§ descriptive names ('name' attribute)

§ internal names ('internal_name' attribute)

§ OSGUID ('osguid' attribute)

§ database table names or column names ('table' or 'dbname' attribute)

Example of a simple query

All folders of the 'Patient' cabinet are requested with the following search request. Since object names within a cabinet do not have to be unambiguous – e.g. a document type can have the same name as a folder type – the object type can be defined using the 'type' attribute. Valid values are 'FOLDER', 'REGISTER' and 'DOCUMENT'.

By assigning the value 'ALL' to the 'field_schema' <Fields> attribute, the DMS executor is instructed to return all DMS fields of the queried object, i.e. in this case the patient folder.

```
<?xml version="1.0" encoding="UTF-8" ?>
<DMSQuery>
<Archive name="Patient">
<ObjectType name="Patient" type="FOLDER" >
<Fields field_schema="ALL"/>
</ObjectType>
</Archive>
</DMSQuery>
```

Define fields

If not all index data are required, only the intended fields can be specified. The 'field_schema' fields attribute must then be set to 'DEF'.

```
<?xml version="1.0" encoding="UTF-8" ?>
<DMSQuery>
<Archive name="Patient">
<ObjectType name="Patient" type="FOLDER" >
<Fields field_schema="DEF">
<Field name="Name"/>
<Field name="First name"/>
<Field name="ZIP code / city"/>
<Field name="Place"/>
</Fields>
</ObjectType>
</Archive>
</DMSQuery>
```

Radio buttons

Radio buttons play a special role as they represent a group of fields. If such a group contains a static group field, it can be used as a search field. Apart from that, any single list box can be selected to represent the group.

System fields

System fields can also be queried with the <Field> element. The 'system' attribute therefore has to be set to 1. System fields can be indicated by an internal name, a GUID or their database field name. An overview of the available system fields can be found in DMS Reference. In the following example, the number of document files is queried in addition to all index data of the 'Images' object.

```
<?xml version="1.0" encoding="UTF-8" ?>
<DMSQuery>
<Archive name="Patient">
<ObjectType name="Images" type="DOCUMENT" >
<Fields field_schema="ALL"/>
<Field internal_name="OBJECT_COUNT" system="1"/>
</Fields>
</ObjectType>
</Archive>
</DMSQuery>
```

Sorting

In order to sort the hit lists of a LOL search request according to certain fields, the 'sortpos' attribute has to be set to a value greater than 0. Additionally, with the 'sortorder' attribute, the results can be sorted in an ascending (ASC value) or descending (DESC value) order. If the 'sortorder' attribute is not indicated, the results will be displayed in an ascending order unless the 'sortorder' attribute has been explicitly set for a field with higher sorting priority. In such a case, the value will also be assigned to the following fields without this attribute.

If the 'sortpos' attribute has been set for several fields, the fields with a lower 'sortpos' value will have higher priority.

The following search request queries all registers of the 'admission' type in the 'patient' cabinet. All index data as well as the number of document files will be queried. The hits will be sorted in an ascending way by the 'author' field and afterwards in a descending way by the 'date' field.

```
<?xml version="1.0" encoding="UTF-8" ?>
<DMSQuery>
<Archive name="Patient">
<ObjectType name="Admission" type="REGISTER" >
<Fields field_schema="ALL"/>
<Field name="Author" sortpos="1" sortorder="ASC"/>
<Field name="Date" sortpos="2" sortorder="DESC"/>
</Fields>
</ObjectType>
</Archive>
</DMSQuery>
```

Results of hierarchical search requests cannot be output in a sorted way.

Search conditions

Conditions can be defined not only for the requested object but for parent and child objects too. Therefore the clause object (ConditionObject attribute) for which the search conditions are to apply needs to be firstly defined. Search conditions for several objects can be defined simultaneously in a search request.

In the following example, all patients who were admitted to ward 1 on January 1, 2004 and who are taller than 180 cm will be retrieved.

```
<?xml version="1.0" encoding="UTF-16" ?>
```

```
<DMSQuery>
<Archive name="Patient">
<ObjectType name="Patient" type="FOLDER">
<Fields field_schema="DEF">
<Field name="Name"/>
<Field name="First name"/>
<Field name="ZIP code / city"/>
<Field name="Place"/>
</Fields>
<Conditions>
<ConditionObject name="Visit" type="REGISTER">
<FieldCondition name="Station" operator="=">
<Value>1</Value>
</FieldCondition>
<FieldCondition name="Start">
<Value>2004/01/01</Value>
</FieldCondition>
</ConditionObject>
<ConditionObject name="Body mass" type="DOCUMENT">
<FieldCondition name="Height [cm]" operator="&gt;">
<Value>180</Value>
</FieldCondition>
</ConditionObject>
</Conditions>
</ObjectType>
</Archive>
</DMSQuery>
```

As can be seen in this example, the '>' operator (operator='&gt;' attribute) has to be encoded so that the document is XML-compliant. If no operator is indicated in a condition, the operator '=' will be used. Following relational operators are valid:

Relational operators

Operator	XML-compliant format
<	&lt;
<=	&lt;=
=	=
!=	!=
<<New configuration name>>	&gt;
>=	&gt;=
BETWEEN	BETWEEN
NOT BETWEEN	NOT BETWEEN
IN	IN
NOT IN	NOT IN

Wildcards

The same wildcards as in enaio® client can be used for search requests on text fields.

Placeholder	Description
*	any string
?	Single character
~	Phonetic search (only MSSQL and Oracle)

The backslash is used as an escape character, i.e. /? is used to search for a question mark.

Zero value

The <NULL/> element is available to check a value for ZERO, and it has to be used instead of a <Value> element.

Special values

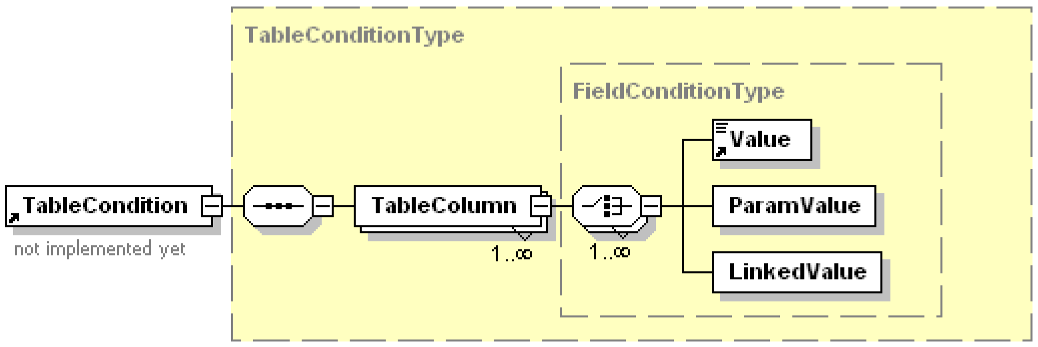
Special values can be depicted with the <SpecialValue> element. The following values are therefore available:

#COMPUTER-GUID#"	GUID of the logged-on user's computer
#COMPUTER-NAME#"	Name of the logged-in user's computer
#COMPUTER-IP#"	IP address of the logged-on user's computer
#CREATOR#"	Link to the basic parameter field "Creator"
#CREATIONDATE#	Link to the basic parameter field "Date of creation"
#ARCHIVIST#	Link to the basic parameter field "Archivist"
#ARCHIVINGDATE#"	Link to the basic parameter field "Date of archiving"
#USER#"	Name of the logged-on user
#OWNER#"	Link to the basic parameter field "Owner" which contains the owner's GUID
"#DATE#	current date

Characteristics of the DMS field types

Search conditions for table fields

Dialog elements of the table type have one or multiple columns which can be individually queried. In the XML request format this looks as follows:



Instead of a <FieldCondition> element a <TableCondition> element is indicated to which the table is referenced. The <TableColumn> element defines the table column for which the search criterion will be established.

Example:

```
<DMSQuery>
<Archive name="Test cabinet">
<ObjectType name="WinDoc">
<Fields field_schema="ALL" />
<Conditions>
<ConditionObject name="WinDoc">
<TableCondition name="MyTable">
<TableColumn name="1st column" operator="=">
<Value>11</Value>
</TableColumn>
</TableCondition>
</ConditionObject>
</Conditions>
</ObjectType>
</Archive>
</DMSQuery>
```

List with multi-selection

In multi-selection lists the wildcard '*' is always added in front or behind the search term.

Date

Basically, the date can be indicated in all valid formats. The following rule applies to two-digit years: when the entered year xy is smaller than 50, the date will become 20xy. When the entered year xy is bigger than or equal to 50, the date will become 19xy.

Example:

04	leads to 2004
76	leads to 1976

If the equal sign is used for an incomplete date, a corresponding time period will be searched.

Example:

=1999	leads to	BETWEEN 1999/01/01 AND 1999/12/31
!= 1999	leads to	NOT BETWEEN 1999/01/01 AND 1999/12/31

If two values (with two <Value> elements) are indicated within a date condition, they are automatically treated as one area and a 'BETWEEN' or 'NOT BETWEEN' is used in the SQL statement.

Text

The LIKE operator is automatically used if wildcards are used.

Radio buttons

If a search criterion is used for a radio button group, a possibly available group field as well as any radio button field of the group can be used to set the search criterion.

Boolean (AND/OR) links of conditions

By default, search conditions are linked with a logical AND. However, there is also the possibility to link search conditions with OR or any combination of AND and OR. For this purpose, the various field conditions have to be grouped using the <FieldGroup> element. Field groups may contain other field groups.

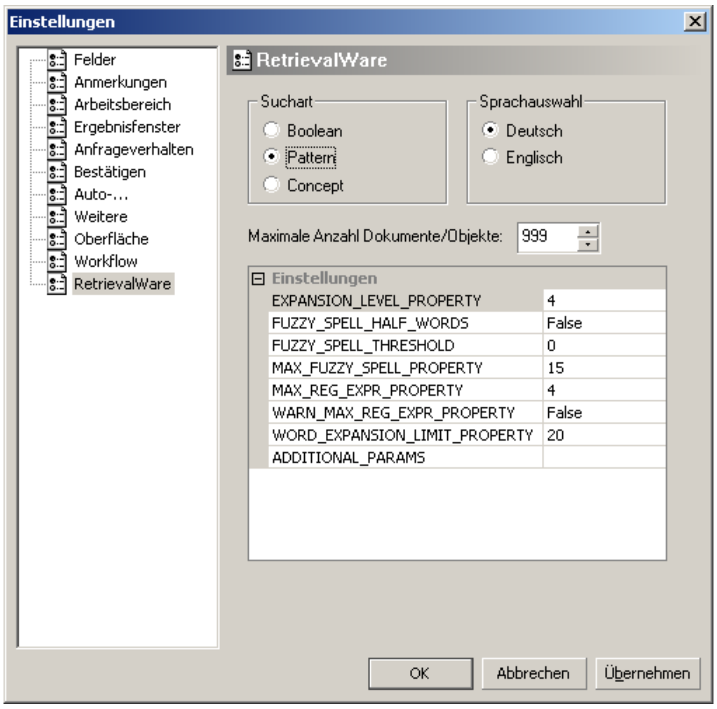
```
<ConditionObject name="Press archive">
<FieldGroup operator="OR">
<FieldCondition name="Specialist area">
<Value>Technology</Value>
</FieldCondition>
<FieldCondition name="Created by:">
<Value>Schmitz</Value>
</FieldCondition>
</FieldGroup>
<Created>
<From>2002/01/01</From>
<To>2003/12/31</To>
</Created>
</ConditionObject>
```

Link of the conditions with multiple <ConditionObject> elements

As already mentioned, conditions can be defined for several object types within a search request. If multiple <ConditionObject> elements refer to the same object type, the contained conditional groups will be linked with OR.

Full text conditions

A full text search can be carried out with the <Fulltext> element if the system is configured accordingly. Regular expressions can be written as text of the <Fulltext> element. If 'RetrievalWare' instead of the MS Indexserver is used for full text indexing, additional parameters can be set as attributes of the <Fulltext> element to control the search result. These attributes match the RetrievalWare settings as they can be configured in the client.



Suchart: Boolean [✓ Pattern] Concept

Sprachauswahl: [✓ Deutsch] Englisch

Maximale Anzahl Dokumente/Objekte: 999

Einstellungen

EXPANSION_LEVEL_PROPERTY	4
FUZZY_SPELL_HALF_WORDS	False
FUZZY_SPELL_THRESHOLD	0
MAX_FUZZY_SPELL_PROPERTY	15
MAX_REG_EXPR_PROPERTY	4
WARN_MAX_REG_EXPR_PROPERTY	False
WORD_EXPANSION_LIMIT_PROPERTY	20
ADDITIONAL_PARAMS	

name	Type	Use	Default
EXPANSION_LEVEL_PROPERTY	xs:short	Optional	4
FUZZY_SPELL_HALF_WORDS	xs:boolean	Optional	false
FUZZY_SPELL_THRESHOLD	xs:short	Optional	0
MAX_FUZZY_SPELL_PROPERTY	xs:short	Optional	15
MAX_REG_EXPR_PROPERTY	xs:short	Optional	4
WARN_MAX_REG_EXPR_PROPERTY	xs:boolean	Optional	false
WORD_EXPANSION_LIMIT_PROPERTY	xs:short	Optional	20
Mode	FullTextSearchModeType (Pattern | Boolean | Concept)	Optional	Pattern

The result of the full text search is limited by optionally set conditions on index data.

Location information

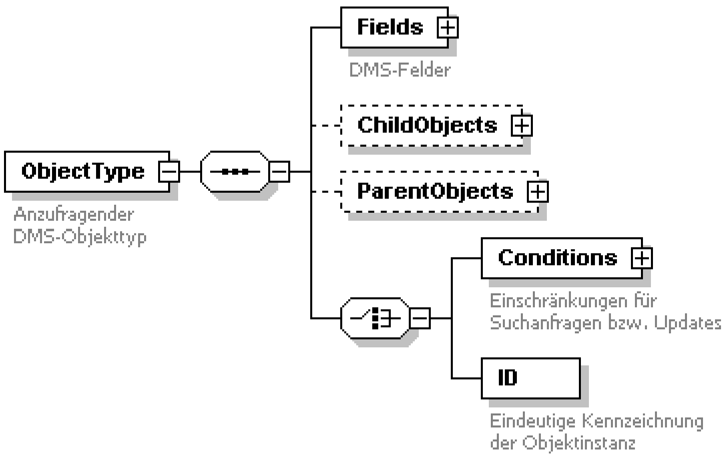
When searching for documents or registers it is very often desirable to obtain information about the location, e. g. folders or parent registers. Therefore it is set in the <ParentObjects> element to which extent information about parent objects is to be queried.

Linear search requests can be set for the folder and a maximum of one register. Folder, register and object fields are displayed in a hit line.

With hierarchical queries, the whole object path is specified if a <ParentsObjects> element exists. With <SubObjectType> elements, fields can be specified for the folder and single registers.

Export of hierarchical structures

Hierarchical search requests differ from linear search requests, especially because whole object structures can be exported. First, the search for the superior object is specified as described before. The search request will get the <ChildObject> element as sub element. This element contains the attributes 'export_depth' and 'child_schema'.


Fields
DMS-Felder
ChildObjects
ParentObjects
ObjectType
Anzufragender DMS-Objekttyp
Conditions
Einschränkungen für Suchanfragen bzw. Updates
ID
Eindeutige Kennzeichnung der Objektinstanz

Specification of export depth

The 'export_depth' attribute sets the export depth, i.e. the number of the object levels which will be exported. 0 means that no child objects are exported, 1 only exports direct child objects, etc.

Specification of object schema

The following values are available for the 'child_schema' attribute:

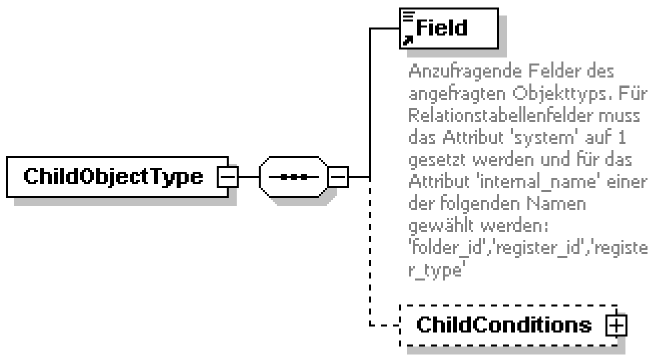
Value	Description
DEF	User-defined
REG	All registers are automatically added.
DOC	All documents are automatically added
ALL	All registers and documents are automatically added.

Specification of child object types

Certain child object types can be set with the <ChildObjectType> element. Within <ChildObjectType> elements, fields can be set according to the same rules as the main search object.

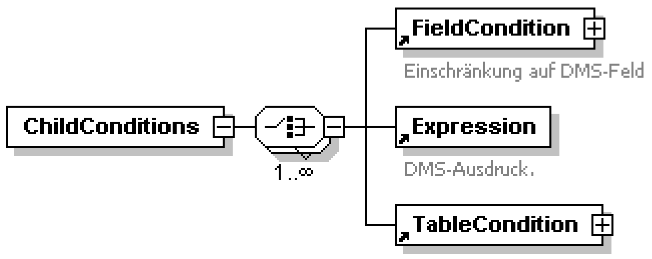
Note:

A child object type is allowed only once in the child object type list.


Field
Anzufragende Felder des angefragten Objekttyps. Für Relationstabellenfelder muss das Attribut 'system' auf 1 gesetzt werden und für das Attribut 'internal_name' einer der folgenden Namen gewählt werden: 'folder_id','register_id','register_type'
ChildObjectType
ChildConditions

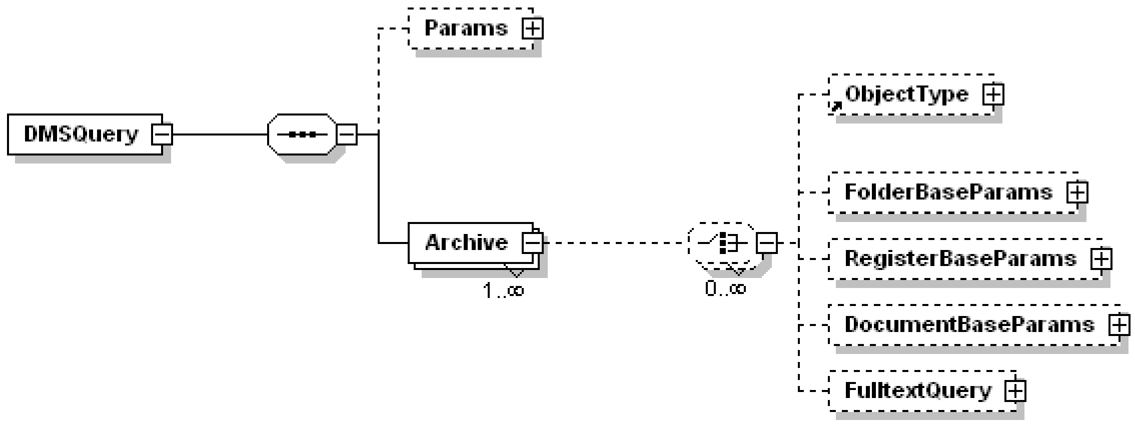
Restrictions for child objects

The selection of child objects can be limited further by additional conditions.



Basic parameter searches

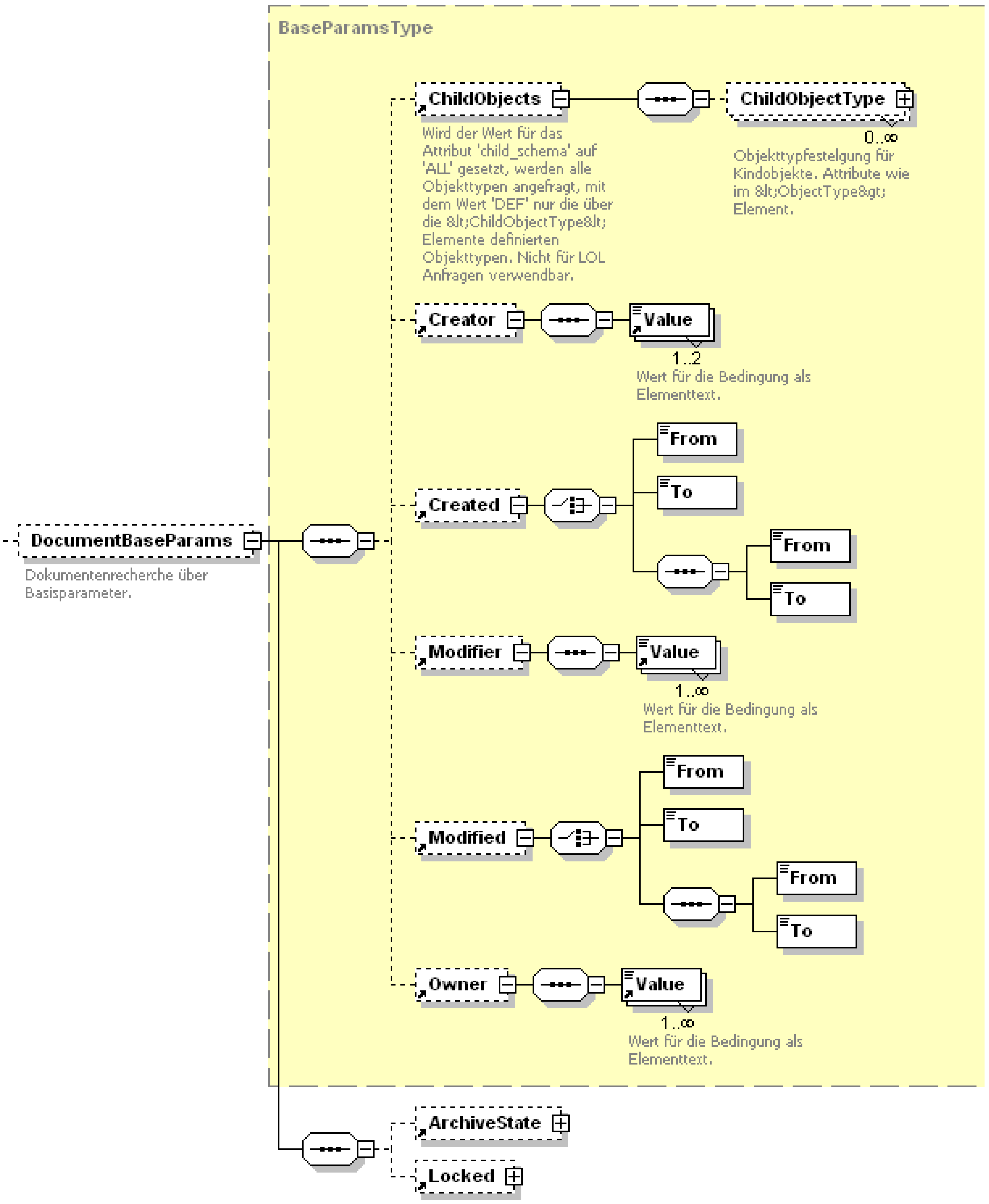
It is possible to search for folders, registers, and documents using basic parameters. After a cabinet has been defined with the <Archive> element, the details for a folder, register or document search can be set using the basic parameters.



Within the corresponding element (<FolderBaseParams>, <RegisterBaseParams> or <DocumentBaseParams) the following search conditions can be defined for the basic parameters:

XML element	Description
<Creator>	Creator
<Created>	Creation date:
<Modifier>	User who modified the object last
<Modified>	Date of the last modification If two values are indicated, the period between these dates will be searched.

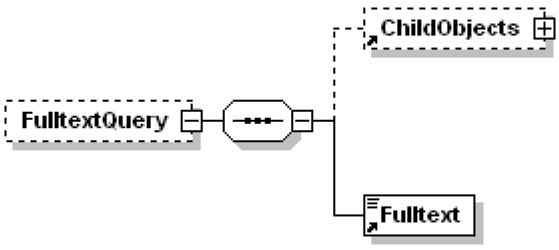
<Owner>	User name of the owner
<ArchiveState><ArchiveStateValue>	For documents: archiving status. Permitted values: ARCHIVED, ARCHIVABLE, NOT_ARCHIVABLE, NO_PAGES, PAGE_ERROR, REFERENCE
<Locked><LockStateValue>	For documents: Check-out status. Permitted values: UNLOCKED, SELF, OTHERS, EXTERNAL


BaseParamsType
ChildObjects
Wird der Wert für das Attribut 'child_schema' auf 'ALL' gesetzt, werden alle Objekttypen angefragt, mit dem Wert 'DEF' nur die über die <ChildObjectType< Elemente definierten Objekttypen. Nicht für LOL Anfragen verwendbar.
ChildObjectType
0..∞
Objekttypfestelgung für Kindobjekte. Attribute wie im <ObjectType> Element.
Creator
Value
1..2
Wert für die Bedingung als Elementtext.
Created
From
To
From
To
DocumentBaseParams
Dokumentenrecherche über Basisparameter.
Modifier
Value
1..∞
Wert für die Bedingung als Elementtext.
Modified
From
To
From
To
Owner
Value
1..∞
Wert für die Bedingung als Elementtext.
ArchiveState
Locked

With the <ChildObject> element, the selection of object types can be limited or the field selection and the sort sequence in the hit list can be defined.

Full text searches

If full text indexing is configured in enaio®, a search against all object types accordingly configured in the graphical editor can be carried out. For this purpose, after specifying the cabinet with the <Archive> element, the full text regular expression is indicated with the <Fulltext> element in the <FulltextQuery> element. Details about the <Fulltext> element can be found in the section Full Text Conditions.



With the <ChildObject> element, the selection of object types can be limited or the field selection and the sort sequence in the hit line can be defined.

Example:

Full text search for the word 'meningitis' in all referral letters and diagnoses.

```
<?xml version="1.0" encoding="UTF-8" ?>
<DMSQuery>
<Archive name="Patient">
<FulltextQuery>
<ChildObjects child_schema="DEF">
<ObjectType name="Doctor's letter">
<Fields>
<Field name="Date" />
<Field name="Senior doctor" />
<Field name="Type" />
</Fields>
</ChildObjectType>
<ObjectType name="Diagnosis">
<Fields field_schema="ALL" />
</ChildObjectType>
</ChildObjects>
<Full text>Meningitis</Full text>
</FulltextQuery>
</Archive>
</DMSQuery>
```

Parameterization of search requests

To make it easier to reuse search requests, search conditions can be parameterized. All parameters below the <DMSQuery> element are defined and initialized. In the search conditions, these parameter values can be referenced with their parameter name using the 'ref' attribute in the <ParamValue> element.

Example:

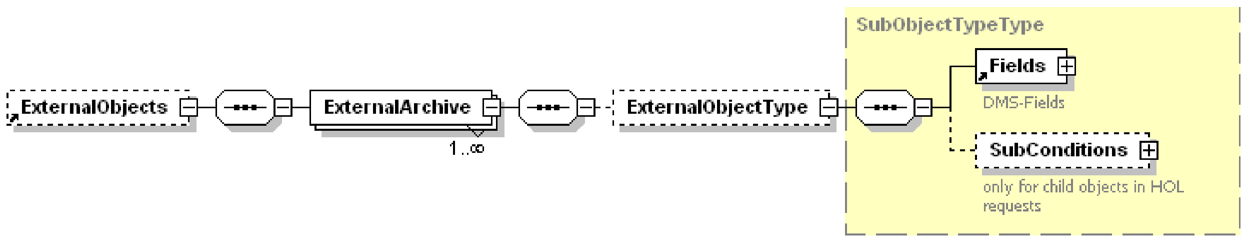
```
<DMSQuery>
<Params>
<Param name="PatID">3987</Param>
</Params>
<Archive name="Patient">
```

```
<ObjectType name="Patient">
<Fields field_schema="ALL" />
<Conditions>
<ConditionObject name="Patient">
<FieldCondition name="PatientID">
<ParamValue ref="PatID" />
</FieldCondition>
</ConditionObject>
</Conditions>
</ObjectType>
</Archive>
</DMSQuery>
```

Thus, a client application can change the value for the condition without detailed knowledge of the search structure by using the parameter.

Multi-cabinet search requests

If information about an object is required in a hit list and the information is contained in an object from another cabinet, this can be formulated in an HOL search request with the <ExternalObjects>



element.

The <ExternalObject> element will be defined below the <ObjectType> element. An external object is linked to the initial object with a field reference. For this purpose, an arbitrary alias name is therefore defined by the 'link_name' attribute of the initial object field which will provide the value for the link. This alias name is referenced with the 'ref' attribute of the <LinkedValue> in the field condition of the external object.

Example:

For every B/W document, the address for the author has to be provided.

```
<DMSQuery>
<Archive name="Press archive">
<ObjectType name="S/W scans">
<Fields>
<Field name="Document type" />
<Field name="Author" link_name="Author" />
<Field name="Date" />
</Fields>
<ExternalObjects>
<ExternalArchive name="Addresses">
<ExternalObjectType name="Addresses">
<Fields field_schema="ALL" />
<SubConditions>
<FieldCondition name="Name:">
<LinkedValue ref="Author" />
</FieldCondition>
</SubConditions>
</ExternalObjectType>
</ExternalArchive>
</ExternalObjects>
```

```
</ObjectType>
</Archive>
</DMSQuery>
```

Detailed information

Basic parameters

By setting the 'baseparams' flag to '1', all basic parameters will be queried in addition to the index fields. A separate element group will be created for this purpose, especially in the HOL.

For all object types values for

§ Creator

§ Creation date:

§ Modifier

§ Modification date

§ Owner

will be specified. The value for the owner is the name output as text and the user GUID as 'osguid' attribute.

The check-in or check-out status and the archiving status as well as retention times are additionally determined for documents.

```
<BaseParams>
<Creator>love</Creator>
<Created>2003/12/12</Created>
<Modifier>root</Modifier>
<Modified>2003/12/14</Modified>
<Owner guid="BC1123CDFAAAS">love</Owner>
<ArchiveState>ARCHIVABLE</ArchiveState>
<Locked>SELF</Locked>
</BaseParams>
```

Object status

By setting the 'status' attribute to '1' in the <DMSQuery> element the following status information will be indicated:

§ Links

And especially for documents:

§ Module type

§ Archiving status

§ Checked-in/checked-out

§ Number of pages

§ Number of real document pages (system field: OBJECT_DOCPAGECOUNT). If these are known.

Rights

By setting the 'rights' attribute to '1' in the <DMSQuery> element the access rights on every object will be determined for the searching user. The following rights will be determined:

Attribute	Description
modify_index	The user is allowed to change the index data of an object
delete_object	The user is allowed to delete the object
export_object	The user is allowed to open and/or export the object
modify_object	The user is allowed to edit the document files.

Example:

```
<Rights modify_index="1" delete_object="1" export_object="1"
modify_object="1" />
```

Object type relations

The number of object instances of a given type can be limited with object type relations. Use the 'ObjectInserts' job parameter and the 'object_inserts' search request attribute to query the number of object instances which can still be inserted taking into account object type relations and the rights system. For every possible child object type which can be inserted according to the rights system, there is an <ObjectInsert> element below the <Rights> element which has the attributes to indicate the object type ('type' attribute) and the number of instances which can be inserted ('count' attribute). If the value for 'count' is '-1' there are no limits.

Additionally, in the <Rights> element there is the 'object_inserts' attribute which indicates whether object type relations have been queried.

If the 'rights' attribute has not been set to '1', the attribute for access rights in the <Rights> element will be set to '-1'.

```
<Rights object_inserts="1" modify_index="1" delete_object="1" export_object="1"
modify_object="1">
<ObjectInserts type="65536" count="-1" />
<ObjectInserts type="131108" count="-1" />
<ObjectInserts type="131119" count="0" />
<ObjectInserts type="196608" count="-1" />
<ObjectInserts type="196619" count="0" />
<ObjectInserts type="196622" count="-1" />
<ObjectInserts type="262144" count="-1" />
<ObjectInserts type="6488064" count="198" />
</Rights>
```

File information

By setting the 'fileinfo' attribute to '1' in the <DMSQuery> element, the following file information will be extracted for every document and written into a <FileProperties> element as attribute.

For document references (green arrow), file information of the linked document is returned if the job parameter [FollowDocLink] is set to '1' (or in the <DMSQuery> element). In this case, also the attributes 'linkid' and 'linktypeid' are written.

The following file properties can be determined:

Attribute	Description
count	Number of files

size	Size of the documents in bytes
extension	File type standard extension
mimetype	Mime type
linkid	ID of the linked object
linktypeid	Type ID of the linked object
documentpagecount	Number of document pages (if known)

Example:

```
<Object id="415">
<FileProperties count="1" size="179489" extension="jpg"
mimetype="image/jpeg"/>
</Object>
```

Notes

If the DMSQuery attribute 'remarks' is set to '1,' notes and note links, if available, will be returned for every requested document. Text notes and object notes (links) will be output. This function is available for HOL search requests only.

Attribute/day	Description
id	Note ID
type	Note type (1 = white, 2 = yellow, 3 = green, 4 =)
relation	With object relation 1
medium	Media ID if notes are filed in the work directory, 0 if the note's text is filed in the database.
Creator	Name of the creator with the creator's internal ID as attribute.
Created	Date of creation with timestamp as attribute.
Modifier	Last editor with the internal ID as attribute.
Modified	Date of the last modification with the timestamp as attribute.
Text	Note text with internal ID, if notes have been filed in the database. Empty if it is an object note.

Example:

```
<Object id="81">
<Remarks>
<Remark id="1413" type="1" relation="0" medium="4">
<Creator id="16">MAIER</Creator>
<Created value="1143560855">2006/03/28 17:47:35</Created>
<Modifier id="18">SCHMIDT</Modifier>
<Modified value="1946463756">2006/04/30 14:50:12</Modified>
<Text id="">Note text</Text>
</Remark>
```

```
</Remarks>
</Object>
```

Language settings

With the 'lang_id' or the 'query_language' attribute, you can specify in which language the search request will be carried out. The field names or object names are then searched depending on the language. Without any specification, the default language will be used.

Icons

By setting the 'icon' attribute in the DMSQuery to '1', the system is instructed to return the icon IDs of all user-defined icons. The icon IDs from the archive area, as well as user-defined icon IDs from a hit list will be determined. The DMSContent elements '<Object' (HOL search request) and '<Row>' (LOL search request) will additionally get the 'iconid' attribute.

The cnv.GetIcons job can be used to obtain an image file on the basis of the icon ID.

Document variants

For documents with the 'W-documents' module type, variants can be created and administered in enaio®. By setting the 'variants' attribute in the DMSQuery element to '1,' variants will be determined for every W-document and output according to their hierarchical structure through <DocumentVariant> and <DocumentVariants> elements. However, no index data will be determined.

Attribute	Description
is_active	'1' if this variant is active, otherwise '0'.
doc_id	Document ID of this variant
doc_ver	Version name
doc_parent	ID of the original variant that was used to generate this variant. Note that, IDs of documents that were already deleted or no longer exist in the system may be contained.

Example:

```
<DocumentVariant is_active="1" doc_id="64758" doc_ver="Original"
doc_parent="0">
<DocumentVariants level="0">
<DocumentVariant is_active="0" doc_id="73405" doc_ver="1.0.0"
doc_parent="64758">
<DocumentVariants level="1">
<DocumentVariant is_active="0" doc_id="73406" doc_ver="1.1.0"
doc_parent="73405"/>
<DocumentVariant is_active="0" doc_id="73407" doc_ver="2.0.0"
doc_parent="64758">
<DocumentVariants level="1" />
</DocumentVariant>
</DocumentVariants>
</DocumentVariant>
```

Note:

When searching for W-documents, the active variant will be returned.

Define multiple search requests in a search document

Multiple search requests can be defined in a search document.

Example:

```
<DMSQuery>
<Archive name="Addresses">
<ObjectType name="Addresses">
…
</ObjectType>
</Archive>
<Archive name="Patient">
<ObjectType name="Color images" alias="F2">
      …
</ObjectType>
<ObjectType name="Grayscale images" alias="G1">
…
</ObjectType>
<ObjectType name="Color images" alias="F1">
…
</ObjectType>
</Archive>
</DMSQuery>
```

To specifically access results of search requests with the same object type, it is possible to use the optional attribute '**alias**' in the <ObjectType> element. This alias name will then be returned in the <ObjectType> elements of the result documents.

For a hierarchical search request with the determination of a location, the attribute will additionally be written to the folder's <ObjectType> element.

Browsing Hit Lists

To call hit lists by page, a search request has to be carried out multiple times and the starting point has to be indicated to the job with the 'PageSize' parameter and the 'Offset' parameter. The maximum number of hits can be limited with 'MaxHits'.

In the result document you will find the <Statistics> element at the end of a hit list with the attributes 'startpos', 'pagesize' and 'total_hits'.

```
<Statistics startpos="20" pagesize="20" total_hits="50" />
```

The element helps you to determine how many hits the queried page actually contains and whether there are any further hits. The value of the 'total_hits' attribute can at most be the maximum value of the 'MaxHit' input parameter.

Example:

From the total search result only a number of 20 hits is desired to be displayed. At most, only the first 100 hits are of interest. However, in fact 120 objects match the search request. PageSize=20 and MaxHits=101 are set for every call. If MaxHits was set to 100, it would not be possible to see whether there are more than 100 hits.

Page	**Input**	**Output**
1	Offset=0	<Statistics startpos='0' pagesize='20' total_hits='101' />
2	Offset=20	<Statistics startpos='20' pagesize='20' total_hits='101' />

3	Offset=40	<Statistics startpos='40' pagesize='20' total_hits='101' />
4	Offset=60	<Statistics startpos='60' pagesize='20' total_hits='101' />
5	Offset=80	<Statistics startpos='80' pagesize='20' total_hits='101' />

After calling the first page, the querying person will know that there are (101-1)/20 = 5 pages. With page 5, startpos + pagesize = 100 and the maximum number of hits is achieved. As there are 101 > 100 hits, the querying person will know that there would have been more results.

DMS.GetObjectDetails

Description:

This job is used to determine index data of a single DMS object. The location is irrelevant. Data of inactive variants can be determined, too. The result is returned in HOL format in the DMS Content. All input parameters that can be specified for the job DMS.GetResultList in order to control the output information can also be used in the same way for GetObjectDetails.

By default, this job sets the request property [FollowDocLink] to '1' if it is not disabled by the job parameter. This will cause the document properties of the linked DMS object to be returned by default if [FileInfo] is requested and if it is a document reference object (see File Information).

Parameter:

§ ObjectID (INT): ID of the object instance

ObjectType (INT): object type. If this parameter is not indicated or indicated as value '-1', the job itself will specify the object type.

[SystemFields] (String): list separated with semicolon from additionally requested system fields. The internal names of system fields are expected (see System Fields). If basic parameters, status, file info are requested, they are given priority, i.e. the information requested with SystemFields is returned in the corresponding results block (e.g. within <baseparams>) and not redundantly returned.

Example: SystemFields=OBJECT_MEDDOCID;OBJECT_MEDDOCNA

Other possible parameters: See job DMS.GetResultList

Return values:

XML (BASE64): index data in XML format

DMS.GetDeletedObjects

Description:

This job is used to output the contents of the trash can of the logged-on user. The result is returned in DMS Content format. The same output format options apply as for the job DMS.GetResultList.

A query within the trash can can be performed, too. To do so, a query in DMSQuery XML format must be stated and passed as an 'XML' parameter.

Parameter:

Flags (INT): flags to control the output format

§ 0x00000010 = XML result is returned as a file, otherwise as buffer

§ 0x00001000 = XML result is encoded as UTF-8, otherwise UTF-16

[UserID] (INT): ID of the user whose trash can is to be read. With the value -1 the whole system trash can will be read, the default value '0' indicates the logged on user. To read the system trash can or the trash can of another user, the system role 'enaio® Client: View system trash can' is required, access is otherwise denied 'error code (-1069).

§ [XML] (BASE64): Search request in DMSQuery format

[CheckParams] (INT): 0 = if no value for a search request parameter, conditions referencing this parameter are ignored. This is the default value. 1 = an error message will be returned if a referenced parameter was not defined

Return values:

Count (INT): number of returned datasets

TotalHits (INT): number of available hits

Depending on the input flag

[XML] (BASE64): hit list in XML format

Or

[FileCount] (INT): only one file is returned

[File list]: name and path of the XML file containing the hit list

DMS.GetLinkedObjects

Description:

This job is used to output objects linked to a given object. The result is returned in DMS Content format. The same output format options apply as for the job DMS.GetResultList.

The output does not comprise any notes. In addition to the index fields (see output format option 'FieldSchema') the Relations GUID is returned:

Example: `<Column object="S/W-Scans" type="DOCUMENT" name="osrelid" system="1" datatype="TEXT" dbname="osrelid" ostype="X" osguid="1300" size="32">HYP_ID</Column>`

Parameter:

Flags (INT): flags to control the output format

§ 0x00000010 = XML result is returned as a file, otherwise as buffer

ObjectID (INT): object ID whose the links are to be displayed

Return values:

TotalHits (INT): number of linked objects

Depending on the input flag

[Result] (BASE64): object list in XML format (or simple text format if required)

Or

[FileCount] (INT): only one file is returned

[File list]: name and path of the XML file containing the hit list

DMS.GetForeignObjects

Description:

This job can be used to determine all objects with a document reference to a given object (aka green arrow link). The result is returned in DMS Content format. The same output format options apply as for the job DMS.GetResultList.

Parameter:

Flags (INT): flags to control the output format

- 0x00000010 = XML result is returned as a file, otherwise as buffer

ObjectID (INT): ID of the target object whose document links are to be displayed. The target object ID of the green arrow link.

[ObjectTypes]: (STRING) a semicolon-separated list of the link types. If this parameter is not set, all types from all cabinets are searched. The link type can be transferred as a type ID or as an internal name of the type. If an archive (cabinet) is transferred as the type, all the document types it contains are used as source type.

Example: `ObjectTypes=1;internal_email;262432`

All document types from the cabinet with the ID 1, one document type with the internal name "interne_email," and documents with the object type ID 262432 are considered as source types.

The job otherwise supports the same input parameters as DMS.GetObjectDetails or DMS.GetResultList.

Return values:

TotalHits (INT): number of visible document link objects to the target object.

TotalForeignObjects (INT): total number of document links to the target object. The security system is not considered when determining the number. If the input parameter "ObjectTypes" is used, only the number of links found in the specified types is returned.

Depending on the input flag

[Result] (BASE64): object list in XML format (or simple text format if required)

Or

[FileCount] (INT): only one file is returned

[File list]: name and path of the XML file containing the hit list

DMS.SelectDistinctFieldValues

Description:

This job returns a list of all values of a given DMS field. Table control elements and multiple parameter fields cannot be included.

Parameter:

- Flags (INT): always 0
- ObjectType (INT): Object type

FieldName: (STRING):field name

[Notation] (INT): type of field name: 0 (Default) = DMS name, 1 = internal name, 2 = database field name

[Filter:] (STRING):restriction/search criteria. An * is automatically appended for text fields.

[SortOrder] (STRING):sort order. Permitted values: 'ASC' (default) or 'DESC'

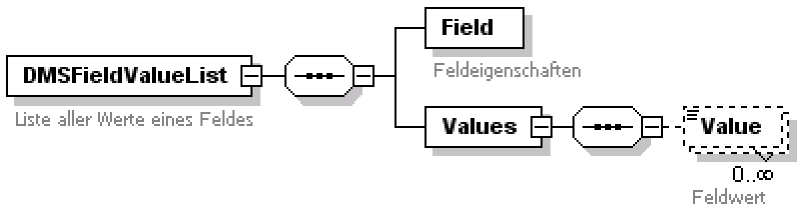
[MaxHits] (INT):max number of values to be returned. Default=-1 (all)

Return values:

TotalHits (INT) total field value number

Count (INT): number of output field values (see input parameters Max Hits)

FieldValues (BASE64): object list in the UTF-8 encoded DMSFieldValueList XML format:



DMS.GetUserTrayObjects

Description:

This job returns objects with a type as well as typeless objects from the logged-in user's filing tray. The result is returned in DMS Content format. The same output format options apply as for the job DMS.GetResultList.

Parameter:

Flags (INT): flags to control the output format

§ 0x00000010 = XML result is returned as a file, otherwise as buffer

Return values:

TotalHits (INT): number of objects in the user tray

Depending on the input flag

[Result] (BASE64): object list in XML format

Or

[FileCount] (INT): only one file is returned

[File list]: name and path of the XML file containing the hit list

Example of an output (HOL format):

```
<DMSContent format="HOL" version="4.60.617.4328" timestamp="2004-12-03T14:25:01"
user="LOVE" station="MLOVE">
<Archive name="press archive" id="1" osguid="8EE74447EFC7430F8793F2939A9C044F">
<ObjectType name="Word texts" id="262144" maintype="4" cotype="0"
osguid="838360EF620443EF9A66A280CF52F4AE" type="DOCUMENT" module="WINDOWS"
table="object2">
<ObjectList>
<Object id="73145">
<Fields>
```

```
<Field name="links" system="1" datatype="INTEGER" dbname="links" ostype="9"
osguid="1114" size="10">0</Field>
<Field name="count" system="1" datatype="INTEGER" dbname="count" ostype="9"
osguid="1101" size="10">1</Field>
<Field value="2" name="flags" system="1" datatype="INTEGER" dbname="flags"
ostype="9" osguid="1102" size="10">NOT_ARCHIVABLE</Field>
<Field value="0" name="lockuser" system="1" datatype="INTEGER" dbname="lockuser"
ostype="9" osguid="1116" size="10">UNLOCKED</Field>
<Field value="4" name="maintype" system="1" datatype="INTEGER" dbname="maintype"
ostype="9" osguid="1108" size="10">WINDOWS</Field>
<Field name="Author" datatype="TEXT" dbname="field1" ostype="X"
osguid="F60B3D9B9E9C4FE8AE6CB78E4DE25826" size="50">Love</Field>
<Field name="Source" datatype="TEXT" dbname="field2" ostype="X"
osguid="E3EDAF50F4F4404C860E6043D7894272" size="150">Source1</Field>
<Field value="4711" name="Text2" datatype="TEXT" dbname="field3" ostype="X"
osguid="18C0D7D305DE40BB87A160B1115FC2A3" size="50">Document text</Field>
</Fields>
</Object>
</ObjectList>
<Statistics startpos="0" pagesize="-1" total_hits="1"/>
</ObjectType>
</Archive>
<TypelessObjects>
<TypelessObject id="61359" maintype="2" module="BLACKWHITE" user="LOVE"
timestamp="2002/10/18 11:19:10" pagecount="1"/>
<TypelessObject id="61360" maintype="2" module="BLACKWHITE" user="LOVE"
timestamp="2002/18/10 11:19:12" pagecount="10"/>
<TypelessObject id="63399" maintype="3" module="COLOR" user="LOVE"
timestamp="2003/01/21 15:23:36" pagecount="1"/>
</TypelessObjects>
<Messages/>
</DMSContent>
```

DMS.GetWorkflowObjects

Description:

This job returns objects with a type as well as typeless objects from the logged-in user's filing tray. The result is returned in DMS Content format. The same output format options apply as for the job DMS.GetResultList.

Parameter:

Flags (INT): flags to control the output format

§ 0x00000010 = XML result is returned as a file, otherwise as buffer

ID<1..n> (INT): ObjectIDs. A parameter must be specified for each object instance. For example

ID1=123

ID2=245

…

Return values:

TotalHits (INT): number of objects actually found in the workflow tray

Depending on the input flag

[Result] (BASE64): object list in XML format

Or

[FileCount] (INT): only one file is returned

[File list]: name and path of the XML file containing the hit list

DMS.ExecuteStoredQuery

Description:

This job can be used to execute saved search requests. The result is returned in DMS Content format.

Parameter:

Flags (INT): flags to control the output format

§ 0x00000010 = XML result is returned as a file, otherwise as buffer

§ 0x00001000 = XML result is encoded as UTF-8, otherwise UTF-16

QueryID (INT): ID of the search request

[CheckParams] (INT): 0 = if no value for a search request parameter, conditions referencing this parameter are ignored. This is the default value. 1 = an error message will be returned if a referenced parameter was not defined

If it is a search request with parameters, the request parameters have to be passed as job parameters. The $ characters, which enclose parameters in the search request, must be omitted, e.g. VAR1 or STAT3.

In addition, the following parameters can be set for formatting the returned XML document: RequestType, OutputFormat, BaseParams, Offset, Pagesize, MaxHits, Rights, DateFormat, Variants, FileInfo, Baseparams. The description of these parameters can be found in the description of the job dms.GetResultList

Return values:

Count (INT): number of returned datasets

TotalHits (INT): number of available hits

Depending on the input flag

[XML] (BASE64): hit list in XML format

Or

[FileCount] (INT): only one file is returned

[File list]: name and path of the XML file containing the hit list

DMS.GetStoredQuery

Description:

This job returns saved search requests in the DMS Query Format. All object fields will be applied in the list of the queried fields. The same output results apply as for dms.GetResultList.

Parameter:

Flags (INT): Flags must be 0

QueryID (INT): search request ID

[QueryMode] (INT): defines the required search request behavior.

Available values are: Auto (-1)= automatically use what was defined in the search request. Folder(0) = Query against folders Register(1)=Query against register Dokument(2)=Query against documents Unfiltered (-2) folders+registers+documents are returned.

Default is 'unfiltered (-2)'.

Return values:

[Query] (BASE64): search request in XML format (UTF-8 encoded)

[ExpertMode] (INT): 1, if the saved search request is an expert search request.

Example:

Saved search request:

```
[196608@0]
#OSACT#=1
#OSPOS001#=$STAT1$
[SYSTEM]
NAME=Stat1
IDENT=73706
VARREQUEST=1
DEFACTION=0
```

Converted search request:

```
<DMSQuery requesttype="LOL" outputformat="LOL">
<Params>
<Param name="$STAT1$"></Param>
</Params>
<Archive name="press archive">
<ObjectType name="color images" alias="Stat1">
<Fields>
<Field name="date" system="0"></Field>
<Field name="author" system="0"></Field>
<Field name="source" system="0"></Field>
<Field name="content" system="0"></Field>
<Field name="viewable" system="0"></Field>
</Fields>
<Conditions>
<ConditionObject name="color images">
<FieldCondition name="author" operator="=">
<ParamValue ref="$STAT1$"></ParamValue>
</FieldCondition>
</ConditionObject>
</Conditions>
</ObjectType>
</Archive>
</DMSQuery>
```

DMS.AddStoredQuery

Description:

This job is used to create a new saved search request. The search request must be transferred in DMS Query Format. The query will be converted into the internal format for saved searches. As the format for saved search requests only allows a subset of DMS search possibilities, certain limits apply. See dms.ConvertQuery.

Parameter:

Flags (INT): Flags must be 0

Name (STRING): Search request name

Query (STRING/BASE64): search in the DMSQuery XML Format

[TreeParent] (INT): ID of the desktop folder where the search request is located

[Scope] (STRING): scope of application. Allow values: 'public' for public search requests, 'private' for user-related search requests. The default value is 'private'.

[IconID] (INT): ID of an icon displayed in the client. Default=0

[DefAction]: Action to be carried out when opening the stored query.

0=execute, 1=edit, 2=determine count. Default=0

Return values:

QueryID (INT): ID of the search request

DMS.UpdateStoredQuery

Description:

This job is used to update an existing saved search request and/or its properties. The search request must be transferred in DMS Query Format. The query will be converted into the internal format for saved searches. As the format for saved search requests only allows a subset of DMS search possibilities, certain limits apply. See dms.ConvertQuery.

Parameter:

Flags (INT): Flags must be 0

QueryID (INT): ID of the search request

[Query] (STRING/Base64) search request in DMSQuery XML format

[Name] (STRING): new name of the search request if the search request is to be renamed

[IconID] (INT): ID of an icon displayed in the client. Default=0

[DefAction]: Action to be carried out when opening the stored query.

0=execute, 1=edit, 2=determine count. Default=0

Return values: none

DMS.RemoveStoredQuery

Description:

This job is used to delete an existing saved search request.

Parameter:

Flags (INT): Flags must be 0

QueryID (INT): ID of the search request

Return values: none

DMS.ConvertQuery

Description:

With this job different search request formats can be converted into each other.

At present, the following formats are supported:

DMS – DMSQuery XML format

STQ – format for saved search requests

ABN – subscription format

Parameter:

Flags (INT): Flags must be 0

Query (STRING or BASE64): Search request type

InputFormat (STRING): input format (DMS, STQ, ABN)

OutputFormat (STRING): output format (DMS, STQ, ABN)

If STQ (saved search request) is chosen as the output format, the following job parameters still have to be defined.

Name (STRING): query ID

QueryID (INT): ID of the search request

[IconID] (INT): ID of an icon displayed in the client. Default=0

[DefAction]: Action to be carried out when opening the stored query.

0=execute, 1=edit, 2=determine count. Default=0

If ABN (subscription) is chosen as the output format, the following job parameters still have to be defined.

[GarbageMode] (INT): 1=only objects from the trash can are taken into account. 0=objects from the trash can are not taken into account.

Return values:

Query (BASE64): search request text. UTF-8-encoded for the DMSQuery XML format, otherwise ANSI.

Restrictions:

As the search possibilities of the DMSQuery XML format exceed those of other formats, the following limitations apply for the DMSQuery XML format as input format:

* Hierarchical structures are not supported, i.e. <ParentObjects>, <ChildObjects>, and <ExternalObjects> are ignored.

* Parameter names must have the format $VARnnn$ or $STATnnn$ where nnn can be a number between 000 and 999.

* Field groups within conditions cannot be used.

* Only one value can be specified for each condition.

* No conditions can be formulated for basis parameters and system fields.

* Saved search requests (STQ) in expert mode cannot be converted.

DMS.GetObjectHistory

Description:

This job returns the editing history for a given object in XML format.

Parameter:

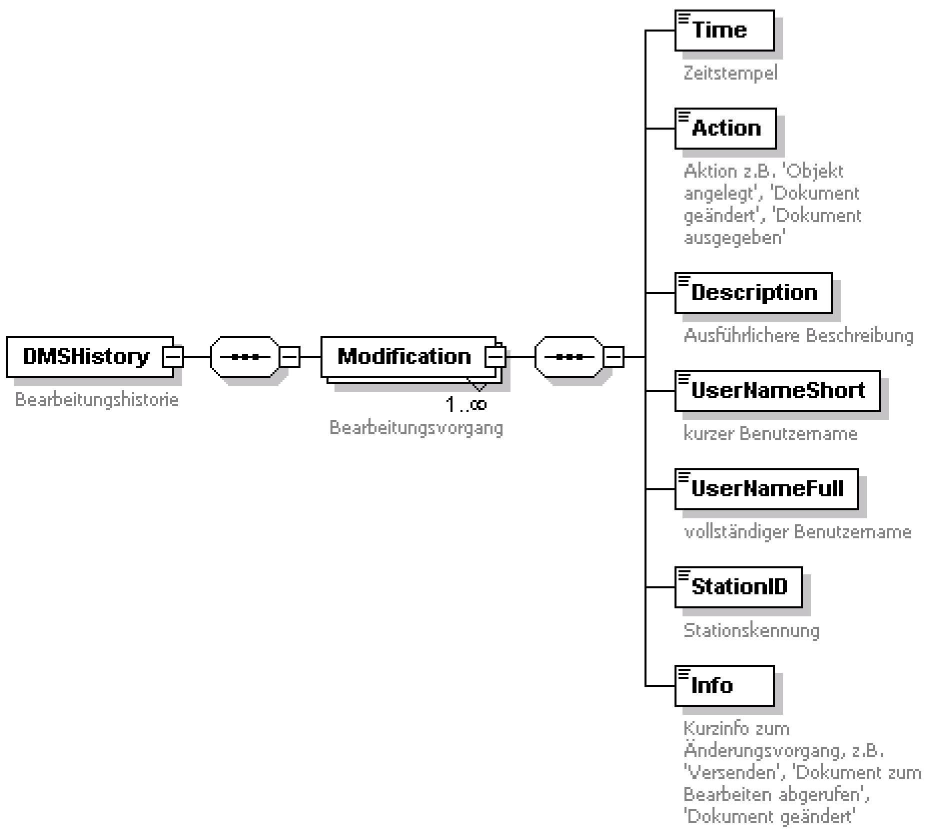
Flags (INT): Flags must be 0

ObjectID (INT): ID of the object

[LangID(INT)]: Language in which the action descriptions are to be output. Action descriptions are available in German (7), English (9), and French (12). If this parameter is not specified or if the language is not available, German is used as the default language.

[Encoding (STRING)]: XML encoding for the result. 'UTF-16' (default) and 'UTF-8' are possible.

Return values: History (Base64): editing history in XML format:


DMSHistory
Bearbeitungshistorie
Modification
1..∞
Bearbeitungsvorgang
Time
Zeitstempel
Action
Aktion z.B. 'Objekt angelegt', 'Dokument geändert', 'Dokument ausgegeben'
Description
Ausführlichere Beschreibung
UserNameShort
kurzer Benutzername
UserNameFull
vollständiger Benutzername
StationID
Stationskennung
Info
Kurzinfo zum Änderungsvorgang, z.B. 'Versenden', 'Dokument zum Bearbeiten abgerufen', 'Dokument geändert'

Note: if the user was deleted at the same time, UserNameShort and UserNameFull will not be filled in.

DMS.GetShadowData

Description:

This job is used to read the index data of an object from the shadow table.

Parameter:

Flags (INT): Flags must be 0

Guid (String): GUID of the change progress from the history table à DMS.GetObjectHistory

[ObjectID (INT)]: ID of the object

[Encoding (STRING)]: XML encoding for the result. 'UTF-16' (default) and 'UTF-8' are possible.

Return values: ShadowData (Base64): Index data in DMSContent –HOL format

DMS.GetObjectsByDigest

Description:

This job returns object information based on a hash value (fingerprint) of any file. It can be used to determine if a file was already filed in the system. This job uses the 'std.FindDocumentDigest' job and subsequently executes a validity check. The input parameter 'Digest' is a fingerprint of the file which can be determined with the SHA2-256 algorithm. Help functions are available in the respective client libraries to determine the digest from a file without having to transfer the file to the server. (See JDL:DigestUtil, CDL:CalcFileDigest)

Parameter:

Flags (INT): Flags must be 0

Digest (String): hash value of the file whose existence is to be checked. (SHA2-256Bit)

Return values:

TotalHits (INT): Number of hits

ObjectIds (String): Comma-separated list of the object IDs of the found documents.

TypeIds (String): Comma-separated list of object types.

DMS Reference

§ System fields

§ Date formats

System fields

The following system fields can be included in the XML search request:

Internal name [@internal_name]	Field number [@osguid]	Database field name [@fieldname]	Type	Length
OBJECT_ID	1100	id	Text	10
OBJECT_COUNT	1101	anzahl	Text	10
OBJECT_FLAGS	1102	flags	Text	10
OBJECT_AVID	1103	archivist	Text	255
OBJECT_AVDATE	1104	archived	Date	
OBJECT_CRID	1105	anleger	Text	255
OBJECT_CRDATE	1106	created	Date	
OBJECT_TIME	1107	timestamp	Timestamp	10
OBJECT_MAIN	1108	main type	Text	10

OBJECT_CO	1109	subtype	Text	10
OBJECT_MEDDOCID	1110	medium_doc	Text	10
OBJECT_MEDDIAID	1111	medium_dia	Text	10
OBJECT_MEDDOCNA	1112	name_doc	Text	24
OBJECT_MEDDIANA	1113	name_dia	Text	24
OBJECT_LINKS	1114	left	Text	10
OBJECT_VERID	1115	version	Text	10
OBJECT_LOCKUSER	1116	lockuser	Text	10
OBJECT_SYSTEMID	1117	systemid	Text	10
OBJECT_MODIFYTIME	1118	modifytime	Timestamp	10
OBJECT_MODIFYUSER	1119	modifyuser	Text	256
OBJECT_FOREIGNID	1124	foreignid	Text	10
OBJECT_USERGUID	1125	osowner	Text	32
OBJECT_DELETED	1126	deleted	Text	10
OBJECT_INDEXHISTFLAGS	1127	indexhistflags	Text	10
OBJECT_DOCHISTFLAGS	1128	dochistflags	Text	10
OBJECT_OSSD	1129	ossd	Text	32
OBJECT_MIMETYPEID	1900	mimetypeid	Text	10
OBJECT_FILESIZE	1902	filesize	Text	10
OBJECT_RETENTION_PLANNED	1903	retention_planned	Date	10
OBJECT_RETENTION	1904	retention	Date	10
STAMM_ID	1000	id	Text	10
STAMM_TIME	1001	timestamp	Timestamp	10
STAMM_LINKS	1002	left	Text	10
REG_ID	1120	id	Text	10
REG_STAID	1121	stamm_id	Text	10
REG_PARID	1122	parent_id	Text	10
SDSTA_ID	1130	stamm_id	Text	10
SDOBJ_ID	1131	object_id	Text	10
SDOBJTYPE	1132	objecttype	Text	10
SDREG_ID	1133	register	Text	10
SDDEL	1134	delete	Text	10

SDTIME	1135	time stamp	Text	10
SDREG_TYPE	1136	regtype	Text	10
FOLDERID	1181	folderid	Text	10
FOLDERTYPE	1182	foldertype	Text	10
REGISTERID	1183	registerid	Text	10
REGISTERTYPE	1184	registertype	Text	10
PARENTREGID	1185	parentregid	Text	10
PARENTREGTYPE	1186	parentregtype	Text	10
MDDEL	1140	delete	Text	5
MDTIME	1141	timestamp	Timestamp	10
MDMAP_ID	1142	mappe_id	Text	10
MDSTA_ID	1143	stamm_id	Text	10
MDOBJ_ID	1144	object_id	Text	10
MDOBJTYPE	1145	objecttype	Text	10
MDMOD	1146	modul	Text	5
MDIN	1147	inbox	Text	10
MDOUT	1148	ausgang	Text	10
MDCOUNT	1149	count	Text	5

Date formats

Formatting instructions are introduced with a percent sign (%). Character strings which do not start with % will be copied to the result string without any changes. The following formatting instructions can be used:

Formatting	Description
%a	abbreviated name of the week day
%A	name of the week day
%b	abbreviated name of the month
%B	name of the month
%c	Date and time according to the local settings
%d	Day of the month, numeric (01-31)
%j	Day of the year, numeric (001-366)
%m	Month, numeric (01-12)
%U	Calendar week (with Sunday as the first day of the week) (00-53)
%w	Week day, numeric (0-6; Sunday is 0)
%W	Week day, numeric (0-6; Monday is 0)
%x	Date according to the local settings
%X	Time according to the local settings
%y	Two-digit year (00-99)

%Y	Four-digit year
%z, %Z	(Abbreviated) name of the time zone; empty if time zone unknown
%%	Percent sign

Security system

§ DMS.CheckPermission

§ DMS.CheckPermissions

§ DMS.CopySD

§ DMS.CreateSD

§ DMS.DeleteSD

§ DMS.ReadSD

§ DMS.SetSD

Detailed Description

With enaio® version 4.50, you are provided with an additional security system at object level (SSOL) besides the existing rights system. With this system a so-called Security Descriptor (SD) can be created for every object, which refers to an Access Control List (ACL), i.e. a list of access control entries. Access control entries allow to specify for users or a user group access authorizations needed to edit index data or edit, delete and export objects.

Jobs for editing access structures in this security system are implemented in the DMS Executor. A job also exists with DMS.CheckPermission that checks access rights to a specific object independently of the used rights system.

List of Access Control Entries in XML format

XML is used to describe the list of access control entries. The jobs DMS.CreateSD, DMS.ReadSD, and SMS.SetSD use the same XML schema, which can be called with the job DMS.GetXMLSchema.

Example:

List of access control entries in XML format (DMSAccess).

```
<DMSAccess timestamp="" version="4.50">
<ACL ossd="" object_type="" object_id="">
<UserACE modify_index="0" modify_object="0" delete_object="0"
export_object="0" osuid=""/>
<GroupACE modify_index="0" modify_object="0" delete_object="0"
export_object="0" osgid=""/>
</ACL>
</DMSAccess>
```

Explanation of the <DMSAccess> attributes:

§ timestamp: creation time of the Access Control List (format: YYYY/MM/DDTHH:MM:SS)

§ version: product version number

Explanation of the <ACL> attributes:

§ ossd (STRING): GUID of the Security Descriptor

§ obj_type (long): Object type

§ obj_id (long): ID of the object instance

Explanation of the <UserACE> or <GroupACE> attributes:

- osuid (STRING): GUID of the user
- osgid (STRING): GUID of the user group
- modify_index (long): access type write index data
  - 0 = not set
  - 1 = allowed
  - 2 = not allowed
- modify_object (long): access type edit object
  - 0 = not set
  - 1 = allowed
  - 2 = not allowed
- delete_object (long): access type delete object
  - 0 = not set
  - 1 = allowed
  - 2 = not allowed
- export_object (long): access type export object
  - 0 = not set
  - 1 = allowed
  - 2 = not allowed

Glossary

- **SSOL -** Security System at Object Level
- **ACE -** Access Control Entry: contains the allowed or denied access types for a user or a user group
- **ACL -** Access Control List: list of all ACEs assigned to an object
- **SD -** Security Descriptor: a security descriptor, which identifies an ACL, can be created for every object. SSOL (see above) applies to objects with SD, the existing rights system applies to objects without SD.

DMS.CheckPermission

Description:

With this job, access permissions for single DMS objects can be checked independently of the used security system. To check whether it is allowed to insert an object at a specific location, the location has to be defined with the FolderID or RegisterID parameter and set to '0' with the ObjectID and the access type 'W' checked.

Parameter:

Flags (INT): not currently supported-> transfer 0

Access (STRING): access type to be checked (e.g. 'RWXDU' checks all access types)

- R = read index data
- W = write index data

- § X = open/execute object
- § D = delete object
- § U = write object
- § ObjectType (INT): Object type

ObjectID (INT): ID of the object

[RegisterType] (INT): type of parent register

- § 0 = in no register (directly on the folder level)
- § -1 = independent of registers

[RegisterID] (INT): ID of the parent register

[FolderID] (INT): ID of the folder

Return values:

Access (STRING): permitted access types (format corresponds the 'Access' input parameter)

DMS.CheckPermissions

Description:

With this job, access permissions can be checked for a list of DMS objects. As this job is aimed at performance, no filing tray objects and no inactive variants are verified.

Parameter:

Flags (INT): not currently supported-> transfer 0

Access (STRING): access type to be checked (e.g. 'RWXDU' checks all access types)

- § R = read index data
- § W = write index data
- § X = open/execute object
- § D = delete object
- § U = write object

ObjectType<n> (INT): Object type. <n> is a consecutive number beginning with 1.

ObjectList<n> (String): comma-separated list of the object IDs of the type ObjectType<n>

[RegisterType] (INT): type of parent register

- § 0 = in no register (directly on the folder level)
- § -1 = independent of registers

[RegisterID] (INT): ID of the parent register

[FolderID] (INT): ID of the folder

Return values:

ObjectType<n> (INT): n-th object type

ObjectList<n> (String): comma-separated list of the IDs and the determined rights for the object type n. The object ID is separated from the determined rights by a colon. See example below.

Example:

<u>Call:</u>
Flags=0

Access =WRXDU

FolderID=380

FolderType=1

RegisterID=0

RegisterType=0

ObjectType1=393216

ObjectList1=65493

ObjectType2=131072

ObjectList2=72272,72273,72274

<u>Return:</u>

ObjectType1=131072

ObjectList1=72272:RWXUD,72273:RW---,72274:R-X--

ObjectType2=393216

ObjectList2=65493:RWXUD

Note:

The order of object types and of objects in an ID list does not have to correspond to the order in the input parameters.

DMS.CopySD

Description:

This job assigns a copy of access control entries of an object to one or multiple other objects. When copying, access rights are transferred to target objects only for those users and user groups for whom no entry exists in the corresponding target object. On the other hand, existing access rights of users for whom no entries exist in the source object will not be concerned.

Parameter:

Flags (INT): not currently supported-> transfer 0

ObjectID (INT): ID of the output object

ObjectType (INT): type of output object

Destination (STRING): list of target objects in XML format

Example:

Structure of the destination

```
<DMSAccess>
<ACL object_type="XXX1" object_id="YYY1"/>
<ACL object_type="XXX2" object_id="YYY2"/>
<ACL object_type="XXX3" object_id="YYY3"/>
```

```
</DMSAccess>
```

Note:

Detailed description of the destination

§ object_type: Object type

§ object_id: ID of the object

DMS.CreateSD

Description:

This job creates a Security Descriptor for each object instance defined in the parameter 'XmlInfo.'

Parameter:

Flags (INT): not currently supported-> transfer 0

XmlInfo (STRING): list of objects in XML format for which security descriptors are to be created

Return values:

XmlInfo: (STRING): ACL list in XML format

Example:

Structure of the XmlInfo input parameter

```
<DMSAccess>
<ACL obj_type="1" obj_id="61967" />
<ACL obj_type="1" obj_id="61968" />
</DMSAccess>
```

Example:

returned ACL list in XML format (description)

see ACL-XML-Schema)

```
<DMSAccess timestamp="2004/04/08T12:59:25" version="4.50">
<ACL ossd="6DBEC785D3CEB4D894B" obj_type="1" obj_id="61967" />
<ACL ossd="2AB8AB82E0CEB4D8BDD" obj_type="1" obj_id="61968" />
</DMSAccess>
```

See also:

DMS.SetSD

DMS.DeleteSD

Description:

Use this job to delete the Access control entry of a user/group or the whole access control list for a specified object.

Parameter:

Flags (INT): flags can have the following values

§ 0 – the object is specified with the 'ObjectType' and 'ObjectId' parameters The SD of the object and thereby all ACEs for this object will be deleted.

§ 1 – the security descriptor (and thus the object) is specified with the 'Ossd' parameter. All ACEs which belong to a user/user group marked by the 'Osuid' parameter will be deleted.

- 2 – the object is specified with the 'ObjectType' and 'ObjectId' parameters All ACEs which belong to a user/user group marked by the 'Osuid' parameter will be deleted.
- ObjectType (INT): Object type

ObjectID (INT): ID of the object instance

Ossd (STRING): GUID of the security descriptor

Osuid (STRING): GUID of the user/user group whose access rights are to be deleted

DMS.ReadSD

Description:

This job returns the ACEs of a user/ user group or all users/user groups for a given object in XML format.

Parameter:

Flags (INT): 0 = ACL of the object is returned; 1 = ACE for users/user groups is returned. (See parameter 'Osuid')

ObjectId (INT): ID of the object instance

ObjectType (INT): Object type

[Osuid] (STRING): users or groups GUID

Return values:

XmlInfo (BASE64): ACL in XML format

Example:

Structure of XmlInfo (description

see DMSAccess)

```
<DMSAccess>
<ACL ossd="XXX1">
<UserACE osuid="YYY1" modify_object="1" export_object="1"/>
<UserACE osuid="YYY2" modify_object="2" export_object="1"/>
<GroupACE osuid="YYY3" export_object="1"/>
</ACL>
</DMSAccess>
```

DMS.SetSD

Description:

This job creates one or multiple Access control entries.

Parameter:

Flags (INT): not currently supported-> transfer 0

XmlInfo (BASE64): Xml-formatted string containing the ACEs

Example:

Structure of XmlInfo (description

see DMSAccess)

```
<DMSAccess>
<ACL ossd="XXX1">
```

```
<UserACE osuid="YYY1" modify_object="1" export_object="1"/>
</ACL>
<ACL ossd="XXX2">
<UserACE osuid="YYY1" modify_object="1" export_object="1"/>
<UserACE osuid="YYY2" modify_object="2" export_object="1"/>
</ACL>
<ACL ossd="XXX3">
<GroupACE osuid="YYY3" export_object="1"/>
</ACL>
</DMSAccess>
```

Relations and Relation Texts

- § DMS.AddRel
- § DMS.AddRelText
- § DMS.AddRelTextLang
- § DMS.DelRel
- § DMS.DelRelText
- § DMS.ModRel
- § DMS.ModRelText
- § DMS.ModRelTextLang
- § DMS.RetrieveRelations
- § DMS.RetrieveRelTexts

Detailed Description

Objects can be linked with relations from version 4.50. The relation between two objects is given a predefined name (relation text) in this case. Each pair of these relations can be defined by object types.

Relations in XML format

Example:

```
<Relations>
<Relation obj_id1="1" obj_typ1="2" obj_id2="3" obj_typ2="4"
valid_from="3443234" valid_to="433444"
ensured="1" checked="8"
reltextid="AF32988234DEFA78979879"
IObjId="9" IObjType="10" >
</Relation>
</Relations>
```

Comment:

The root element can be dropped, if only one entry has been defined.

Description of the attributes of the element 'RelationXML'

- § obj_id1 (LONG): ID of the 1st object
- § obj_typ1 (LONG): type of the 1st object
- § obj_id2 (LONG): ID of the 2nd object
- § obj_typ2 (LONG): type of the 2nd object
- § valid_from (LONG): time stamp for validity start date
- § valid_to (LONG): time stamp for validity end date

- ensured (LONG): Flag: 1=ensured, 0=not ensured
- checked (LONG): Flag: 1=checked, 0=unchecked
- created (LONG): creation timestamp
- created_to (LONG): creation end timestamp (only relevant for search request)
- modified (LONG): timestamp of the last modification
- modified_to(LONG): end timestamp of the last modification (only relevant for search request)
- deletetime (LONG): timestamp for the delete time
- deletetime _to (LONG): end timestamp for the delete time (only relevant for search request)
- deleteuser (STRING): name of the user who deleted the relation from the system
- relid (STRING): GUID of the relation
- reltextid (STRING): reference to the relation text (GUID)
- iobjid (LONG): ID of the object forming the basis for the relation
- iobjtype (LONG): type of the object forming the basis for the relation

Relation texts in XML format

Example:

```
<RelTexts>
<RelText  shortrel="property"
active="1"
objtype1="2"
objtype2="4"
to="is property of"
reverse="is property of"
langid="9"/>
</RelTexts>
```

Remarks:

The <RelTexts> root element can be dropped if only one <RelText> entry has been defined.

Explanation of the <RelText> attributes:

- shortrel (STRING): short description of the relation text (max. 32 characters)
- active (INT): activity status 0=not active, 1=active
- objtype1 (INT): ID of the 1st object
- objtype2 (INT): ID of the 2nd object
- to (STRING):description of the relation between object type 1 and object type 2
- reverse (STRING):description of the relation between object type 2 and object type 1
- langid (INT): language ID (see Language code)

Language codes

DMS Executor supports jobs in multiple languages. The languages have to be configured before in the enaio® system. 'Windows Language Code Identificators' are used as language codes, however, only the main languages are supported. The following table contains the first 31 language codes (hexadecimal):

ID	Language	ID	Language

0x01	Arabic
0x02	Bulgarian
0x03	Catalan
0x04	Chinese
0x05	Czech
0x06	Danish
0x07	German
0x08	Greek
0x09	English
0x0A	Spanish
0x0B	Finnish
0x0C	French
0x0D	Hebrew
0x0E	Hungarian
0x0F	Icelandic
0x10	Italian

0x11	Japanese
0x12	Korean
0x13	Dutch
0x14	Norwegian
0x15	Polish
0x16	Portuguese
0x17	Arabic
0x18	Romanian
0x19	Russian
0x1A	Croatian/Serbian
0x1B	Slovak
0x1C	Albanian
0x1D	Swedish
0x1E	Thai
0x1F	Turkish

DMS.AddRel

Description:

This job creates a relation between two objects.

Parameter:

Flags (INT): not currently supported-> transfer 0

RelationXML (BASE64): relation properties in XML format

Return values:

RelID (STRING): GUID that has been created as ID for this link

Example:

Structure of RelationXML (description:

see XMLRELATION)

```
<Relation obj_id1="1" obj_typ1="2" obj_id2="3" obj_typ2="4"
valid_from="3443234" valid_to="433444"
ensured="1" checked="0"
reltextid="AF32988234DEFA78979879"
IObjId="9" IObjType="10" >
</Relation>
```

See also:

DMS.AddRelText, DMS.AddRelTextLang

DMS.AddRelText

Description:

This job creates a relation text in a default language.

Parameter:

Flags (INT): not currently supported-> transfer 0

RelTextXML (BASE64): relation text and relation properties in XML format

Return:

RelTextID (STRING): GUID defined as ID for this relation entry

Example:

Structure of RelTextXML (description:

see XMLRELTEXT)

```
<RelText shortrel="property" active="1" objtype1="2" objtype2="4"
to="is properties of" reverse="is property of">
</RelText>
```

DMS.AddRelTextLang

Description:

This job adds a relation text in a different language to an existing relation text entry.

Parameter:

Flags (INT): not currently supported-> transfer 0

RelTextXML (BASE64): relation text and relation properties in XML format

Return:

RelTextID (STRING): GUID defined as ID for this relation entry

Example:

Structure of RelTextXML (description:

see XMLRELTEXT)

```
<RelText reltextid="ABC79324DEF879342" langid="9" to="is owner"
reverse="is owned by">
</RelText>
```

See also:

DMS.AddRelText

DMS.DelRel

Description:

This job deletes the specified relation.

Parameter:

Flags (INT): not currently supported-> transfer 0

RelID (STRING): GUID of the relation to be deleted

See also:

DMS.DelRelText

DMS.DelRelText

Description:

This job deletes a relation text. If they are still references to a relation text, an error will be created.

Parameter:

Flags (INT): not currently supported-> transfer 0

RelTextID (STRING): GUID of the relation text which to be deleted

LangID (INT): language code of the text to be deleted (see Language codes) (0 = relation text of all languages will be deleted)

DMS.ModRel

Description:

This job changes the properties of relation entry. Following relation attributes can be changed:

§ reltextid

§ ensured

§ checked

§ valid_from

§ valid_to

§ iobjid

§ iobjtype

Parameter:

Flags (INT): not currently supported-> transfer 0

RelationXML (BASE64): Relations and the properties to be changed in XML format

Example:

Structure of RelationXML (description:

see XMLRELATION)

```
<DMSRelations>
<Relation relid="D2A0988234DEFA78979879" valid_from="3443234"
valid_to="433444" ensured="1" checked="8"
reltextid="AF32988234DEFA78979879" IObjId="9" IObjType="10">
</Relation>
</DMSRelations>
```

DMS.ModRelText

Description:

This job changes the properties of one or multiple relation texts. If the text has to be changed in a language other than the default language, the DMS.ModRelTextLang job has to be chosen.

Parameter:

Flags (INT): not currently supported-> transfer 0

RelTextXML (BASE64): relation texts in XML format

Example:

Structure of RelTextXML (description:

see XMLRELTEXT)

```
<RelText reltextid="ABC79324DEF879342" shortrel="property" active="1"
objtype1="2" objtype2="4" to="is properties of" reverse="is
property of">
</RelText>
```

DMS.ModRelTextLang

Description:

This job changes the relation text for a language specified in XML.

Parameter:

Flags (INT): not currently supported-> transfer 0

RelTextXML (BASE64): relation texts in XML format

Example:

Structure of RelTextXML (description:

see XMLRELTEXT)

```
<RelText reltextid="ABC79324DEF879342" langid="9" to="is owner"
reverse="is owned by">
</RelText>
```

DMS.RetrieveRelations

Description:

This job can be used to search for relations. The search criteria are defined with XML attributes. An upper limited can be specified for the time attributes 'created', 'modified' and 'deletetime' with the XML attributes 'created_to', 'modified_to' and 'delete_to'. It is then searched in the particular time period. Using '-1' means that no upper or lower limit will be used.

Parameter:

Flags (INT): not currently supported-> transfer 0

RelationXML (BASE64): search parameter in XML format

Created_to (INT): search parameter to limit the creation time

Modifytime_to (INT): search parameter to limit the time of the last modification

Deleted_to (INT): search parameter to limit the time of deletion

Direction (INT): flag which indicates whether the search is to be done in the reverse direction, if object type 1 and/or object type 2 are indicated in the XML. 0 = unidirectional, 1=bidirectional search

Return:

RelationsXML (BASE64): all relations in XML format that fulfill the search criteria

Example:

Structure of RelationXML (description:

see XMLRELATION)

```
<Relation obj_id1="" obj_typ1="" obj_id2="" obj_typ2="" valid_from=""
valid_to="" ensured="" checked="" reltextid="" created=”-1” created_to=”4322344”
IObjId=""
IObjType="" >
</Relation>
```

Example:

Structure of RelationXML (description:

see XMLRELATION)

```
<Relations>
<Relation obj_id1="" obj_typ1="" obj_id2="" obj_typ2=""
valid_from="" valid_to="" ensured="" checked="" reltextid=""
IObjId="" IObjType="" >
</Relation>
<Relation obj_id1="" obj_typ1="" obj_id2="" obj_typ2=""
valid_from="" valid_to="" ensured="" checked="" reltextid=""
IObjId="" IObjType="" >
</Relation>
</Relations>
```

DMS.RetrieveRelTexts

Description:

This job returns all relation texts which correspond to the passed search criteria.

Parameter:

Flags (INT): not currently supported-> transfer 0

RelTextXML (BASE64): search criteria in XML format

Return values:

RelTextsXML (BASE64): list of all relation texts that correspond to the search criteria in XML format

Example:

Structure of RelTextXML (description:

see XMLRELTEXT)

```
<RelText shortrel="" active="" objtype1="" objtype2="" to="" reverse=""
langid=""/>
```

Example:

Structure of RelTextsXML (description:

see XMLRELTEXT)

```
<RelTexts>
<RelText reltextid="88E28DCDA7414D0B82AC2AFD0CB354EC"
shortrel="property" langid="7"
active="1" objtype1="2" objtype2="4" to="is properties of"
reverse="is property of"></RelText>
<RelText reltextid="79E28DCDA7414D0B82AC2AFDASB354EC"
shortrel="document" langid="7"
active="1" objtype1="5" objtype2="7" to="is document of"
reverse="is document of"></RelText>
```

```
</RelTexts>
```

Portfolios

§ DMS.AddPortfolio

§ DMS.DelPortfolio

§ DMS.ModPortfolio

§ DMS.RemoveFromPortfolio

§ DMS.RetrievePortfolios

Detailed Description

These jobs are used to search and edit portfolios.

Portfolio XML Format

Example:

```
<Portfolios>
<Portfolio id="123" created="135233432" creator=""
recipient="" subject="">
<Objects>
<Object objecttype_id="13072" id="12">
</Object>
<Object>
</Object>
</Objects>
</Portfolio>
</Portfolios>
```

Description of elements and attributes

Element 'Portfolios'**:** list of portfolios (the root element can be omitted if only one entry has been defined.)

Attributes of the element 'Portfolio'**:**

§ id (long): Portfolio ID

§ created (long): creation timestamp

§ creator (STRING): Name of creator

§ recipient (STRING): Receiver

§ subject (STRING): Title of the portfolio

Element 'Objects'**:** list of objects (folders, registers, documents) that the portfolio contains

Attributes of the element 'OBJECT'**:**

§ objecttype_id (LONG): Object type

§ id (LONG): ID of the object instance

DMS.Addportfolio

Description:

This job creates a new portfolio.

Parameter:

Flags (INT): not currently supported-> transfer 0

PortfolioXML (BASE64): description of the portfolio in XML format

Mode (INT): 0 (default) = searches for available portfolios using the recipient or subject. If one or more portfolios are available, this or the first portfolio is used. 1=Creates a new portfolio.

Return values:

portfolio (INT): ID of the new portfolio (-1 = job failed)

Example:

Structure of portfolioXML (description:

see Portfolio XML Format)

```
<Portfolios>
<Portfolio created="" creator="" recipient=""
subject="">
</Portfolio>
</Portfolios>
```

DMS.Delportfolio

Description:

This job deletes a portfolio. The portfolio which has to be deleted can be identified by its ID, the recipient (recipient ID) and the title (subject) of the portfolio.

Parameter:

Flags (INT): not currently supported-> transfer 0

PortfolioXML (BASE64): description of the portfolio in XML format

Example:

Structure of portfolioXML (description:

see Portfolio XML Format)

```
<Portfolio id="">
</Portfolio>

<!--or-->

<Portfolio recipient="" subject="">
</Portfolio>
```

DMS.RemoveFromportfolio

Description:

This job deletes objects in the portfolio.

Parameter:

Flags (INT): not currently supported-> transfer 0

PortfolioXML (BASE64): description of the portfolio in XML format

Example:

Structure of portfolioXML (description:

see Portfolio XML Format)

```
<Portfolio id="" created="" creator="" recipient=""
subject="">
<Objects>
<Object objecttype_id="" id="">
</Object>
</Objects>
</Portfolio>
```

DMS.Modportfolio

Description:

This job adds objects to an existing portfolio or deletes all objects of the portfolio.

Parameter:

Flags (INT): not currently supported-> transfer 0

Mode (INT): 1 = all objects of the portfolio will be deleted, otherwise 0.

PortfolioXML (BASE64): description of the portfolio in XML format

Example:

Structure of portfolioXML (description:

see Portfolio XML Format)

```
<Portfolio id="" created="" creator="" recipient=""
subject="">
<Objects>
<Object objecttype_id="" id="">
</Object>
</Objects>
</Portfolio>
```

DMS.Retrieveportfolios

Description:

This job returns all portfolios which match the specified search criteria.

Parameter:

Flags (INT): not currently supported-> transfer 0

Created_to (Long): time stamp up to when the portfolio was created. 0=unlimited

PortfolioXML (BASE64): search criteria for portfolio in XML format

GarbageMode (INT): (optional) 1=only objects from the trash can are taken into account. 0=objects from the trash can are not taken into account.

Return values:

PortfoliosXML (BASE64): list of all portfolios found in XML format

Example:

Structure of portfolioXML (description:

see Portfolio XML Format)

```
<Portfolio id="" created="" creator="" recipient=""
subject="">
</Portfolio>
```

User-Related data

- § DMS.DeleteUserData
- § DMS.GetUserData
- § DMS.GetUserDataNames
- § DMS.IsUserData
- § DMS.SetUserData

Detailed Description

The jobs in this section are used to administer any user-related data. Each data entry is indicated by a user defined name (maximum length 100), a type (see below) and a user ID. The user ID is determined by the jobs automatically based on the user ID. The value of the entry is written into a BLOB field and can therefore contain any content.

With the help of the type, the kind of data is defined. The table below (last update: 2004/03/18) contains an overview of the numbers that have already been assigned.

Warning: the type cannot be selected. If a new type is required, it must be registered first of all.

Type	**Description**
1	saved search request
2	External Programs
3	AS.INI
4	folder structure under 'desktop'
5	Extended queries
6	data from current aslisten.dat
7	AS object in desktop area
8	entries for list fields
9	reserved
10	Import configuration dBase III
12	Import configuration dBase IV
13	Import configuration dBase V
13	Import configuration ASCII with separators
14	Import configuration ASCII fixed field length
15	reserved
16	reserved
17	Import configuration XML linear
18	Import configuration MS Access
19	Import configuration Excel 3

20	Import configuration Excel 4
21	Import configuration Excel 5
22	Import configuration Excel 8
23	external import configuration
24	Import configuration ODBC
25-30	reserved for import configurations
31	user/group export configurations
32	user/group import configurations
33	Automatically delete configuration from user trash cans
50	Client/Index scan configuration
51	Stamp templates for layers
52	Next position for public saved search requests
53	WF add-on favorites
54	Window gadget settings
80-85	User-related configuration information for additional applications

DMS.DeleteUserData

Description:

This job deletes the user-related dataset which was specified by name and type.

Parameter:

Flags (INT): not currently supported-> transfer 0

Type (INT): see Data Types

Name (STING): identifier of the dataset

DMS.GetUserData

Description:

This job reads user related data for the specified name and type from the database.

Parameter:

Flags (INT): not currently supported-> transfer 0

Type (INT): see Data Types

Name (STRING): Name (STRING): identifier of the dataset

Return values:

IsUserData (INT): 1 = if the requested entry exists, otherwise 0

Value (BASE64): requested value

See also:

DMS.GetUserDataNames, DMS.SetUserData, DMS.DeleteUserData

DMS.GetUserDataNames

Description:

This job lists the names of all entries of the given type for the logged-in user .

Parameter:

Flags (INT): not currently supported-> transfer 0

Type (INT): see Data Types

Return values:

Name[1..n] (STRING): entry name

See also:

DMS.GetUserData

DMS.IsUserData

Description:

This job verifies whether an entry exists already for the indicated name and type. By default, the existence is only verified for the user. If the check has to be carried out independently of the user – i.e. for the type only – the corresponding flag has to be set.

Parameter:

Flags (INT): 1 = the existence of the entry is verified depending on the user, otherwise 0.

Type (INT): see Data Types

Name (STRING): Name (STRING): identifier of the dataset

Return values:

IsUserData (INT): 1 = if the requested entry exists, otherwise 0

DMS.SetUserData

Description:

This job saves user-related data in the database. If no entry for the specified name and type exists, a new entry is created. The check to ascertain whether an entry already exists can be done using the job DMS.IsUserData

Parameter:

Flags (INT): reserved, must be 0

Type (INT): see Data Types

Name (STRING): Name (STRING): identifier of the dataset

Value (BASE64): value to be saved

DMS.GetXMLSchema

Description:

This job returns the specified XML schema as a file.

Parameter:

Flags (INT): must be 0

Schema (STRING): name of the schema

§ DMSData = Schema for the import

§ DMSQuery = schema for DMS search requests

§ DMSContent = Schema for DMS results

§ DMSAccess = Schema for SSOL jobs

§ DMSObjDef = Schema for the object definition

§ Relation = Schema for relations

§ RelText = Schema for relation texts

§ portfolio = Schema for portfolios

Return values:

File list: name and path of the XSD file

Medical Engine (Namespace med)

The medical engine provides access to medical information. The current task of the jobs is to summarize and return laboratory values based on the LOINC system (see http://www.loinc.org/).

Please refer to the specification HL7 version 3 (see http://www.hl7.org/) for the jobs SaveMedicalRecord, GetMedicalRecord, and NotifyMedicalRecord.

§ LoincResults

§ LoincObservations

§ LoincUnits

§ LoincViewSets

§ PatientData

§ CreateLaboratoryReport

§ UpdatePatientId

§ UpdateVisitId

§ ObservationInsert

§ ObservationRequestHistory

§ ObservationResultHistory

§ ObservationValues

§ SaveMedicalRecord

§ GetMedicalRecord

§ NotifyMedicalRecord

§ GetSystemOID

Field Label

The individual field labels are organized in groups. These groups correspond to the tables of the underlying database. If one or multiple field labels are indicated for the 'FieldSelectors' parameter in the job, only these will be taken into account. If no field label is indicated, all values of the group will be returned. Single field labels can be found in the result document at the same level as the XML result.

Observations Group

Master data of all examinations, e.g. laboratory results and vital signs or ECG, are summarized in this group. Coding is done according to the LOINC standard! (leading Observation-ID->LoincID). In ObservationResults it is referenced to observations (with LOINC-ID).

FieldSelector	**Description**
ObsID	LOINC-ID
ObsLastupdatedTS	Time of the last modification in form of a time stamp.
ObsName	Examiniation name
ObsStdValType_ID	Default data format of the examination result (-> mdObsValueTypes)

ObsStdUnit_ID	default unit of measurement (-> units)
ObsStdRefRange	Standard set of values (= set of values between OBS-STDMINVALUE and OBS_STDMAXVALUE) based on the default unit of measurement (only relevant for numeric value types obsOvtObjectID). Alternative names from the LOINC default database (separated by ;).
ObsStdMinValue	Lower limit for the standard set of values based on the default unit of measurement (only relevant for numeric value types obsOvtObjectID).
ObsStdMaxValue	Upper limit for the standard set of values based on the default unit of measurement (only relevant for numeric value types obsOvtObjectID).
ObsPosition	Display position (esp. use in axvbPatCurve.dll). Consecutive numbering in steps of 10.
ObsColor	Display color as Hex-RGB value (esp. use in axvbPatCurve.dll).
ObsScsClass	Classification of examinations according to the SCS system.
ObsLnComponent	1. Part of the Loinc name. Syntax: [analyte].[subclass].[sub-subclass]^[time delay] post [amount] [substance] [route]^adjustment. **Note:** LoincName=[LnComponent]:[LnProperty]:[LnTimeAspct]:[LnSystem]:[LnScaleType]:[LnMethodType]
ObsLnProperty	2. Part of the Loinc name. Physical measurement category. Corresponds with the LOINC table Properties.
ObsLnTimeAspct	3. Part of the Loinc name. Measurement scenario Syntax [TimeAspect]^[Modifier] first part corresponds with the LOINC table TimeAspects.
ObsLnSystem	4. Part of the Loinc name. Examination object Syntax [System]^[Supersystem] if super system is not explicitly coded -> default super system = patient
ObsLnScaleType	5. Part of the Loinc name. Scale type (continuous or discontinuous). Corresponds with the ScaleTypes table
ObsLnMethodType	6. Part of the Loinc name. Measurement method Codes are self-explanatory, otherwise refer to table 14 in LOINCManual.pdf
ObsLnRelatedNames	original alternative names from the LOINC standard database (separated by ;)
ObsLnClass	LOINC classification Corresponds with the Classes-Class Types LOINC table.
ObsLnClassType	Corresponds to LOINC table class types: 1 - Laboratory Class, 2 - Clinical Class, 3 - Claims Attachments, 4 - Surveys
ObsIUPAC	Alternative IUPAC coding
ObsMolarMass	Molar mass (only relevant for determination of molecules)

Group Obsrequests

Data of the laboratory search request is summarized in this group. Single queries are always subordinate to a patient's stay. The results can be found in tdObservationResults. Results (e.g. laboratory findings) transferred from sub systems by HL7 are to be inserted as valid. Possiblities to change data depend on the status.

FieldSelector	**Description**
ObrID	unique ObjectID in terms of a PrimaryKey (GUID)
ObrLastUpdatedTS	Timestamp of the last modification
ObrState	currently invalid
ObrCreationDT	Time of creation
ObrCreationUser	User, who created
ObrReleaseDT	Time of implementation
ObrCancelDT	Time of cancelation
ObrCancelUser	User who canceled
ObrCancelReason	Reason for cancelation
ObrCommonOrder_ID	Reference to the optionally related order (-> tdCommonOrders). Warning: no enforcement of referential integrity for this relation!
ObrPlacerOrderID	Order number of the ordering party (only relevant in connection with order/entry tdCommonOrders)
ObrFillerOrderID	Editing number of the service center and HL7 key field for identifying an examination.
ObrObservationDT	Time of examination or time when samples were taken
ObrObsOrderSet_ID	Standard set LOINC ID (-> mdObsOrderSets) Represents one or multiple examination(s). The latter is also called profile, set or collection process (e.g. full blood count).
ObrArchiveDocID	Reference to the diagnostic findings in the archive (ObjectID)

Group Obsresults

Examination results and diagnostic findings are summarized in this group. This object is subordinate to an examination requirement/report which also contains the time of the examination. Warning: options to change data depend on the state of the parent ObservationRequest!

FieldSelector	**Description**
ObxID	unique ObjectID in terms of a PrimaryKey (GUID)
ObxLastUpdatedTS	Timestamp of the last modification
ObxState	Approval status -> moReleaseStateEnum
ObxObservation_ID	LOINC-ID of the examination (-> mdObservations)
ObxObservation_CE	Examination name as copied HL7-CodedElement: identifier^text^coding

	system^Alternate identifier^alternate text^alternate coding system
ObxSubObs_ID	
ObxValueType_ID	Data format of the Value field (-> mdObsvalueTypes)
ObxValue	Result or measured value according to the format: obxOvtObjectID
ObxUnit_ID	used unit of measurement (-> units)
ObxRefRange	Reference area as text in the original HL7 format (refer to OBX 7)
ObxMinValue	Lower limit for the standard set of values (only relevant for numeric value types obxOvtObjectID)
ObxMaxValue	Upper limit for the standard set of values (only relevant for numeric value types obxOvtObjectID)
ObxAbnormFlag_ID	Evaluation value in relation to the normal range. (-> mdObsAbnormalFlags). Warning: Only the first transferred AbnormalFlag will be applied for an ObsResult imported via HL7.
ObxResultState_ID	Findings result (-> mdObsResultStates)
ObxFootnoteID	User comment: Footnote ID (special use in axvbPatCurve.dll).
ObxFootnoteText	User comment: footnote text (esp. use in axvbPatCurve.dll).
ObxComment	User comment (esp. use in axvbPatCurve.dll).
ObxCreationDT	Time of creation
ObxCreationUser	Creator
ObxReleaseDT	Time of implementation
ObxCancelDT	Time of cancelation
ObxCancelUser	User who canceled
ObxCancelReason	Reason for cancelation

Group Units

The measurement units used for quantification of the observations (tdObservationResults, mdObservations) are summarized in this group. (->See also compilation in table _PropertiesAndUnits_ of the LOINC-MDB). Conversions can be found in the Conversions table. 'ID' = HL7 name

FieldSelector	Description
UntID	ID compliant with HL7 2.4 specifications
UntLastupdatedTS	Timestamp of the last modification
UntName	Lengthy name of the unit of measurement
UntAbbreviation	common abbreviation of the unit as to be displayed to the user

Group ViewSets

The Viewsets group is a hierarchical depiction of the Viewsets and associated Views. All identifiers which start with Ovws belong to the Viewset. Identifiers which start with Ovd are subordinate Views. ViewSets: contains examination sets arranged for display in axvbpatCurve.dll. One set can represent one or multiple examinations. The examinations linked with the set can be seen in mmt_obsviewingsetdt. Views: contains 1..N individual examinations belonging to a display set (mdObsViewingSets).

FieldSelector	**Description**
OvsID	unique ObjectID in terms of a PrimaryKey (GUID)
OvsLastUpdatedTS	Timestamp of the last modification
OvsName	Name of the display set
OvdID	unique ObjectID in terms of a PrimaryKey (GUID)
OvdLastUpdatedTS	Timestamp of the last modification
OvdPosition	Display position (1..N) of the examination
OvdObservationID	Examination LOINC ID (-> mdObservations)

med.LoincResults

Description:

This job returns the values of the LOINC system for patients and their stay. The available filters are described under the respective input parameters. All parameters are linked with a logical AND. If no parameters are specified, the call will return all results.

Result sorting:

1. PatientID
2. AccObjectID
3. VisitID
4. ObservationDT
5. CreationDT

Parameter:

[Visits] (STRING): contains a comma-separated list of ID for the stays (see Namespace)

[FieldSelectors] (STRING): contains a comma-separated list of the requested fields; (see: field labels)

[Namespace] (STRING): indicates the system from which the values for patients and stays originate; Empty = both parameters are expected

[ResultType] (STRING): indicates in which format the output parameters are provided (DEFAULT BASE64)

§ STRING = the result will be returned as string parameter

§ BASE64 = the result will be returned Base64-encoded

[Patients] (STRING): not currently implemented (contains a comma-separated list of IDs for patients (see Namespace)

[From] (STRING): not currently implemented (if From is specified, only the results from (inclusive) the specified time are returned. The ObservationDT of the request is taken into account. Format: YYYY/MM/DD)

[To] (STRING): not currently implemented (if To is specified, only the results until the specified time are returned. The ObservationDT of the query is taken into account. Please note that the indication of the date automatically sets the corresponding time to 00:00. If a time has been set for the corresponding data in the database, they will not be displayed as these data are then bigger than the searched date. Format: YYYY/MM/DD)

[Status] (STRING): not currently implemented (contains a comma-separated list of the status. Only LoincResults which have the indicated status will be returned. If 'fields' is not indicated, the LoincResults with status 1 will be returned. Possible values: (at the moment only checked by database results)

§ 0 - invalid dataset

§ 1 - current dataset

§ 2 - canceled dataset

[Extremes] (STRING): not currently implemented (min = returns the smallest value in each case for all requested fields; max = returns the largest value in each case for all requested fields)

Return values:

Result (STRING): returns the result of the search request as XML document

Example:

The following example has been called as value with 'ObsID,ObrID,ObxID,ObxValue,UntID'.

```
<med>
<Patients>
<Patient ID="252" CabinetType="3">
<Visit ID="253">
<Requests>
<OBR ID="E3C1787E-69C2-11D6-82E2-0000D19D9210">
<Results>
<OBX ID="E3C1787F-69C2-11D6-82E2-0000D19D9210" Value="28.0"/>
<OBX ID="E3C1789A-69C2-11D6-82E2-0000D19D9210" Value="neg"/>
<OBX ID="E3C1789B-69C2-11D6-82E2-0000D19D9210" Value="massh."/>
<OBX ID="E3C1789C-69C2-11D6-82E2-0000D19D9210" Value="neg"/>
<OBX ID="E3C1789D-69C2-11D6-82E2-0000D19D9210" Value="neg"/>
<OBX ID="E3C1789E-69C2-11D6-82E2-0000D19D9210" Value="(+)"/>
<OBX ID="E3C1789F-69C2-11D6-82E2-0000D19D9210" Value="Urate+"/>
<OBX ID="E3C178A0-69C2-11D6-82E2-0000D19D9210" Value="neg"/>
</Results>
</OBR>
<OBR ID="E3C18537-69C2-11D6-82E2-0000D19D9210">
<Results>
<OBX ID="E3C18538-69C2-11D6-82E2-0000D19D9210" Value="25.0"/>
<OBX ID="E3C18539-69C2-11D6-82E2-0000D19D9210" Value="neg"/>
<OBX ID="E3C1853A-69C2-11D6-82E2-0000D19D9210" Value="6.60"/>
<OBX ID="E3C1853E-69C2-11D6-82E2-0000D19D9210" Value="87.8"/>
<OBX ID="E3C1853F-69C2-11D6-82E2-0000D19D9210" Value="30.5"/>
<OBX ID="E3C18540-69C2-11D6-82E2-0000D19D9210" Value="237"/>
</Results>
</OBR>
<OBR ID="8775BE70-68EA-11D6-82E2-0000D19D9210">
<Results>
<OBX ID="8775BE72-68EA-11D6-82E2-0000D19D9210" Value="142"/>
<OBX ID="8775BE73-68EA-11D6-82E2-0000D19D9210" Value="4.15"/>
```

```
<OBX ID="8775BE74-68EA-11D6-82E2-0000D19D9210" Value="2.36"/>
</Results>
</OBR>
<OBR ID="E3C19CDA-69C2-11D6-82E2-0000D19D9210">
<Results>
<OBX ID="E3C19CDB-69C2-11D6-82E2-0000D19D9210" Value="0.900"/>
</Results>
</OBR>
</Requests>
</Visit>
</Patient>
</Patients>

<Observations>
<OBS ID="4537-7"/>
<OBS ID="1988-5"/>
<OBS ID="789-8"/>
<OBS ID="5787-7"/>
<OBS ID="5783-6"/>
<OBS ID="5769-5"/>
<OBS ID="8246-1"/>
</Observations>
<Units>
<Unit ID="mg/dL"/>
<Unit ID="Mill/cmm"/>
<Unit ID="mm n.W."/>
<Unit ID="mmol/L"/>
<Unit ID="mmol/l"/>
<Unit ID="pg"/>
<Unit ID="U/l"/>
<Unit ID="x1000/cmm"/>
</Units>
<ViewSets>
<ViewSet OvsID="615B8200-1635-4D89-B5EC-4977802D7719"
OvsLastUpdatedTS="1013087583" OvsName="Set Cholangiolithiasis">
<View OvdID="642345256923659265956346543656239562"
OvdLastUpdatedTS="1013087623" OvdPosition="130"
OvdObservationID="1988-5"/>
<View OvdID="385743657265762475643756783645726526"
OvdLastUpdatedTS="1013087623" OvdPosition="160"
OvdObservationID="2324-2"/>
<View OvdID="298436375638567834568734568734567834"
OvdLastUpdatedTS="1013087623" OvdPosition="180"
OvdObservationID="3040-3"/>
</ViewSet>
<ViewSet OvsID="00C00F47-176E-4A1E-B152-1D7C25865F71"
OvsLastUpdatedTS="1013087583" OvsName="Set heart attack">
<View OvdID="A795DBFE-4440-4248-A358-29D57DB02E90"
OvdLastUpdatedTS="1013087623" OvdPosition="10"
OvdObservationID="1920-8"/>
<View OvdID="A8D5D3A1-4EB3-492C-9CAB-E7908D217784"
OvdLastUpdatedTS="1013087623" OvdPosition="20"
OvdObservationID="1677-4"/>
<View OvdID="345346536536534625762357632756327465"
OvdLastUpdatedTS="1013087623" OvdPosition="60"
OvdObservationID="6598-7"/>
<View OvdID="094632784523856827345823568235872385"
OvdLastUpdatedTS="1013087623" OvdPosition="170"
OvdObservationID="1798-8"/>
</ViewSet>
</ViewSets>
</med>
```

med.LoincObservations

Description:

This job returns the master data of the observations of the LOINC system. The available filters are described under the respective input parameters. All parameters are linked with a logical AND. If no parameters are specified the call returns all observations.

Parameter:

[Visits] (STRING): comma-separated list of IDs for the visits (see Namespace)

[Namespace] (STRING): indicates the system from which the values for patients and stays originate; Empty = both parameters are expected

[ObsIDs] (STRING): contains a comma-separated list of the requested observations

[FieldSelectors] (STRING): contains a comma-separated list of the requested fields; see: field labels

[ResultType] (STRING): indicates in which format the output parameters are provided (DEFAULT BASE64)

§ STRING = the result will be returned as string parameter

§ BASE64 = the result will be returned Base64-encoded

[Patients] (STRING): not currently implemented (contains a comma-separated list of IDs for patients (see Namespace)

[ViewSets] (STRING): not currently implemented (if ShowSets is specified, the set for each observation are output in which the relevant observation is located (default value: false))

Return values:

Result (STRING): returns the result of the search request as XML document

med.LoincUnits

Description:

This job returns the master data of the units of the LOINC system. The available filters are described under the respective input parameters. All parameters are linked with a logical AND. If no parameters are specified, the call will return all units.

Parameter:

[Visits] (STRING): comma-separated list of IDs for the visits (see Namespace)

[Namespace] (STRING): indicates the system from which the values for patients and stays originate; Empty = both parameters are expected

[UnitIDs] (STRING): contains a comma-separated list of ID for the units

[FieldSelectors] (STRING): contains a comma-separated list of the requested fields; see Field labels

[ResultType] (STRING): indicates in which format the output parameters are provided (DEFAULT BASE64)

§ STRING = the result will be returned as string parameter

§ BASE64 = the result will be returned Base64-encoded

[Patients] (STRING): not currently implemented (contains a comma-separated list of IDs for patients (see Namespace)

Return values:

Result (STRING): returns the result of the search request as XML document

med.LoincViewSets

Description:

This job returns the master data of the ViewSets of the LOINC system.

Parameter:

[ResultType] (STRING): indicates in which format the output parameters are provided (DEFAULT BASE64)

§ STRING = the result will be returned as string parameter

§ BASE64 = the result will be returned Base64-encoded

Return values:

Result (STRING): returns the result of the search request as XML document

med.PatientData

Description:

This job returns data of one or more patients in a specified form, independent of the archive structure. So the caller has the possibility to access patient data without knowing the archive's internal structure.

The job uses data which can be found at different locations depending on the archive structure. To guarantee that the same result structure is returned every time, an external query (query_patients.xml) and an external style sheet (query_patients.xsl) will be used. These two files can be found in the server directory etc/med and have to be adapted to the archive's corresponding structure.

The search request is a DMS executor search into which the passed job parameters will be included by the PatientData job. The DMS executor only takes those areas into account for which param tags have been created in the DMS search request. The following DMS search param tags will be inserted:

§ PatientID – the value set in PatientID will be inserted

§ Surname – the value set in Surname Parameter will be inserted

§ Firstname – the value set in Firstname Parameter will be inserted

§ Visit – a tag will be inserted for every value of the list passed to Firstname Parameter

Parameter:

[PatientID] (STRING): contains the patient ID of the requested data, use of wildcards is possible for this parameter (the patient ID is a customer-specific value and has nothing to do with the internal IDs of the archive/Loinc system.)

[Surname] (STRING): contains the surname of the requested patient, use of wildcards is possible for this parameter

[Firstname] (STRING): contains the surname of the requested patient, use of wildcards is possible for this parameter

[Visits] (STRING): contains a comma-separated list of ID for the visits (see Namespace), the use of wildcards is possible for this parameter

[Namespace] (STRING): indicates the system from which the values for visits originate; Empty = both parameters are expected

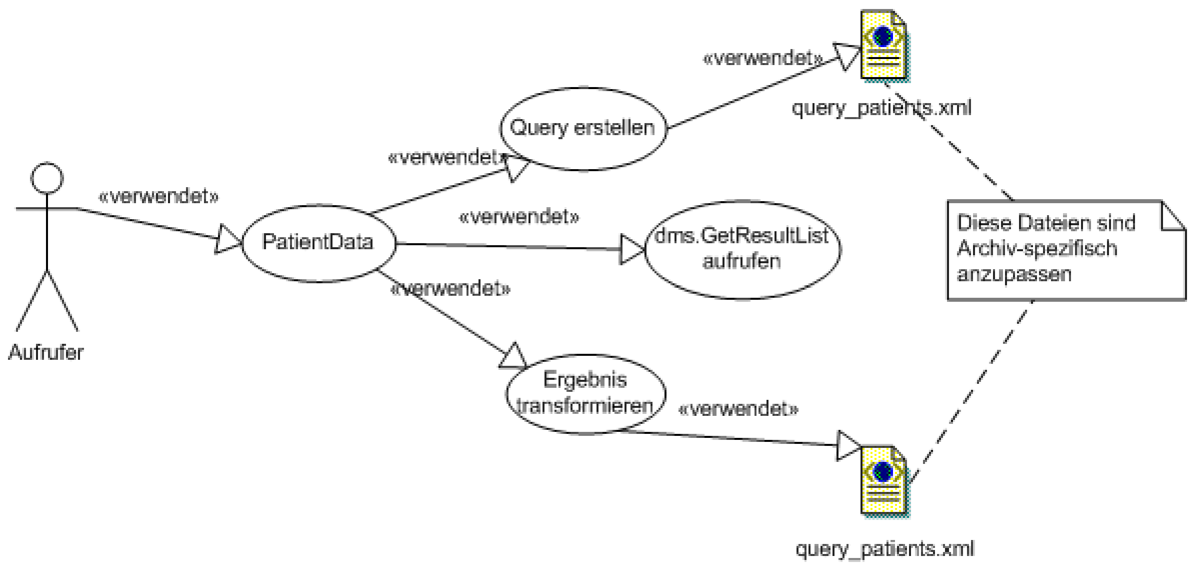
[ResultType] (STRING): indicates in which format the output parameters are provided (DEFAULT BASE64)

§ STRING = the result will be returned as string parameter

§ BASE64 = the result will be returned Base64-encoded

Return values:

Result (STRING): returns the result of the search request as XML document



Internal structure of the job:

The following examples demonstrate the structure of filed XSL documents and the accordingly created XML documents.

Filed DMS search (query_patients.xml):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DMSQuery>
<Params/>
<Archive name="PATIENT">
<ObjectType name="Patient">
<Fields>
<Field name="Patient ID"/>
<Field name="Name"/>
<Field name="First name"/>
</Fields>
<Conditions>
<ConditionObject name="patient">
<FieldCondition name="first name">
<ParamValue ref="Firstname"/>
</FieldCondition>
<FieldCondition name="Name">
<ParamValue ref="Surname"/>
</FieldCondition>
<FieldCondition name="Patient ID">
<ParamValue ref="PatientID"/>
</FieldCondition>
</ConditionObject>
<ConditionObject name="Visit">
<FieldCondition name="Case ID">
```

```
<ParamValue ref="Visit"/>
</FieldCondition>
</ConditionObject>
</Conditions>
<ChildObjects child_schema="def" export_depth="5">
<SubObjectType name="Visit">
<Fields>
<Field name="Case ID"/>
<Field name="Admission date"/>
</Fields>
</SubObjectType>
<SubObjectType name="Movement">
<Fields fields_shema="ALL"/>
</SubObjectType>
</ChildObjects>
</ObjectType>
</Archive>
</DMSQuery>
```

DMS query with inserted parameters for transfer to the DMS executor:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DMSQuery>
<Params>
<Param name="Surname">Meier</Param>
<Param name="Firstname">Er*</Param>
<Param name="Visit">28*</Param>
</Params>
<Archive name="PATIENT">
<ObjectType name="Patient">
<Fields>
<Field name="Patient ID"/>
<Field name="Name"/>
<Field name="First name"/>
</Fields>
<Conditions>
<ConditionObject name="patient">
<FieldCondition name="first name">
<ParamValue ref="Firstname"/>
</FieldCondition>
<FieldCondition name="Name">
<ParamValue ref="Surname"/>
</FieldCondition>
<FieldCondition name="Patient ID">
<ParamValue ref="PatientID"/>
</FieldCondition>
</ConditionObject>
<ConditionObject name="Visit">
<FieldCondition name="Case ID">
<ParamValue ref="Visit"/>
</FieldCondition>
</ConditionObject>
</Conditions>
<ChildObjects child_schema="def" export_depth="5">
<SubObjectType name="Visit">
<Fields>
<Field name="Case ID"/>
<Field name="Admission date"/>
</Fields>
</SubObjectType>
<SubObjectType name="Movement">
<Fields fields_shema="ALL"/>
</SubObjectType>
</ChildObjects>
```

```
</ObjectType>
</Archive>
</DMSQuery>
```

Filed style sheet (query_patients.xsl):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:output method="xml"/>
<xsl:template match="/">
<xsl:element name="PatientData">
<xsl:apply-templates
select="DMSContent/Archive/ObjectType[@name='PATIENT']/ObjectList"
mode="Patient"/>
</xsl:element>
</xsl:template>
<!--
Output patient data
-->
<xsl:template match="Object" mode="Patient">
<xsl:element name="Patient">
<xsl:attribute name="OSID">
<xsl:value-of select="@id"/>
</xsl:attribute>
<xsl:attribute name="Surname">
<xsl:value-of select="Fields/Field[@name='Name']"/>
</xsl:attribute>
<xsl:attribute name="Firstname">
<xsl:value-of select="Fields/Field[@name='Vorname']"/>
</xsl:attribute>
<xsl:attribute name="PatientID">
<xsl:value-of select="Fields/Field[@name='patient ID']"/>
</xsl:attribute>
<xsl:apply-templates
select="ChildObjects/ObjectType[@name='visit']/ObjectList"
mode="Visit"/>
</xsl:element>
</xsl:template>
<!--
Output visits
-->
<xsl:template match="Object" mode="Visit">
<xsl:element name="Visit">
<xsl:attribute name="OSID">
<xsl:value-of select="@id"/>
</xsl:attribute>
<xsl:attribute name="Case ID">
<xsl:value-of select="Fields/Field[@name='Case ID']"/>
</xsl:attribute>
<xsl:attribute name="Admission date">
<xsl:value-of select="Fields/Field[@name='admission date']"/>
</xsl:attribute>

<xsl:apply-templates
select="ChildObjects/ObjectType[@name='Movement']/ObjectList"
mode="Movement"/>
</xsl:element>
</xsl:template>
<!--
Output movements
-->
<xsl:template match="Object" mode="Movement">
```

```
<xsl:element name="Movement">
<xsl:attribute name="OSID">
<xsl:value-of select="@id"/>
</xsl:attribute>
<xsl:attribute name="Time">
<xsl:value-of select="Fields/Field[@name='Time']"/>
</xsl:attribute>
<xsl:attribute name="Movement type">
<xsl:value-of select="Fields/Field[@name='Movement type']"/>
</xsl:attribute>
<xsl:attribute name="Patient status">
<xsl:value-of select="Fields/Field[@name='Patient status']"/>
</xsl:attribute>
<xsl:attribute name="KSt">
<xsl:value-of select="Fields/Field[@name='KSt']"/>
</xsl:attribute>
<xsl:attribute name="Department">
<xsl:value-of select="Fields/Field[@name='Department']"/>
</xsl:attribute>
<xsl:attribute name="Internal movement ID">
<xsl:value-of select="Fields/Field[@name='Internal movement ID']"/>
</xsl:attribute>
<xsl:attribute name="External movement ID">
<xsl:value-of select="Fields/Field[@name='External movement ID']"/>
</xsl:attribute>
<xsl:attribute name="Origin">
<xsl:value-of select="Fields/Field[@name='Origin']"/>
</xsl:attribute>
<xsl:attribute name="Canceled">
<xsl:value-of select="Fields/Field[@name='Canceled']"/>
</xsl:attribute>
</xsl:element>
</xsl:template>
</xsl:stylesheet>
```

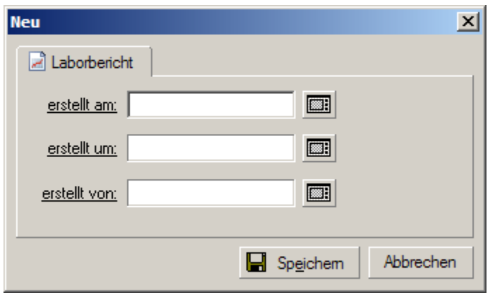
Created result:

```
<?xml version="1.0" encoding="UTF-16"?>
<PatientData>
<Patient OSID="575" Surname="Meier" Firstname="Erwin" PatientID="188123">
<Stay OSID="576" Case ID="287084" Admission date="26.08.2003">
<Movement OSID="577" Time="20030826101449" Movement type=""
Patient status="I" KSt="" Department="" Internal movement ID=""
External movement ID="" Origin="" Canceled="" />
</Visit>
</Patient>
</PatientData>
```

med.CreateLaboratoryReport

Description:

This job creates a laboratory report with the existing data for a stay. The laboratory report is created as PDF document and filed as document related to a stay. If this job is used, the following document type has to exist in the object definition:



Parameter:

[Visits] (STRING): the object ID of the visit in which the laboratory report is to be created

Return values:

[ObjectID] (INTEGER): object ID of the laboratory report added to the visit

Internal structure of the job:

The med.Observation job returns laboratory data for the stay, the med.PatientData job returns patient data. An XSL transformation will be carried out afterwards. The corresponding style sheet labreport.xslt is in the directory ./etc/med. The result of this transformation is an XSL:FO file. This file is converted into a PDF document via the job cnv.ConvertDocument. Afterwards, the PDF document will be added to the stay with the dms.XMLInsert job.

med.UpdatePatientId

Description:

The job resets the patient ID for all entries in the laboratory system with the passed patient ID.

Parameter:

[OldPatientID] (STRING):	object ID of the patient to be replaced by the new object ID
[NewPatientID] (STRING):	new object ID of the patient to replace the old object ID
[VisitID] (STRING):	object ID of the visit.

If this ID is indicated, only those PatientIDs of the data will be changed which also contain the respective StayID.

Return values:

med.UpdateVisitId

Description:

The job replaces the object ID of the stay for all entries in the laboratory system with the passed object ID of the stay.

Parameter:

[OldVisitID] (STRING):	object ID of the visit to be replaced by the new object ID of the visit
[NewVisitID] (STRING):	new object ID of the visit to replace the old object ID

Return values:

med.ObservationInsert

Description:

Adds new laboratory values to the system. With this job, it is possible to add Loinc values for patients to the laboratory system in with secure transactions. Date and time values which have been passed empty will be saved as NULL in the database.

Parameter:

[Input] (BASE64): contains the data to be added in an XML structure

The following XML codings are supported:

UTF-8 (with and without BOM)

if no BOM is present for UTF-8, it will be checked whether the encoding UTF-8 exists in the XML data At the moment, this is only done by string comparison!

iso-8859-1 The XML data are searched for the encoding iso-8859-1. If it is found, the data will be interpreted as ASCII. At the moment, this is only done by string comparison!

UFT-16-little endian (with and without BOM)

Identification is performed in the following order:

UTF-16 BOM check

UTF-8 BOM check

XML data are checked for UTF-8 encoding

XML data are checked for ISO-8859-1 encoding

all other XML data are interpreted as UTF-16

The check of the respective encoding is done context-insensitively. Formats which are not supported are interpreted as UTF 16.

[Encoding] (STRING): specifies the encoding format for the return XML. If this parameter is not indicated or if it is not listed below, UTF-16 will be returned. If the parameter ResultType is called with the value String, the parameter will be ignored and the result will be returned with UTF-16 encoding.

The parameter value is context-insensitive – the following parameter value will be supported:

UTF-8 - the data will be returned UTF-8 encoded.

ASCII - the data will be returned ASCII encoded (ISO-8859-1).

If the parameter ResultType is called with the value String, the parameter will be ignored and the result will be returned as String and not as Base64-encoded parameter. This parameter is to be used for test purposes only. If this parameter is not indicated, the data will be returned Base64-encoded.

The parameter value is context-insensitive – the following parameters will be supported:

STRING - If this value is given, the result will be returned in the output parameter <Result> in the String format.

BASE64- If this value is given, the result will be returned Base64-encoded.

Return values:

[Result] (BASE64/STRING): returns the XML enriched with the fields additionally created by the job. It additionally contains the created IDs, status information and UpdateCounts.

med.ObservationResultHistory

Description:

Job to determine the history of a result (OBX)

This job can be used to track the update history of single results.

All determined results will be returned under the current response (OBR).

They are sorted in the following order:

Patient ID	(obr_PatObjectID)
Visit ID	(obr_VisObjectID)
Request ID	(obr_ID)
Result status	(obx_State)
Result update count (reverse)	(obx_UpdateCount DESC)

Parameter:

[ResultIds] (STRING): IDs of the results (OBX) for which the update history is to be displayed. The individual results are to be displayed comma-separated. These IDs are IDs which are returned by the med.ObservationValues. They are no LOINC-IDs.

[ShowTestValues] (STRING): specifies whether the values marked as test are to be returned or those not marked as test. If this parameter is not indicated, the data which have not been marked will be returned. In order to be taken into account, the values of the search as well as the values of the results must have the same settings. If this parameter is not indicated, the data which have not been marked for test purposes will be returned.

The parameter value is context-insensitive – the following parameter value will be supported:

true	the values marked for test purposes will be returned
false	the values not marked for test purposes will be returned

[Encoding] (STRING): specifies the encoding format for the return XML. If this parameter is not indicated or if it is not listed below, UTF-16 will be returned. If the parameter ResultType is called with the value String, the parameter will be ignored and the result will be returned with UTF-16 encoding.

The parameter value is context-insensitive – the following parameter value will be supported:

UTF-8 -	the data will be returned UTF-8 encoded.
ASCII -	the data will be returned ASCII encoded (ISO-8859-1).

If the parameter ResultType is called with the value String, the parameter will be ignored and the result will be returned as String and not as Base64-encoded parameter. This parameter is to be used for test purposes only. If this parameter is not indicated, the data will be returned Base64-encoded.

The parameter value is context-insensitive – the following parameters will be supported:

STRING - If this value is given, the result will be returned in the output parameter <Result> in the String format.

BASE64- If this value is given, the result will be returned Base64-encoded.

Return values:

[Result] (BASE64/STRING): returns the history of the requested results.

med.ObservationRequestHistory

Description:

Job for determining the history of a search. This job can be used to track the update history of single requests (OBRs).

Parameter:

[ResultIds] (STRING): IDs of the OBRs for which the update history is to be displayed. The individual OBRs are to be displayed comma-separated. It is only possible to track current OBRs. No IDs will be returned for old (already updated) OBRs. Current OBRs can be determined with med.LoincValues and the indication of the relevant stay.

[ShowTestValues] (STRING): specifies whether the values marked as test are to be returned or those not marked as test. If this parameter is not indicated, the data which have not been marked will be returned. In order to be taken into account, the values of the search as well as the values of the results must have the same settings. If this parameter is not indicated, the data which have not been marked for test purposes will be returned.

The parameter value is context-insensitive – the following parameter value will be supported:

true	the values marked for test purposes will be returned
false	the values not marked for test purposes will be returned

[Encoding] (STRING): specifies the encoding format for the return XML. If this parameter is not indicated or if it is not listed below, UTF-16 will be returned. If the parameter ResultType is called with the value String, the parameter will be ignored and the result will be returned with UTF-16 encoding.

The parameter value is context-insensitive – the following parameter value will be supported:

UTF-8 -	the data will be returned UTF-8 encoded.
ASCII -	the data will be returned ASCII encoded (ISO-8859-1).

If the parameter ResultType is called with the value String, the parameter will be ignored and the result will be returned as String and not as Base64-encoded parameter. This parameter is to be used for test purposes only. If this parameter is not indicated, the data will be returned Base64-encoded.

The parameter value is context-insensitive – the following parameters will be supported:

STRING - If this value is given, the result will be returned in the output parameter <Result> in the String format.

BASE64- If this value is given, the result will be returned Base64-encoded.

Return values:

[Result] (BASE64/STRING): returns the history of the requested OBRs.

med.ObservationValues

Description:

Returns the values of the laboratory system. Values from the laboratory system can be determined with this job. The corresponding master data will be additionally returned. Unit master data will be returned according to the result. Viewsets will always be returned complete.

Parameter:

[Patients] (STRING): object IDs of the patients for which the values are to be determined. If data have to be determined for several patients, comma-separated IDs have to be passed. If an empty parameter is passed, an error will be returned. If this parameter is not indicated, no patients will be selected.

[Visits] (STRING): object IDs of the visits for which the values are to be determined. If data have to be determined for several stays, comma-separated IDs have to be passed. If an empty parameter is passed, an error will be returned. If this parameter is not indicated, no stays will be selected.

[State] (STRING): limits the query by status

The parameter value is context-insensitive – the following parameters will be supported:

ALL - Current as well as updated values will be output. This corresponds to specification <1.2> but performs better.

1 Only the current values will be output. This is like calling the job without the parameter State.

With the value 2, no final result will returned as the relations are changed during an update!

[ShowTestValues] (STRING): specifies whether the values marked as test are to be returned or those not marked as test. If this parameter is not indicated, the data which have not been marked will be returned. In order to be taken into account, the values of the search as well as the values of the results must have the same settings. If this parameter is not indicated, the data which have not been marked for test purposes will be returned.

The parameter value is context-insensitive – the following parameter value will be supported:

true the values marked for test purposes will be returned

false the values not marked for test purposes will be returned

[Optimize] (STRING): enables optimization of the request or return of the result.

The parameter value is context-insensitive – the following parameter value will be supported:

NoEmptyFields

if this value is given, all fields containing an empty string will be suppressed in the output. This leads to smaller sizes for output documents and reduced network load. If this parameter is not indicated, the corresponding fields will be returned with an empty string.

[Encoding] (STRING): specifies the encoding format for the return XML. If this parameter is not indicated or if it is not listed below, UTF-16 will be returned. If the parameter ResultType is called with the value String, the parameter will be ignored and the result will be returned with UTF-16 encoding.

The parameter value is context-insensitive – the following parameter value will be supported:

UTF-8 - the data will be returned UTF-8 encoded.

ASCII - the data will be returned ASCII encoded (ISO-8859-1).

If the parameter ResultType is called with the value String, the parameter will be ignored and the result will be returned as String and not as Base64-encoded parameter. This parameter is to be used for test purposes only. If this parameter is not indicated, the data will be returned Base64-encoded.

The parameter value is context-insensitive – the following parameters will be supported:

STRING - If this value is given, the result will be returned in the output parameter <Result> in the String format.

BASE64- If this value is given, the result will be returned Base64-encoded.

Return values:

[Result] (BASE64/STRING): returns the requested laboratory values as an XML document

med.SaveMedicalRecord

Description:

With this job, medical documents can be saved in the archive. It is provided by Optimal Systems GmbH.

Documents can only be saved when the status code of the medical record has the values 'new', 'active' or 'completed'. With the status codes 'obsolete' or 'canceled', the job will return an exception. The caller will be given an error message which can be programmatically received.

The storage code for documents which will be saved or queried has to meet the consistency requirements for medical records according to the HL V2.x Specification (Chapter 9, Medical Records/Information Management (Document Management)).

Attachments and documents which do not comply with the CDA standard have to be provided in a file list. The first document of the list is the Medical Record Document and the other documents are attachments which have to be saved in the system together with the Medical Record. The references to the attachments can be used with the MedicalRecordAttachments parameter.

The MedicalRecord.id.root or MedicalRecord.id.extension parameters are optional. If they are not indicated, the system will generate these IDs.

Parameter:

If no ClincialDocument parameter is passed

[Patientnumber] (STRING): patient number of the patient whose medical record is requested.

[Visitnumber] (STRING): case number of the patient's visit for which the medical record is requested.

[MedicalRecordContainer] (STRING): the document in the archive that is to be used for saving the medical record.

[MedicalRecordAttachmentsContainer] (STRING): the document in the archive that is to be used for saving medical record attachments.

[MedicalRecord.statusCode] (STRING): status code of the medical record. Available values are:

new

active

completed

obsolete

canceled

[MedicalRecord.completionCode] (STRING): completion code of the medical record. Available values are:

AU - authenticated

DI - dictated

DO - documented

IP - in progress

IN - incomplete

LA - legally authenticated

PA pre-authenticated

[MedicalRecord.availibilityTime] (STRING): availability time of the document. This value is returned as HL7 time stamp.

[MedicalRecord.storageCode] (STRING): storage code of the medical record. Available values are:

AC - active

AA - active and archived

AR - archived (not active)

PU - purged

Either

[ClinicalDocument] (BASE64): CDA document to be saved as a medical record.

Or

[MedicalRecord.id.root] (STRING): ROOT ID of the medical record.

[MedicalRecord.id.extension] (STRING): extension ID of the medical records.

Optional

If nothing is indicated, it is assumed that no relations exist to a parent object.

[MedicalRecord.relatedDocument.parentDocument.id.root] (STRING): root ID of the related document.

[MedicalRecord.relatedDocument.parentDocument.id.extension] (STRING): extension ID of the related document.

[`MedicalRecord.relatedDocument.typeCode`] (STRING): relation to the parent object. Available values are:

RPLC – will replace the related document A new main variant is created.

XFRM – the returned document is a transformation of the related document. E.g. a digitally signed version. A new subvariant iwill be created.

APND – the returned document is an attachment/additional document to the related document. A new main variant 1.0 will be created.

Return values:

[ObjectID] (STRING): object ID under which the document was saved in the archive. This ID is created by the system and cannot be created by the caller.

[MedicalRecord.id.root] (STRING): root of the ID of the document to be requested.

[MedicalRecord.id.extension] (STRING): extension of the ID of the document to be requested.

[MedicalRecord.statusCode] (STRING): status code of the medical record. Available values are:

new

active

completed

obsolete

canceled

[MedicalRecord.completionCode] (STRING): completion code of the medical record. Available values are:

AU authenticated

DI dictated

DO documented

IP in progress

IN incomplete

LA legally authenticated

PA pre-authenticated

[MedicalRecord.availibilityTime] (STRING): availability time of the document. This value is returned as HL7 time stamp.

[MedicalRecord.storageCode] (STRING): storage code of the medical record. Available values are:

AC active

AA active and archived

AR archived (not active)

PU purged

med.GetMedicalRecord

Description:

A medical record which was saved in the archive with the med.SaveMedicalRecord job can be queried with this job.

Two different modes can be used for queries.

Combination of the OID (equals root ID) and the instance ID (equals extension ID).

Object ID from the archive

Afterwards additonal parameters will be listed which can be/have to be indicated when a job is called.

Documents can only be returned when the status code of the medical record has the values 'new', 'active' or 'completed'. With the statuses 'obsolete' or 'cancelled', the caller will only be provided the Status code of the medical record.

The storage code for documents which will be saved or queried has to meet the consistency requirements for medical records according to the HL V2.x Specification (Chapter 9, Medical Records/Information Management (Document Management)).

The documents will be provided in a file list. The first document of the list is the Medical Record Document, other documents are attachments which were saved in the system together with the Medical Record. References to the attachments can be used with the MedicalRecordAttachments output parameter.

Parameter:

[Patientnumber] (STRING): patient number of the patient whose medical record is requested.

[Visitnumber] (STRING): case number of the patient's visit for which the medical record is requested.

[MedicalRecordContainer] (STRING): the document in the archive that is to be used for saving the medical record.

[MedicalRecordAttachmentsContainer] (STRING): the document in the archive that is to be used for saving medical record attachments.

Either

[MedicalRecord.id.root] (STRING): root of the ID of the document to be requested.

[MedicalRecord.id.extension] (STRING): extension of the ID of the document to be requested.

or

[ObjectID] (STRING): object ID from the archive of the document to be requested.

Return values:

[DMSContent] (STRING): DMSContent that was generated by this request.

[MedicalRecord.statusCode] (STRING): status code of the medical record. Available values are:

new

active

completed

obsolete

canceled

[MedicalRecord.completionCode] (STRING): completion code of the medical record. Available values are:

AU authenticated

DI dictated

DO documented

IP in progress

IN incomplete

LA legally authenticated

PA pre-authenticated

[MedicalRecord.availibilityTime] (STRING): availability time of the document. This value is returned as HL7 time stamp.

[MedicalRecord.storageCode] (STRING): storage code of the medical record. Available values are:

AC active

AA active and archived

AR archived (not active)

PU purged

[MedicalRecordAttachments] (STRING): list of names/links of the attachments as a comma-separated list.

[FileCount] (INT): number of returned documents/files, including the medical record.

med.NotifyMedicalRecord

Description:

The document status and transitions can be edited with this job. For further information refer to HL7 V3 Specifications.

Parameter:

[Patientnumber] (STRING): patient number of the patient whose medical record is requested.

[Visitnumber] (STRING): case number of the patient's visit for which the medical record is requested.

[MedicalRecordContainer] (STRING): the document in the archive that is to be used for saving the medical record.

[MedicalRecordAttachmentsContainer] (STRING): the document in the archive that is to be used for saving medical record attachments.

[MedicalRecord.id.root] (STRING): root of the ID of the document to be requested.

[MedicalRecord.id.extension] (STRING): extension of the ID of the document to be requested.

[MedicalRecord.statusCode] (STRING): status code of the medical record. Available values are:

new

active

completed

obsolete

canceled

[MedicalRecord.completionCode] (STRING): completion code of the medical record. Available values are:

AU authenticated

DI dictated

DO documented

IP in progress

IN incomplete

LA legally authenticated

PA pre-authenticated

[MedicalRecord.availibilityTime] (STRING): availability time of the document. This value is returned as HL7 time stamp.

[MedicalRecord.storageCode] (STRING): storage code of the medical record. Available values are:

AC active

AA active and archived

AR archived (not active)

PU purged

Return values:

[MedicalRecord.id.root] (STRING): root of the ID of the document to be requested.

[MedicalRecord.id.extension] (STRING): extension of the ID of the document to be requested.

[MedicalRecord.statusCode] (STRING): status code of the medical record. Available values are:

new

active

completed

obsolete

canceled

[MedicalRecord.completionCode] (STRING): completion code of the medical record. Available values are:

AU authenticated

DI dictated

DO documented

IP in progress

IN incomplete

LA legally authenticated

PA pre-authenticated

[MedicalRecord.availibilityTime] (STRING): availability time of the document. This value is returned as HL7 time stamp.

[MedicalRecord.storageCode] (STRING): storage code of the medical record. Available values are:

AC active

AA active and archived

AR archived (not active)

PU purged

med.GetSystemOID

Description:

The system's unique OID can be determined with this job.

ASN.1 Notation: {iso(1) member-body(2) de(276) din-certco(0) gesundheitswesen(76) instanzen-identifikatoren(3) organisationen(1) optimal-systems(11)}

dot notation: 1.2.276.0.76.3.1.11

URN-notation: urn:oid:1.2.276.0.76.3.1.11

Return values:

[OID] (STRING):	returns the OID for the system

MNG Engine (Namespace mng)

This engine provides jobs for the administration of groups and users in enaio®.

§ mng.AddUserGroupAsc

§ mng.CreateGroup

§ mng.CreateUser

§ mng.DeleteGroup

§ mng.DeleteUser

§ mng.EmptyGroup

§ mng.GetGroupAttributes

§ mng.GetGroupList

§ mng.GetGroupMembers

§ mng.GetUserAttributes

§ mng.GetUserGroups

§ mng.GetUserProfile

§ mng.GetUserList

§ mng.RemoveUserGroupAsc

§ mng.SetGroupAttributes

§ mng.SetUserAttributes

§ mng.StoreUserProfile

The following jobs can only be executed by users with administrative rights:

§ mng.AddUserGroupAsc

§ mng.CreateGroup

§ mng.CreateUser

§ mng.DeleteGroup

§ mng.DeleteUser

§ mng.EmptyGroup

§ mng.RemoveUserGroupAsc

§ mng.SetGroupAttributes

§ mng.SetUserAttributes

mng.AddUserGroupAsc

Description:

This job adds the specified users to a group. The user and the group can be specified either via the GUID or ID.

Parameter:

Flags (INT4): not currently used

AdmInfo (BASE64): group assignment in XML format

Example:

Structure of AdmInfo

```
<AdmInfo>
<Associations>
<Association osuid="" osgid=""/>
<Association osuid="" osgid=""/>
<! - -OR- - >
<Association user_id="" group_id=""/>
<Association user_id="" group_id=""/>
</Associations>
</AdmInfo>
```

Note:

Detailed description of AdmInfo

§ [osuid] (STRING): GUID of the user

§ [osgid] (STRING): Group GUID

§ [user_id] (STRING): User ID

§ [group_id] (INT): Group ID

See also:

mng.RemoveUserGroupAsc

mng.CreateGroup

Description:

This job creates a new user group. An entry is created in the database table 'gruppen'. The ID and the Osguid are generated by the job and returned as XML.

Parameter:

Flags (INT4): not currently used

GroupInfo (BASE64): group properties in XML format

HasEncoding (boolean): GroupInfo contains ncoding (e.g. UTF-8)

Return values:

GroupInfo (BASE64): group properties in XML format (id and osguid are set)

Example:

Structure of GroupInfo

```
<AdmInfo>
<Groups>
<Group name="Test" profile="0" description=""/>
</Groups>
</AdmInfo>
```

Note:

Detailed description of GroupInfo

§ id (INT): Group ID

§ name (STRING): Group name

- osguid (STRING): Group GUID
- profile (LONG): ID of the profile user who is assigned to the group
- description (STRING):description for the group

mng.CreateUser

Description:

This job creates a new user. A new data record is created in the database table 'benutzer'. The ID and the Osguid are generated by the job and returned as XML.

Parameter:

Flags (INT4): not currently used

UserInfo (BASE64): user properties in XML format

HasEncoding (boolean): UserInfo contains ncoding (e.g. UTF-8)

Return values:

UserInfo (BASE64): user properties in XML format (id and osguid are set)

Example:

Structure of UserInfo

```
<AdmInfo>
<Users>
<User account_type="0" comment="" user="TESTUSER" flags="1"
changed="1" langid="0" locked="0" logincount="0"
loginstation="" logintime="0" name="Peter Muster"
osemail=""
password="B62441422712357307" profile="-1" server_id="3"
station="" supervisor="0" validfrom="" validto="">
</User>
</Users>
</AdmInfo>
```

Note:

Detailed description of XmlInfo

User: contains information on a user

- account_type (INT): account type
    - NULL/0 = user login
    - 1 = Login of the application server
    - 2 = Login of ANONYMOUS
    - 3 = Login of the application server (e.g. Java server)
- comment (STRING): Field for comments (e.g. phone number)
- user (STRING): User name
- flags (INT): 0 = normal user, 1 = server or ANONYMOUS
- changed (INT): 0 = profile was not changed; 1 = the profile was changed
- id (INT): User ID
- langid (INT): language ID (empty = German)

- locked (INT): 1 = user is locked, otherwise 0
- logincount (INT): number of login attempts
- loginstation (STRING): name of the last login station
- logintime (INT): time of the login (timestamp)
- name (STRING): full name of the user
- osemail (STRING): user e-mail
- osguid (STRING): GUID of the user
- password (STRING): encoded user password
- profile (INT): -1 = user has no profile; 0 = user profile; >0 = ID of the assigned user profile
- server_id (INT): ID of the server
- station (STRING): name of the user workstation
- supervisor (INT): -1 = supervisor, otherwise 0
- validfrom (INT): user account valid from (timestamp)
- validto (INT): user account valid until (timestamp)

mng.DeleteGroup

Description:

This job deletes a group from the database table 'gruppen'. A group can only be deleted if it has no members (database table 'bgrel').

Parameter:

Flags (INT4): indicates the parameter by which the group is to be identified

- 0 - Parameter GroupGuid
- 1 - Parameter GroupId
- 2 - Parameter GroupName

[GroupGuid] (BASE64): Group GUID

[GroupId] (BASE64): Group ID

[GroupName] (BASE64): Group name

See also:

mng.EmptyGroup

mng.DeleteUser

Description:

This job deletes a user from the database table 'benutzer'. Group memberships of the user (bgrel), system roles (ossysroles), subscriptions (osabonnement) and personal settings (osconf) will be deleted. Furthermore it is possible to forward portfolios and the contents of the inbox to another user resp. to entirely delete these data.

Parameter:

Flags (INT4): indicates by which parameter the user is to be identified

§ 0 - Parameter User/where applicable Target

§ 1 - Parameter UserGuid/where applicable TargetGuid

§ 2 - Parameter UserId/where applicable TargetId

InheritanceFlags (INT4): indicates whether portfolios and the contents of the inbox are to be forwarded to another user

§ 0 – portfolios and e-mails will be deleted

§ 1 – portfolios will be forwarded

§ 2 – e-mails will be forwarded

§ 3 - portfolios and e-mails will be forwarded

[User] (BASE64): User name

[UserGuid] (BASE64): User ID

[UserId] (BASE64): User name

[Target] (BASE64): user name (receives portfolios/emails)

[TargetGuid] (BASE64): user ID (receives portfolios/emails)

[TargetId] (BASE64): user name (receives portfolios/emails)

mng.EmptyGroup

Description:

This job empties a group. All user assignments will be deleted (database table 'bgrel').

Parameter:

Flags (INT4): indicates the parameter by which the group is to be identified

§ 0 - Parameter GroupGuid

§ 1 - Parameter GroupId

§ 2 - Parameter GroupName

[GroupGuid] (BASE64): Group GUID

[GroupId] (BASE64): Group ID

[GroupName] (BASE64): Group name

mng.GetGroupAttributes

Description:

This job returns the properties of the specified group.

Parameter:

Flags (INT4): not currently used

Group (STRING): Group name

Return values:

XmlInfo (STRING): group properties in XML format

Example:

Structure of XmlInfo

```
<AdmInfo>
<Groups>
<Group id="0" name="STANDARD" osguid="C9BBC4B0D7754065B3EA6232D7B70003"
profil="0" description="">
</Group>
</Groups>
</AdmInfo>
```

Note:

Detailed description of XmlInfo

- id (INT): Group ID
- name (STRING): Group name
- osguid (STRING): Group GUID
- profile (LONG): ID of the profile user who is assigned to the group
- description (STRING):description for the group

See also:

mng.GetGroupList, mng.SetGroupAttributes

mng.GetGroupList

Description:

This job returns a list of all groups.

Parameter:

Flags (INT4): not currently used

Return values:

GroupList (STRING): list of all defined groups in XML format

Example:

Structure of GroupList

```
<AdmInfo>
<Groups>
<Group id="0" name="STANDARD" osguid="C9BBC4B0D7754065B3EA6232D7B70003"
profile="0" description=""></Group>
<Group id="157" name="TEST" osguid="B36506740D764731836365D04333D3AD"
profile="79" description=""></Group>
<Group id="18" name="ALL EMPLOYEES" osguid="65A56409BB3FFFC687FCC9B90"
profile="0" description=""></Group>
</Groups>
</AdmInfo>
```

Note:

Detailed description of GroupList

- id (INT): Group ID
- name (STRING): Group name
- osguid (STRING): Group GUID
- profile (LONG): ID of the profile user who is assigned to the group

§ description (STRING):description for the group

See also:

mng.GetGroupAttributes, mng.GetGroupMembers

mng.GetGroupMembers

Description:

This job returns a list of all members of the specified group.

Parameter:

Flags (INT4): indicates which parameter is to be used for the search

§ 0 = search using the parameter GroupName

§ 1 = search using the parameter GroupGUID

§ 2 = search using the parameter GroupID

[GroupName] (STRING): Group name

[GroupGUID] (STRING): Group GUID

[GroupID] (INT4): Group ID

Return values:

UserList (STRING): list of all group members

Example:

Structure of UserList

```
<AdmInfo>
<Users>
<User user="ROOT" id="2" name=""
osguid="C97ABFC32E09431192E4B13CF47293D6"></User>
<User user="Testuser" id="49" name="Peter Muster"
osguid="6759985B74A44747ACC93F031913006C"></User>
</Users>
</AdmInfo>
```

Note:

Detailed description of UserList

Users: list of all group members

§ user (STRING): User name

§ id (INT): User ID

§ name (STRING): full name of the user

§ osguid (STRING): GUID of the user

mng.GetUserAttributes

Description:

This job returns the properties of the specified user.

Parameter:

Flags (INT4): not currently used

User (STRING): user name from the DB field 'user.user'

Return values:

XmlInfo (STRING): user information in XML format

Example:

Structure of XmlInfo

```
<AdmInfo>
<Users>
<User account_type="0" comment="" user="TESTUSER" flags="1"
changed="1" id="70" langid="0" locked="0" logincount="0"
loginstation="" logintime="0" name="Peter Muster"
osemail="" osguid="A95EA1EEA16A4EDF916400F6E2F5BCF9"
password="B62441422712357307" profile="-1" server_id="3"
station="" supervisor="0" validfrom="" validto="">
</User>
</Users>
</AdmInfo>
```

Note:

Detailed description of XmlInfo

User: contains information on a user

§ account_type (INT): account type
  - § NULL/0 = user login
  - § 1 = Login of the application server
  - § 2 = Login of ANONYMOUS
  - § 3 = Login of the application server (e.g. Java server)

§ comment (STRING): Field for comments (e.g. phone number)

§ user (STRING): User name

§ flags (INT): 0 = normal user, 1 = server or ANONYMOUS

§ changed (INT): 0 = profile was not changed; 1 = the profile was changed

§ id (INT): User ID

§ langid (INT): ID of the object definition language (empty = German)

§ locked (INT): 1 = user is locked, otherwise 0

§ logincount (INT): number of login attempts

§ loginstation (STRING): name of the last login station

§ logintime (INT): time of the login (timestamp)

§ name (STRING): full name of the user

§ osemail (STRING): user e-mail

§ osguid (STRING): GUID of the user

§ profile (INT): -1 = user has no profile; 0 = user profile; >0 = ID of the assigned user profile

§ server_id (INT): server ID

§ station (STRING): name of the user workstation

§ supervisor (INT): -1 = supervisor, otherwise 0

- validfrom (INT): user account valid from (timestamp)
- validto (INT): user account valid until (timestamp)

See also:

mng.SetUserAttributes

mng.GetUserGroups

Description:

This job returns a list of all groups to which the specified user belongs.

Parameter:

Flags (INT4): not currently used

UserGUID (STRING): GUID of the user

Return values:

GroupList (STRING): list of all groups in which the user is located.

Example:

Structure of GroupList

```
<AdmInfo>
<Groups>
<Group id="0" name="STANDARD" osguid="C9BBC4B0D7754065B3EA6232D7B70003"
profile="0" description=""></Group>
<Group id="157" name="TEST" osguid="B36506740D764731836365D04333D3AD"
profile="79" description=""></Group>
</Groups>
</AdmInfo>
```

Note:

Detailed description of GroupList

Groups: list of all groups in which the user is located

- id (INT): Group ID
- name (STRING): Group name
- osguid (STRING): Group GUID
- profile (LONG): ID of the profile user who is assigned to the group
- description (STRING):description for the group

mng.GetUserList

Description:

This job returns a list of all users.

Parameter:

Flags (INT4): not currently used

Return values:

UserList (STRING): list of all users

Example:

Structure of UserList

```
<AdmInfo>
<Users>
<User user="ROOT" id="2" name=""
osguid="C97ABFC32E09431192E4B13CF47293D6"></User>
<User user="Testuser" id="49" name="Peter Muster"
osguid="6759985B74A44747ACC93F031913006C"></User>
</Users>
</AdmInfo>
```

Note:

Detailed description of UserList

Users: list of all users

§ user (STRING): User name

§ id (INT): User ID

§ name (STRING): full name of the user

§ osguid (STRING): GUID of the user

See also:

mng.GetUserAttributes

mng.GetUserProfile

Description:

This job passes the client the profile of a user.

Parameter:

Flags (INT):

§ HIWORD(Flags) = 1: LowDateTime and HighDateTime are returned

§ HIWORD(Flags) = 2: LowDateTime is returned

UserProfile (STRING): UserProfile

Return values:

FileCount (INT): FileCount equals 1

[LowDateTime] (INT): User profile date stamp in LowDateTime format

[HighDateTime] (INT): User profile date stamp HighDateTime

File list: file name with complete path

Return:

(INT): 0 = job successful, otherwise error code

See also:

mng.StoreUserProfile

mng.RemoveUserGroupAsc

Description:

This job deletes an assignment of a user to a group (database table bgrel).

Parameter:

Flags (INT4): indicates which parameter is to be used

§ 0 - AdmInfo

§ 1 – UserGuid

[AdmInfo] (BASE64): Group assignments that are to be deleted

[UserGUID] (STRING): user GUID (user is deleted from all groups where he is located)

Example:

Structure of AdmInfo

```
<AdmInfo>
<Associations>
<Association user_id="" group_id=""/>
<! - -OR- - >
<Association osuid="" osgid=""/>
<Associations>
</AdmInfo>
```

Note:

Detailed description of AdmInfo

§ [osuid] (STRING): GUID of the user

§ [osgid] (STRING): Group GUID

§ [user_id] (STRING): User ID

§ [group_id] (INT): Group ID

mng.SetGroupAttributes

Description:

This job sets the properties of a group.

Parameter:

Flags (INT4): currently not used

GroupInfo (BASE64): group properties in XML format

HasEncoding (boolean): GroupInfo contains ncoding (e.g. UTF-8)

Example:

Structure of GroupInfo

```
<AdmInfo>
<Groups>
<Group id="0" name="STANDARD" osguid="AE38D1BB1F1C4CB98B5695A2935E0169"
profile="0" description="Test"></Group>
</Groups>
</AdmInfo>
```

Note:

Detailed description of GroupInfo

§ id (INT): Group ID

- name (STRING): Group name
- osguid (STRING): Group GUID
- profile (LONG): ID of the profile user who is assigned to the group
- description (STRING):description for the group

See also:

mng.GetGroupAttributes

mng.SetUserAttributes

Description:

This job sets the properties of a user.

Parameter:

Flags (INT4):

UserInfo (BASE64): properties in XML format

HasEncoding (boolean): UserInfo contains ncoding (e.g. UTF-8)

Example:

Structure of UserInfo

```
<AdmInfo>
<Users>
<User account_type="0" comment="" user="Test" flags="1"
changed="1" id="67" langid="0" locked="0" logincount="0"
loginstation="" logintime="0" name="Peter Muster" osemail=""
osguid="EF989801BA8847199335DD4FDEF30BC5"
password="BF754341546553351243620206006521266514574240603407"
profile="66" server_id="3" station="" supervisor="0" validfrom=""
validto=""/>
</Users>
</AdmInfo>
```

Note:

Detailed description of XmlInfo

User: contains information on a user

- account_type (INT): account type
    - NULL/0 = user login
    - 1 = Login of the application server
    - 2 = Login of ANONYMOUS
    - 3 = Login of the application server (e.g. Java server)
- comment (STRING): Field for comments (e.g. phone number)
- user (STRING): User name
- flags (INT): 0 = normal user, 1 = server or ANONYMOUS
- changed (INT): 0 = profile was not changed; 1 = the profile was changed
- id (INT): User ID
- langid (INT): language ID (empty = German)

§ locked (INT): 1 = user is locked, otherwise 0

§ logincount (INT): number of login attempts

§ loginstation (STRING): name of the last login station

§ logintime (INT): time of the login (timestamp)

§ name (STRING): full name of the user

§ osemail (STRING): user e-mail

§ osguid (STRING): GUID of the user

§ password (STRING): encoded user password

§ profile (INT): -1 = user has no profile; 0 = user profile; >0 = ID of the assigned user profile

§ server_id (INT): server ID

§ station (STRING): name of the user workstation

§ supervisor (INT): -1 = supervisor, otherwise 0

§ validfrom (INT): user account valid from (timestamp)

§ validto (INT): user account valid until (timestamp)

See also:

mng.GetUserList, mng.GetUserAttributes

mng.StoreUserProfile

Description:

This job saves the user profile received by the client and writes a history file (same name, file extension bac). The passed profile file is deleted on the client.

Parameter:

Flags (INT):

§ HIWORD(Flags) = 2: save date stamp in LowDateTime format; otherwise use LowDateTime format and HighDateTime format

UserProfile (STRING): Name under which the file is to be saved

LowDateTime (INT): date stamp in LowDateTime format

HighDateTime (INT): date stamp in HighDateTime format

File list: name and path of the profile file

Return:

(INT): 0 = job successful, otherwise error code

See also:

mng.GetUserProfile

OCR Engine (Namespace ocr)

This engine provides jobs for text recognition.

§ ocr.DoOCR

§ ocr.DoDocOCR

ocr.DoOCR

Description:

This job performs text recognition for the passed file and returns the recognized text as a text file.

Parameter:

Flags (INT4): not currently used

Type (INT4): 1 = texts in multiple columns (e.g. newspaper articles) are to be recognized, otherwise 0

File list: name and path of the file for which text recognition is to be performed

Return values:

File list: name and path of the text file that contains the recognized text

ocr.DoDocOCR

Description:

This job performs text recognition for the specified enaio® document and returns the recognized text as a text file.

Parameter:

Flags (INT4): not currently used

Type (INT4): 1 = texts in multiple columns (e.g. newspaper articles) are to be recognized, otherwise 0

DocID (INT4): ID of the document

Return values:

File list: name and path of the text file that contains the recognized text

Standard Engine (Namespace std)

Functions for file-oriented document management are implemented in the standard engine. In particular functions for saving and loading documents, archiving and realizing document exchange between multiple servers.

The Standard engine takes care of the management of the application server's WORK, CACHE and archive area.

§ Work, cache, and archive management

§ File administration

§ Internal jobs

§ Other jobs

Work, Cache and Archive Management

§ std.CleanUpCache

§ std.ClearFromCache

§ std.DoArchive

§ std.DoPrefetch

§ std.MoveToCache

§ std.StoreInCache

§ std.StoreInCacheByID

§ std.StoreInCacheDirect

§ std.StoreInWork

§ std.UndoArchive

std.CleanUpCache

Description:

This job clears the contents of CACHE.

Parameter:

[Flags (INT): priority (default value 2)

§ 0 = files will be deleted until the minimum cache size is reached

§ 1 = files will be deleted which have exceeded the maximum limit for days inside the cache

§ 2 = combination of priority 0 and 1: Attempts to delete files will be made until the minimum cache size has been reached, however, only files that have been in the CACHE for more than the maximum number of days will be deleted. It is possible that the maximum cache size is not exceeded.

§ 8 = files will be deleted until the minimum document number is reached (see parameter Low)

High (INT): maximum cache size in MB, if this value is exceeded, the CACHE will be cleared

Low (INT): minimum cache size in MB; if flags = 8, the minimum document number (in thousands) will be indicated here

Days (INT): maximum number of days documents remain in the cache [ExtendDiagnostic] (INT): Logging options

- 0 = in this case, only files are stated in the report when they cannot be deleted
- 1 = all deleted files are logged (default=0)

Return:

(INT): 0 = job successful, otherwise error code

std.ClearFromCache

Description:

This job deletes the specified object from the cache.

Parameter:

dwObjectID (INT): ID of the object

dwObjectType (INT): Object type

Return:

(INT): 0 = job successful, otherwise error code

std.DoArchive

Description:

This job archives all objects of a given type set as 'archivable'. This job will only work if archiving was set up correctly first (media, sets,...).

Parameter:

dwObjectType (INT): type of objects to be archived

Return:

(INT): 0 = job successful, otherwise error code

std.DoPrefetch

Description:

This job loads the specified object from a storage medium (e.g. jukebox) into the cache to reduce access times.

Parameter:

Flags (INT): transfer options

- 0 = slide file and objects are transmitted
- 1 = only the object files are transmitted
- 2 = only the slide file of the object is transmitted

dwObjectID (INT): ID of the object

dwObjectType (INT): Object type

DocState (INT): contains the state of the document in the LOWORD and the Read-Write flag in the HIWORD (the job processes archived documents, only)

FileCount (INT): number of files, but which have no role to play here

Return:

(INT): 0 = job successful, otherwise error code

std.GetDocumentSlide

Description:

This job creates a slide for a document and returns it.

Parameter:

ObjectID (INT): ID of the object

Flags (INT):

`GETDOCSLIDE_FLAG_NEEDEDONLY(0): the requested size is returned, and if not available then an error`

GETDOCSLIDE_FLAG_DEFAULTALLOWED(1): the requested size is returned and if not available the 96x96, to know what is returned.

GETDOCSLIDE_FLAG_BOTH(2): if available, the queried size will be returned and 96x96

GETDOCSLIDE_FLAG_ALL(3): all slides

Several files will be returned for GETDOCSLIDE_FLAG_ALL and GETDOCSLIDE_FLAG_BOTH.

They have extensions such as DIA001, DIA002 etc.

There are also output parameters like DIA001, DIA002 etc. which determine the size of each slide:

DIA001 = 10x10

DIA002 = 96x96 etc.

Width and height can be 0 for GETDOCSLIDE_FLAG_NEEDEDONLY and GETDOCSLIDE_FLAG_DEFAULTALLOWED

default values will be used in this case:

if (dwWidth == 0) dwWidth = DEFAULT_SLIDE_WIDTH;

if (dwHeight == 0) dwHeight = DEFAULT_SLIDE_HEIGHT;

Height (INT): height of the slide; if 0, 96 will be used

Width (INT): width of the slide; if 0, 96 will be used

Return:

(INT): 0 = job successful, otherwise error code

std.MoveToCache

Description:

This job transfers the specified object from one server group to another. The client can only open this object as read-only object.

Parameter:

dwObjectID (INT): ID of the object

dwObjectType (INT): Object type

Return:

(INT): 0 = job successful, otherwise error code

std.StoreInCache

Description:

This job sends the specified document files to an enaio® client. If the enaio® client still has the document in its cache, the digest value can be calculated and passed to the job. The server also calculates the digest value of the specified document. If both digest values are identical, the server does not send the document to the enaio® client.

Parameter:

Flags (INT): transfer options

§ 0 = slide file and objects are transmitted

§ 1 = only the object's pages are transmitted

§ 2 = only the slide file of the object is transmitted

dwObjectID (INT): ID of the object

dwObjectType (INT): type of the selected document

DocState (INT): indicates whether the document is to be opened for reading or writing

§ HIWORD = 0 -> the document is opened for editing

§ HIWORD = 1 -> the document is opened for viewing

FileCount (INT): number of files, but which have no role to play here

[Digest] (STRING): digest value calculated by client application for the requested document.

[IncludeDeleted] (BOOLEAN): in the case of true, documents in the trash can are also taken into account.

[IgnoreHashCheck] (BOOLEAN): in the case of true, the hash value/signature check is turned off, although it may possibly be switched on in the registry.

[AddAnnotations] (BOOLEAN): if available and true, annotations are then also burned into the image files.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

FileCount (INT): number of files transferred to the cache

std.StoreInCacheByID

Description:

This job sends the specified document files to an enaio® client.

Parameter:

Flags (INT): transfer options

§ 0 = slide file and objects are transmitted

§ 1 = only the object files are transmitted

§ 2 = only the slide file of the object is transmitted

dwObjectID (INT): ID of the object

[Convert] (INT): 0 = no conversion
1 = documents with main type 1, 2, 3 or 4 are converted to PDF.
8 = TIFF documents with main type 2 or 3 are combined into a multipage TIFF

[WhenCOLDThenTIFF] (INT): 1 = ASCII cold files are returned in TIFF format (only observed if Convert= 0 is set)

[AddAnnotations] (BOOLEAN): if available and true, annotations are then burned into the image files.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

FileCount (INT): number of files transferred to the cache

File list: path and name of files that were transferred to the cache

std.StoreInCacheDirect

Description:

This job returns a specified document via its ID. The job can be used if the enaio® server and enaio® client run on the same computer.

Parameter:

Flags (INT): transfer options

§ 0 = slide file and objects are transmitted

§ 1 = only the object files are transmitted

§ 2 = only the slide file of the object is transmitted

dwObjectID (INT): ID of the object

Path (STRING): path to which the object files are written

[AddAnnotations] (BOOLEAN): if available and true, annotations are then also burned into the image files.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

FileCount (INT): number of files transferred to the cache.

File_N (STRING): name of the nth source file

std.StoreInWork

Description:

This job copies all specified documents into the specified WORK directory (it is based on the object type and object ID). If it already contains files with the specified ObjectID, these files are deleted first.

Parameter:

Flags (INT):

§ 0 = Flags: the files will be deleted

§ 1 & Flags: the files will not be deleted (use for 'StoreInWorkDirect')

§ 2 & Flags: no HardLinks are created for the files (use for variant administration)

dwObjectID (INT): ID of the object

dwObjectType (INT): Object type

FileCount (INT): Number of files

File list: name and path of files to be written to the work directory.

bAddFiles (INT): if 1, new files do not replace existing files but will be added.

bAddFront (INT): if 1, new files will be added in the front.

DocFlagsNeeded (INT): if available, the column 'Flags' in the object table is set to this value.

Return:

(INT): 0 = job successful, otherwise error code

std.UndoArchive

Description:

This job restores the specified document from the jukebox in the WORK directory.

Parameter:

Flags (INT): options for document status

§ 1 = documents which are written to the WORK directory receive the status 'archivable'

§ 0 = documents which are written to the WORK directory receive the status 'not archivable'

dwObjectType (INT): object type to be dearchived

dwObjectID (INT): ID of the object

Return:

(INT): 0 = job successful, otherwise error code

File administration

§ std.DeleteDocumentVersion

§ std.DeleteObject

§ std.DeleteDocument

§ std.DeleteRemark

§ std.FindDocumentDigest

§ std.GetDocStatistics

§ std.GetDocStream

§ std.GetDocumentDigest

§ std.GetDocumentPage

§ std.GetDocumentStream

§ std.GetDocVariant

§ std.SetActiveVariant

§ std.GetDocVersion

§ std.GetObjectInfo

§ std.GetRemark

§ std.GetSignedDocument

§ std.MergeDocuments

§ std.MergeFolder

§ std.RestoreDocVersion

§ std.RestoreObject

§ std.SetHistory

§ std.StoreRemark

§ std.StoreSignedDocument

§ std.Unknown2Known

§ std.SetPlannedRetention

std.FindDocumentDigest

Description:

This job looks for a document with the same hash value. The search is performed in the osdochash table, only entries with osguid = NULL or osguid = '', i.e. versions, are not checked.

Parameter:

Flags (INT): not currently used, should be 0.

Digest (STRING): the hash value being searched

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Result: Document IDs and types of found documents, in the following format: 11=131072;22=131072; etc.

std.CalcDocumentDigest

Description:

This job provides a given document with the hash value and, optionally, the signature. Hash value and signature are guaranteed by the server. To use this job, please contact the support team of OPTIMAL SYSTEMS.

Parameter:

dwObjectID (INT): ID of the object

dwObjectType (INT): selected object type

Flags (INT): currently not used, should be 0.

Pwd (STRING): if the password is incorrect, no action is performed.

Sign (BOOLEAN): if true, the document will be signed.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Digest (STRING): hash value of files.

std.DeleteDocumentVersion

Description:

This job deletes a specific version of a document and, if necessary, the corresponding digital signature.

Parameter:

Flags (INT): options for deletion

§ 0 = deletes document version only

§ 1 = deletes digital signature

OSOBJID (INT): ID of the document

OSOBJTYP (INT): Document type

OSID (STRING): version ID of the document

Return:

(INT): 0 = job successful, otherwise error code

std.DeleteObject

Description:

This job identifies an object (document, register, folder) based on the specified parameters and deletes it. The parameters 'dwParentID' and 'dwParentType' are only applied to documents. If a document has several entries in the table 'sdrel' and dwParentID = 0, all entries are deleted from this table. If this parameter is not 0, only 1 entry will be deleted.

Parameter:

sDeleteMethod (STRING): method for deletion

§ 'Delete' -> object will be deleted (without trash can) or objects which are already located in the trash can are permanently deleted

§ 'DeleteWithDocs' -> valid for cabinet/register; all registers/documents contained in the object are deleted

§ 'Recycle' -> The object is moved to the trash can.

dwObjectID (INT): ID of the object

dwObjectType (INT): object type to be deleted

dwParentID (INT): ID of the parent of the selected object

dwParentType (INT): parent type of the object to be deleted [sSpecific] (STRING): children = sub-objects are deleted, otherwise empty

Return:

(INT): 0 = job successful, otherwise error code

Return values:

[sInfo]: only returned if a register/cabinet to be deleted contains sub-objects

[Clause]: always 1

std.DeleteDocument

Description:

This job deletes the document files for a document. This document remains as a document without pages.

Parameter:

dwObjectID (INT): ID of the object

dwObjectType (INT): object type to be deleted

Return:

(INT): 0 = job successful, otherwise error code

std.DeleteRemark

Description:

This job deletes a note belonging to a document.

Parameter:

RemIdent (INT): Note ID

dwObjectType (INT): Object type

Return:

(INT): 0 = job successful, otherwise error code

std.FindDocumentDigest

Description:

This job looks for a document with the same hash value. The search is performed in the osdochash table, only entries with osguid = NULL or osguid = '', i.e. versions, are not checked.

Parameter:

Flags (INT): not currently used, should be 0.

Digest (STRING): the hash value being searched

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Result: Document IDs and types of found documents, in the following format: 11=131072;22=131072; etc.

std.GetDocStatistics

Description:

This job returns the document information for the specified object.

Parameter:

dwObjectType (INT): type of the objects

dwObjectID (INT): ID of the object

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Statistics (STRING): document information separated by '@'

§ Timestamp of the document

§ Nr. of the server group where the document is located

§ Name of the media where the document is located ('WORK' is also a valid media name)

§ Number of documents of the object

§ File size of the document in bytes (only if number = 1, slides will not be counted)

§ File name (incl. path)

§ No. of the module type

§ Note information, the contents is 'NOTE' if the note has been made on an object

§ current document state (DocState)

§ Object type

std.GetDocStream

Description:

This job was replaced by GetDocumentStream.

std.GetDocumentDigest

Description:

This job returns the digest value of the document files.

Parameter:

dwObjectID (INT): ID of the object

dwObjectType (INT): Document type

[dwFlags] (INT): 0 = digest value is read from DB; 1 = digest value is calculated.

[CheckSignature] (BOOLEAN): the signature is also checked in the case of 1. In this case, the output parameter Signature is returned.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

[LocalDigest]: Digest value that was calculated (only with flags = 1)

TableDigest: digest value from DB (flags = 0,1)

[Signature]:

0 – Signature exists and was verified

1 – Signature does not exist

2 – Signature could not be verified due to a technical error

3 – Signature exists and is invalid

std.GetDocumentPage

Description:

This job returns a specified page of a specified document.

Parameter:

dwObjectID (INT): ID of the object

Page (INT): page number of the document

Return:

(INT): 0 = job successful, otherwise error code

Return values:

nModule (INT): main document type

File list: path and name of the file that was transferred

std.GetDocumentStream

Description:

This job reads the section of a file, creates a new file out of it and returns it.

Parameter:

dwObjectID (INT): ID of the object

dwObjectType (INT): Object type

PageNum (INT): page number from which will be read

dwOffset (INT): offset (will be read from this byte)

dwLength (INT): number of bytes that will be read

Return:

(INT): 0 = job successful, otherwise error code

Return values:

File list: path and name of the created file

Count (INT): number of created files

Note:

Errors which can occur in this job.

§ ERR_WRONGPARAM: the offset is larger than the file.

§ ERR_GETOBJECTDEFPARAM: The object type is unknown.

§ ERR_GETPARAMETER: too few (valid) parameters transferred.

§ ERR_SQLSELECT: errors occurred during DB search requests.

§ ERR_NOTDEFINED_YET: The object is located on a remote server.

§ ERR_INSOURCE: the file cannot be found or opened.

§ ERR_INDESTINATION: the destination file cannot be created.

std.GetDocVariant

Description:

This job creates a new variant of the selected document.

Parameter:

sDocVer (STRING): number of the new variant

bTransferPlannedRetention (BOOLEAN) indicates if the set retention date will be applied or not.

dwParentID (INT): ID of the object

dwObjectType (INT): selected object type

dwMainType (INT): Main type of new variant

bAddFiles (INT): if 1, the new files (if attached) will be added and will not replace the old ones.

bAddFront (INT): if 1, the new files will be added in the front.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

dwVariantID: ID of newly created variant

std.SetActiveVariant

Description:

This job activates a variant.

Parameter:

dwPrevActVarID (INT): ID of the current variant

dwNextActVarID (INT): ID of the variant to be activated

dwObjectType (INT): selected object type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

std.GetDocVersion

Description:

This job returns the version of the selected document.

Parameter:

GUID (STRING): ID of the version

dwObjectID (INT): ID of the object

dwObjectType (INT): selected object type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

File list: path and name of the file, which was returned

std.GetObjectInfo

Description:

This job returns the desired information (status or size) of the selected object.

Parameter:

dwInfoFlag (INT): 0 = status of the object is determined, 1 = size of the object is determined

dwObjectType (INT): type of the objects

dwObjectID (INT): ID of the object

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ServerInfo (STRING): contains the requested information

- dwInfoFlag = 0: status is determined
  - 0 = document does not contain any pages
  - 1 = Document is archived
  - 2 = Document is not archived / cannot be archived
  - 3 = Document state cannot be determined
- dwInfoFlag = 1: size of the documents is determined (total of all object documents)
  - 0 = document does not contain any pages
  - [N] = the size of the document files (incl. SLIDE) in bytes

std.GetRemark

Description:

This job returns a note which is identified by the object type and the RemIdent (notes ID).

Parameter:

Flags (INT): must always be 0

RemIdent (INT): Note ID

dwObjectType (INT): Object type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

FileCount (INT): is always 1 since only one file is output

File list: path and file name of the searched note

std.GetSignedDocument

Description:

This job returns a digitally signed document from the server to the client.

Parameter:

OSOBJID (INT): ID of the selected object

OSID (STRING): ID of the version

OSOBJTYP (INT): selected object type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

nFileCount (INT): Number of files

dwUserID (INT): User ID

dwTimeStamp (INT): Model creation time

sTrust (STRING):

sFirstName (STRING): first name of the user

sStation (STRING): Computer name

sName (STRING): User name

sClass (STRING):

std.MergeDocuments

Description:

This job adds document pages of one specified object to another destination object. The document types of both objects must match. The source document will 'lose' its document pages after this action.

Parameter:

dwObjectID1 (INT): ID of the destination object

dwObjectType1 (INT): destination object type

PageCount1 (INT): page count of the destination object

dwObjectID2 (INT): ID of the object to be attached

dwObjectType2 (INT): type of object to be attached

PageCount2 (INT): page count of the object to be attached

Return:

(INT): 0 = job successful, otherwise error code

std.MergeFolder

Description:

This job merges two folders within the database by moving all documents and registers from the source folder to the destination folder.

Parameter:

Flags (INT): 1 = the source folder will be deleted, otherwise 0

dwObjectType (INT): object type

FolderDest (INT): ID of the destination folder

FolderSource (DWORD): ID of the source folder

Return:

(INT): 0 = job successful, otherwise error code

std.RestoreDocVersion

Description

This job restores a document version as the current document.

Parameter:

GUID (STRING): ID of the version to be restored

OSSTATION (STRING): computer name of the calling client

dwObjectID (INT): ID of the object

dwObjectType (INT): Object type

dwUserId (INT): User ID

Return:

(INT): 0 = job successful, otherwise error code

std.RestoreObject

Description:

This job restores an object from the trash can.

Parameter:

sRestoreMethod (STRING): string describing the restore method

§ SingleRestore = restore the specified object only

§ RestoreAllEntity = restore sub-objects of the specified object

dwObjectID (INT): ID of the object

dwObjectType (INT): Object type

dwParentID (INT): ID of the object to where the specified object is to be restored

dwParentType (INT): object type to where the specified object is to be restored

Return:

(INT): 0 = job successful, otherwise error code

std.SetHistory

Description:

This job adds an entry in the 'osobjhist' database tab for the indicated object.

Parameter:

sInfo (STRING): information about the action that was carried out

dwObjectID (INT): ID of the object

dwObjectType (INT): Object type

Return:

(INT): 0 = job successful, otherwise error code

std.StoreRemark

Description:

This job saves a note in a directory within the NOTE directory which is specified by the object type.

Parameter:

RemIdent (INT): Note ID

dwObjectType (INT): object type from which the directory within the NOTE directory is determined and then created.

File list: path and name of the note file (TXT) to be stored

Return:

(INT): 0 = job successful, otherwise error code

std.StoreSignedDocument

Description:

This job saves a signed document on the server.

Parameter:

OSFIRSTNAME (STRING): first name of the user

OSNAME (STRING): User name

OSOBJCLASS (STRING): Object class

OSSTATION (STRING): Workstation

OSTRUST (STRING):

OSOBJID (INT): ID of the selected object

OSTIMESTAMP (INT): Timestamp

OSOBJTYP (INT): selected object type

OSUSERID (INT): User ID

OSSIGNATURE file: file with signature content will be deleted

OSSIGHEADER file: file with signature header content will be deleted

OSSIGTEXT file: file with signature text content will be deleted

File for saving: file to be saved

Return:

(INT): 0 = job successful, otherwise error code

std.Unknown2Known

Description:

This job converts a typeless document into a document with the passed type.

Parameter:

Flags (INT):

- LOWORD(Flags) = 1 then:
  - HIWORD(Flags) = 0 --> the original module number is specified by the HIWORD parameter (dwObjectType)
  - HIWORD(Flags) != 0 --> the original module number is specified by the HIWORD parameter (flags)

dwObjectID (INT): ID of the object

dwObjectType (INT): object type to which the specified document is to be transferred

Return:

(INT): 0 = job successful, otherwise error code

std.SetPlannedRetention

Description:

This job sets the intended retention time for a document.

Parameter:

Flags (INT): not currently supported, should be 0

dwObjectID (INT): ID of the object

dwObjectType (INT): object type to which the specified document is to be transferred

sRetentionDate (STRING): retention date in the format YYYY/MM/DD

Output parameters:

sRetentionDate (STRING): set retention time in the format YYYY/MM/DD

Return:

(INT): 0 = job successful, otherwise error code

std.AdjustRetentions

Description:

The job adjusts the retention date of an archived document to the scheduled retention date.

Parameter:

Flags (INT): not currently supported, should be 0

dwObjectID (INT): object ID (optional)

dwObjectType (INT): object type. If dwObjectID is not indicated, documents of this type will be adjusted.

Output parameters:

sRetentionDate (STRING): set retention time in the format YYYY/MM/DD; the parameter is set only if dwObjectID was indicated.

Return:

(INT): 0 = job successful, otherwise error code

Internal Jobs

Internal jobs are generally used only by the standard DMS engine itself.

§ std.ObjectTransfer

std.ObjectTransfer

Description:

This job transfers an object from a foreign (not local) work directory into the local work or cache directory or writes the complete local work directory into the local cache directory or deletes the local work directory.

Parameter:

sAction (STRING): action to be executed

§ FromForeignWorkToLocalWork = transfers an object from a foreign (not local) work directory into the local work directory

§ FromForeignWorkToLocalCache = transfers an object from a foreign (not local) work directory into the local cache directory

§ MoveLocalWorkToLocalCache = writes the whole local work directory to the local cache directory

§ DeleteLocalWork = deletes the local work directory

dwObjectID (INT): ID of the object

dwObjectType (INT): selected object type

Return:

(INT): 0 = job successful, otherwise error code

Other jobs

§ std.CheckSource

§ std.ConfigVarc

- std.DiskSpace
- std.FileTransfer
- std.GetTemplates
- std.IndexDataChanged
- std.PackDirectory
- std.TransformIndexData
- std. ZipDocument

std.CheckSource

Description:

This job replaces a path variable in the format '%ETCPATH%' with an absolute path and checks whether the indicated file exists in this path.

Parameter:

Flags (INT): not currently supported-> transfer 0

Source (STRING): path variable to be replaced and the name of the file whose existence is to be checked

Return:

(INT): 0 = job successful, otherwise error code

Return values:

State (INT): 0 = file does not exist; 1 = file exists

Note:

supported path variables

- %SERVERROOT%
- %ETCPATH%
- %WORKPATH%
- %LOGPATH%
- %CLIENTETC%

std.ConfigVarc

Description:

This job can only be executed using the Enterprise Manager. This job configures the parameters of the virtual archives.

Parameter:

SAction (STRING): action to be executed

- Get = returns the parameters of VARC
- Set = sets the parameters of VARC

Return:

(INT): 0 = job successful, otherwise error code

std.DiskSpace

Description:

This job determines the free storage space on the hard disk where the work directory is located and sends the information by e-mail to the system administrator (defined in the registry).

Return:

(INT): 0 = job successful, otherwise error code

std.FileTransfer

Description:

This job transfers one or more files from a source path to a destination path (flags = 0/2), deletes the indicated files (flags = 4) or provides the digest value of a file (flags = 5).

Parameter:

Flags (INT): job options (see table)

FileCount (INT): Number of passed File_ parameters

File_[0..n] (STRING): see table

File list: see table

Return values:

Count (INT): always 0 (no longer used)

[Digest]: digest value of the file (only with flags = 5)

Note:

Only certain parameters will be used depending on the respective job options. The following table depicts the use of parameters.

Input parameters				Return
Flags	**FileCount**	**File_[0..n]**	**File list**	**Digest**

0 = files will be copied with the WinApi 32 function 'CopyFile'	Number of passed File_ parameters	Without input files:
		File_0 = destination path+file name
		File_1 = source path+file name
		File_2 = destination path+file name
		File_3 = source path+file name
		etc.
		With input files:
		File_0 = destination path+file name
		File_1 = destination path+file name
		etc.
			Path and name of the source file(s)	no

2 = files are	Number of	Without input files:	Path and name of	no

copied with the WinApi32 functions 'CreateFile', 'ReadFile' and 'WriteFile'	passed File_ parameters	File_0 = destination path+file name
		File_1 = source path+file name
		File_2 = destination path+file name
		File_3 = source path+file name
		etc.
		With input files:
		File_0 = destination path+file name
		File_1 = destination path+file name
		etc.
			the source file(s)	

4 = files will be deleted	Number of passed File_ parameters	File_0 = path+file name
		File_1 = path+file name
		etc.
			no	no

5 = digest value will be calculated	Number of passed File_ parameters	File_0 = path+file name
		File_1 = path+file name
		etc.
			no	Calculated digest value for file(s)

std. GetTemplates

Description:

This job returns a list of all templates for the indicated object type. If the parameter 'dwObjectType' is not passed or passed with the value -1, the templates will be returned for all object types.

Parameter:

[dwObjectType] (INT): Object type for which the templates are to be returned

Return values:

nCount (INT): Number of templates

ObjectType[1 ... nCount] (INT): selected object type

TemplateId[1 ... nCount] (INT): template ID

Aliases[1 ... nCount] (STRING): alias name

Editor[1 ... nCount] (STRING): editor name

FileName[1 ... nCount] (STRING): File name

Extension[1 ... nCount] (STRING): extension name

NameSpace[1 ... nCount] (STRING): namespace name

Return:

(INT): 0 = job successful, otherwise error code

std.IndexDataChanged

Description:

This job tells the index server that the index data of an object have changed.

Parameter:

Flags (INT): 1 = the 'osowner' field in DB tab 'objectXXX' is changed, otherwise the 'osowner' field is not changed

Action (INT): action executed with the object (actions are listed in the asdll.h file)

dwObjectID (INT): ID of the object

dwObjectType (INT): Object type

Info (STRING): information that is written to the 'osobjhist' database tab

GUID (STRING): unique key of the 'osobjhist' DBtab

Return:

(INT): 0 = job successful, otherwise error code

std.PackDirectory

Description:

This job packs the contents of the full text directory into a CAB file and sends the file to the receiving server. The receiving server unpacks the CAB file and adds all full text files into its full text directory.

Parameter:

dwPort (INT): IP port of the receiving server

sComString (STRING): IP address of the receiving server in the format addr1#port1;addr2#port2; etc. 'addr' corresponds to the ComString column from the server table.

sAction (STRING): options for this job

§ Pack = pack the contents of the full text directory into a CAB file

§ Send = send CAB file

§ DeleteSource = contents of the full text directory is deleted

sRoot (STRING): directory from which subdirectories and files are to be unpacked

dwCabSize (INT): max. size of the CAB file in KB

sCabDir (STRING): directory into which the CAB files are to be saved

sAddress (STRING): string in which all servers of the destination group are listed (in the format AS.ini file)

Return:

(INT): 0 = job successful, otherwise error code

std.TransformIndexData

Description:

This job exports full text index data according to its parameters. This job is called in the archive by axacwexp.dll.

Parameter:

dwObjectType (INT): Object type

sAction (STRING): action options

§ DeleteIndex = full text index data will be deleted

§ NewIndex = new full text index data will be inserted

§ UpdateIndex = as 'NewIndex', but additionally to RW the field '<UPDATE>' will be inserted as well.

bIgnoreFieldSpecific (BOOL): full text properties for fields

§ 0 = full text properties apply to every field

§ 1 = full text properties are ignored by every field

[dwObjectID] (INT): object ID (if ID = 0 or not available in ListParameterIn, then all objects with type = dwObjectType are considered. Otherwise only this object will be taken into account.)

[bWriteFiles] (BOOL): export options (default value is 0)

§ 0 = only index data (without document files) will exported for full text search

§ 1 = document files will be exported for full text search

[iAction] (INT): action to be executed (defined in the file asdll.h)

std. ZipDocument

Description:

This job packs one or more file(s).

Parameter:

bDeleteSource (INT): 1 = source files are deleted, otherwise 0

Input file list: name and path of the file(s)

Return:

(INT): 0 = job successful, otherwise error code

Full-Text Engine (Namespace vtx)

The full text engine is for processing the clients' full text requests. Various search engines can be transparently integrated for the client.

The following full text search engines are supported at the moment:

name	Short name (registry)
Microsoft SQL Server Full Text	ms
OSFTS	lu
Microsoft Index Server (discontinued)	mi
Convera RetrievalWare	rw

The OSFTS engine is often also referred to as the Lucene engine or simply osfts.

The Microsoft Index Server is still supported, but it is no longer recommended for new installations.

Full text engine jobs

§ vtx.CleanupClient

§ vtx.CloseQuery

§ vtx.GetDocument

§ vtx.GetEngineName

§ vtx.GetSimilarDMSObjects

§ vtx.GetMaxHits

§ vtx.IsOntologySearchEnabled

§ vtx.IsSearchForSimilarDMSObjectsEnabled

§ vtx.OpenObjectQuery

§ vtx.OpenWordListQuery

§ In each job description below "Engines", the short name of the full text engine is noted that supports the job. If a full text engine does not support a job, executing the job has no effect and an empty result set will be returned.

vtx.CleanupClient

Description:

This job deletes all full-text queries for the indicated client application and frees related resources.

Parameter:

Flags (INT): not currently supported

Client (STRING): Client

Return:

(INT): 0 = job successful, otherwise error code

Engines:

rw

vtx.CloseQuery

Description:

This job closes a query which is identified by a GUID, thereby freeing resources.

Parameter:

Flags (INT): not currently supported

Guid (STRING): query guid. See GUID return parameter for OpenObjectQuery.

Return:

(INT): 0 = job successful, otherwise error code

Engines:

lu, rw

vtx.GetDocument

Description:

This job returns a document which is identified by the query Guid and the HitId.

Parameter:

Flags (INT): not currently supported

Guid (STRING): query guid. See GUID return parameter for OpenObjectQuery.

HitId (INT): document's hit ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

File list: path and name of the requested file

Engines:

lu, rw

vtx.GetEngineName

Description:

This job returns the short name of the full text engine (e.g. ms: for Microsoft SQL Server Full Text) from the registry (path VTX -> engine).

Parameter:

Flags (LONG): not currently used

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Result (STRING): short name of the full text engine. See list of supported full text engines.

Engines:

All engines

vtx.GetSimilarDMSObjects

Description:

This job returns a list for a DMS object containing other objects whose text is similar to the contents of the passed object.

Note:

This search for objects is currently only supported if OSFTS is used as the full-text engine, 'intrafind' is configured there as the analyzer, and the OKM license is available. Whether the search is supported, can be determined with vtx.IsSearchForSimilarDMSObjectsEnabled.

Parameter:

ObjectID (INT): ID of the DMS object for which similar objects are to be found.

MaxHits (INT): optional parameter that determines the maximum number of hits to be returned. If this parameter is not passed, the maximum number of hits will be determined by registry entries.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Result (STRING): consists of an object ID, an object type, the ranking, and a hit ID (comma-separated) for every found document (comma-separated).

Engines:

lu

See also:

vtx.IsSearchForSimilarDMSObjectsEnabled

vtx.IsOntologySearchEnabled

Description:

This job indicates whether the full text engine supports the search for related words.

Parameter:

- none

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Result (BOOL): Is the search for similar words supported?

Engines:

All engines. But TRUE currently only for 'lu'.

See also:

vtx.OpenObjectQuery

vtx.IsSearchForSimilarDMSObjectsEnabled

Description:

This job indicates whether the full text engine supports the search for DMS documents with similar content.

Parameter:

- none

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Result (BOOL): Is the search for documents with similar content supported?

Engines:

All engines. But TRUE currently only for 'lu'.

See also:

vtx.GetSimilarDMSObjects

vtx.GetMaxHits

Description:

This job returns the upper limit for the maximum number of hits for full text searches which has been configured on the server. For all full text search jobs, it is recommended that the upper limit indicated there does not exceed the server-side maximum.

Parameter:

- none

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Result (INT): server-side configured upper limit for the maximum number of hits, which clients should use when calling full text search jobs.

Engines:

All engines.

See also:

vtx.OpenObjectQuery, vtx.GetSimilarDMSObjects, vtx.OpenWordListQuery

vtx.OpenObjectQuery

Description:

This job queries an object and returns the result.

Note:

This ontological search (query parameter ONTOLOGY=1) is currently only supported if OSFTS is used as the full-text engine, 'intrafind' is configured there as the analyzer, and the OKM licence is available. Using vtx.IsOntologySearchEnabled can determine whether the search is supported.

If OSFTS is used with 'intrafind,' the LIS license is generally also required, otherwise the job call returns the error code 0xC1DA0BDA.

Parameter:

Flags (INT): flags that are transferred to the full text engine

Query (STRING): search request in INI format. See example.

MessageLanguageId (INT): optional parameter for the ID of the language in which the possibly returned message (see return value 'Message') is to be composed (Windows Language IDs, see for example http://msdn.microsoft.com/en-us/library/ms776294.aspx; e.g. 7=German)

Return:

(INT): 0 = job successful, otherwise error code, e.g. 0xC1DA0BDA, if OSFTS with 'intrafind' is used and the license LIS is missing.

Return values:

Result (STRING): consists of an object ID, an object type, the ranking, and a hit ID (comma-separated) for every found document (comma-separated).

AlternativeQuerySuggestion (STRING): suggestion for alternative queries. This value is only returned if OSFTS is used as the full-text engine and if this provides an alternative suggestion.

OntologyTerms (STRING): list of related search terms. Format: <Term>, <Degree of relation (indicated in percent)>, ... For example: 'flight,55;travel,43;vacation,12;'. The return value is only returned when ONTOLOGY = 1 has been set in the query

Guid (STRING): Guid

Message (STRING): search result message

Engines:

All engines.

Example:

Only the first three lines are required when using Microsoft SQL Server.

```
[PAGE00]
#OSTYPE#=262144
FULLTEXT=horse
ONTOLOGY=1
#OSACT#=1
FELD0=#OSPOS000#;Author;field1;X;50;0;0
FELD1=#OSPOS001#;source;field2;X;150;0;0
FELD2=#OSPOS002#;Text2;field3;X;50;0;0
[PDMSPParams]
EXPANSION_LEVEL_PROPERTY=4
FUZZY_SPELL_HALF_WORDS=FALSE
FUZZY_SPELL_THRESHOLD=0
MAX_FUZZY_SPELL_PROPERTY=15
MAX_REG_EXPR_PROPERTY=50
WARN_MAX_REG_EXPR_PROPERTY=FALSE
```

```
WORD_EXPANSION_LIMIT_PROPERTY=20
LMPI_SET_LANG_PROPERTY=de
MAX_DOCS_PROPERTY=999
RwareQueryType=P
OSQueryType=Object
OSSelectedTypes=current
```

See also:

vtx.IsOntologySearchEnabled

vtx.OpenWordListQuery

Description:

This job queries a word list. At the moment, this job can only return a useful result when RetrievalWare is used as full text software.

Parameter:

Flags (INT): flags that are transferred to the full text engine

Query (STRING): Query

Param (STRING): parameter for the search request

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Result (STRING): consists of an object ID, an object type, the ranking, and a hit ID (comma-separated) for every found document (comma-separated).

Guid (STRING): Guid

Engines:

rw

Workflow Engine (Namespace wfm)

Jobs for processing and managing workflow processes and models are prepared here.

It is not possible to call workflow jobs from server-side workflow events through the 'running context'. This causes the server to crash.

Areas

Organizational structure

wfm.ConfigUserAbsence

Description:

This job defines one or more users as absent/present and informs all servers and enaio® editor-for-workflow.

Parameter:

OrganizationId (STRING): user organization

Users (BASE64): user list in XML format

Return:

(INT): 0 = job successful, otherwise error code

Example:

Structure of Users

```
<Users>
<User Id="" Absent="1"/>
<User ID="" Absent="0"/>
</Users>
```

Note:

Detailed description of users

User: structure with the following contents

- ID (STRING): User ID
- Absent (LONG): Flag
  - 0 = user is present
  - 1 = user is absent

See also:

wfm.GetOrganizations, wfm.GetOrganizationObjects, wfm.GetAbsentUsers

wfm.DeleteOrganisation

Description:

This job deletes an organization. All database entries regarding the organization are deleted from the database (including organizational structure and workflow models)

Parameter:

OrganizationId (STRING): organization ID

Return:

(INT): 0 = job successful, otherwise error code

See also:

wfm.GetOrganizations

wfm.GetAbsentUsers

Description:

This job returns all users of an organization which are set to 'absent'.

Parameter:

OrganizationId (STRING): organization ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ObjectIds (String): GUIDs of the absent user objects, separated by commas

See also:

wfm.GetOrganizations, wfm.ConfigUserAbsence

wfm.GetOrganisationClasses

Description:

This job returns information on the classes of an organization.

Parameter:

OrganizationId (STRING): Organization ID

RequestType (LONG): flag specifies the search request, with the following parameters accordingly set

§ 0 = search all classes

§ 1 = the search is performed for IDs in the parameter 'ClassIds'

§ 2 = the search is performed for names in the parameter 'ClassName'

ClassIds (STRING): comma-separated list of class IDs

ClassName (STRING): class name

AttributeId (STRING): not currently supported

RequestData (INT): specifies the results of the search request

§ 1 = only determine index data of the classes (object ID, name, class ID)

§ 3 = determine index data and class attributes

§ 5 = determine index data and class relations

§ 7 = determine index data, class attributes and class relations

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Classes (BASE64): information on the requested classes in XML format

Example:

Structure of Classes

```
<Classes>
<Class Id="10B9A20E90244BB9B701354C1AB84F8A" Name="department">
<Attributes/>
<ParentObjects>
<ParentObject Id="894CB679F2ED480A89107BF33A1F" Name="organization"/>
</ParentObjects>
<ChildClasses>
<ChildClass Id="12AA95D1D8244E6BB56C70A8D5CEE675" Name="role"/>
</ChildClasses>
</Class>
<Class Id="9AB24246BB9040A29FCD6015CF4F4BD9" Name="person">
<Attributes>
<Attribute Id="33D4AB4B39" Name="surname" AttributeClassId="07F405D">
<AttributeValue><![CDATA[]]></AttributeValue>
</Attribute>
</Attributes>
<ParentObjects>
<ParentObject ID="12AA95D1D8244E6BB56C70A8D5CEE675" Name="role"/>
</ParentObjects>
<ChildClasses/>
</Class>
</Classes>
```

Note:

Detailed description of Classes

- Class: structure containing information on an organization class
    - ID (STRING): class ID
    - Name (STRING): class name
- Attributes: structure containing information on an object attribute
    - ID (STRING): attribute ID
    - Name (STRING): attribute name
    - AttributeClassId (STRING): attribute class ID of the attribute
    - AttributeValue: CDATA with attribute value, where applicable MIME encoded
- ParentObject: structure containing information on an organization object, which is in the organization tree directly above the current class.
    - ID (STRING): ID of the object
    - Name (STRING): object name
- ChildClass: structure containing information on an organization object, which is in the organization tree directly below the current class.
    - ID (STRING): ID of the object
    - Name (STRING): object name

See also:

wfm.GetOrganizations, wfm.GetOrganizationObjects

wfm.GetOrganisationObjects

Description:

This job returns information on the objects of an organization.

Parameter:

OrganizationId (STRING): Organization ID

RequestType (INT): flag specifies the search request, with the following parameters accordingly set

- 0 = find all objects
- 1 = the search is performed for IDs in the parameter 'ObjectIds'
- 2 = the search is performed for names in the parameter 'ObjectName'
- 3 = the search is performed for IDs in the parameter 'ClassIds'
- 4 = the search is performed for names in the parameter 'ClassName'
- 5 = the search is performed for predecessor objects in the parameter 'ObjectName'
- 6 = the search is performed for successor objects in the parameter 'ObjectName'
- 7 = the search is performed for predecessor objects in the parameter 'ClassName'
- 8 = the search is performed for successor objects in the parameter 'ClassName'

ObjectIds (STRING): comma-separated list of object IDs

ObjectName (STRING): Object name

ClassIds (STRING): comma-separated list of class IDs

ClassName (STRING): class name

AttributeId (STRING): not currently supported

AttributeValue (STRING): not currently supported

RequestData (INT): specifies the results of the search request

- 1 = only determine index data of the objects (object ID, name, class ID)
- 3 = determine index data and object attributes
- 5 = determine index data, predecessor and successor objects
- 7 = determine index data, object attributes, predecessor and successor objects

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Objects (BASE64): information on the requested objects in XML format

Example:

Structure of Objects

```
<Objects>
<Object Id="" Name="" ClassId="">
<Attributes>
<Attribute Id="" Name="" AttributeClassId="">
<AttributeValue><![CDATA[]]</AttributeValue>
</Attribute>
<Attribute Id="" Name="" AttributeClassId="">
<AttributeValue><![CDATA[]]</AttributeValue>
</Attribute>
</Attributes>
<ParentObjects>
<ParentObjects Id="" Name="" ClassId="">
</ParentObjects>
<ChildObjects>
<ChildObjects Id="" Name="" ClassId="">
</ChildObjects>
</Object>
</Objects>
```

Note:

Detailed description of objects

- Object: structure containing information on an organization object
  - ID (STRING): ID of the object
  - Name (STRING): object name
  - ClassId (STRING): class ID of the object
- Attributes: structure containing information on an object attribute
  - ID (STRING): attribute ID
  - Name (STRING): attribute name
  - AttributeClassId (STRING): attribute class ID of the attribute
  - AttributeValue: CDATA with attribute value, where applicable MIME encoded

- ParentObject: structure containing information on an organization object, which is in the organizational tree directly above the current object.
    - ID (STRING): ID of the object
    - Name (STRING): object name
    - ClassId (STRING): class ID of the object
- ChildObject: structure containing information on an organization object, which is in the organization tree directly below the current object.
    - ID (STRING): ID of the object
    - Name (STRING): object name
    - ClassId (STRING): class ID of the object

See also:

wfm.GetOrganizations, wfm.GetOrganizationClasses

wfm.GetOrganisations

Description:

This job returns all defined organizations and indicates which organization is enabled. Only one organization at a time can be enabled in the enaio® editor-for-workflow.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Organizations (BASE64): information on all available organizations in XML format

Example:

Structure of Organisations

```
<Organizations>
<Organization Id="" Name="" Active="0"/>
<Organization Id="" Name="" Active="1"/>
</Organizations>
```

Note:

Detailed description of Organisations

Organization: structure containing information on an organization

- ID (STRING): Organization ID
- Name (STRING): organization name
- Active (INT): indicates whether the organization is enabled (1) or not (0)

wfm.GetSubstitutes

Description:

This job returns a substitute for any number of users or roles.

Parameter:

OrganizationId (STRING): Organization ID

UserIds (STRING): comma-separated list of IDs of users/roles for which substitute information is requested

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Substitutes (BASE64): requested substitute information in XML format

Example:

Structure of Substitutes

Example:

Structure of Substitutes

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Substitutes>
<Object ID="BB4FF62D3FE24DD790DE342585917A36">
<Substitute ID="8CD236F862644DAD904CD8C228EF4F23" Name="LEHMANN" Absent="1"
UserGUID="4FEEDE694EB94E7E9C4A847FC32D10E4" Login="Lehmann"/>
</Object>
</Substitutes>
```

Note:

Detailed description of Substitutes

- § ID (String): object ID of the user
- § Substitute: structure which contains all substitute assignments for a specific user/roles
  - § ID (STRING): object ID of the substitute for user/role
  - § Name (STRING): login name of the substitute
  - § Absent (INT): 1 = user is absent, 0 = otherwise
  - § UserGUID (STRING): user GUID from the user table
  - § Login (STRING): user login from the user table

See also:

wfm.GetOrganizations, wfm.GetOrganizationObjects, wfm.SetSubstitutes

wfm.GetUserSubstitutes

Description:

This job identifies all users for whom the querying user is covering

Parameter:

OrganizationId (STRING): Organization ID

UserId (STRING): user ID for whom the information is requested

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Objects (BASE64): requested information in XML format

Example:

Structure of Objects

Example:

Structure of Objects

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Objects>
<Object ID="BB4FF62D3FE24DD790DE342585917A36">
<Substitute ID="8CD236F862644DAD904CD8C228EF4F23" Name="LEHMANN" Absent="1"
UserGUID="4FEEDE694EB94E7E9C4A847FC32D10E4" Login="Lehmann"/>
</Object>
</Objects>
```

Note:

Detailed description of objects

§ Id (String): object ID of the user

§ Object, structure that contains all users to be substituted

  § ID (STRING): object ID of the substitute for user/role

  § Name (STRING): login name of the substitute

  § Absent (INT): 1 = user is absent, 0 = otherwise

  § UserGUID (STRING): user GUID from the user table

  § Login (STRING): user login from the user table

See also:

wfm.GetOrganizations, wfm.GetOrganizationObjects, wfm.SetSubstitutes

wfm.SaveOrganisation

Description:

This job saves an organization and notifies all affected clients.

Parameter:

OrganizationId (STRING): organization ID

Return:

(INT): 0 = job successful, otherwise error code

wfm.SetActiveOrganisation

Description:

This job enables the specified organization.

Parameter:

OrganizationId (STRING): organization ID

Return:

(INT): 0 = job successful, otherwise error code

wfm.SetSubstitutes

Description:

This job sets a substitute for any number of users or roles.

Parameter:

OrganizationId (STRING): Organization ID

Substitutes (BASE64): user substitute assignments in XML format

Return:

(INT): 0 = job successful, otherwise error code

Example:

Structure of Substitutes

```
<Substitutes>
<Substitute ID="12345678901234567890123456789010">
<SubstituteIds>789012345678901A,123456778901B</SubstituteIds>
</Substitute>
<Substitute ID="12345678901234567890123456789011">
<SubstituteIds>123456789012345,123452345678901C</SubstituteIds>
</Substitute>
</Substitutes>
```

Note:

Detailed description of Substitutes

Substitute: structure containing all substitute assignments for a specific user

- ID (STRING): ObjectId of the user/role
- comma-separated list of object IDs of all substitutes for the user/role can be empty

See also:

wfm.GetOrganizations, wfm.GetOrganizationObjects, wfm.GetSubstitutes

workflow model

- wfm.ChangeWorkflowState
- wfm.CopyWorkflow
- wfm.DeleteWorkflow
- wfm.GetWorkflow
- wfm.GetWorkflowData
- wfm.GetWorkflowInfo
- wfm.GetWorkflowList
- wfm.GetWorkflowListByFamily
- wfm.StoreWorkflow
- wfm.ValidateWorkflow

wfm.ChangeWorkflowState

Description:

This job changes/sets the status of a workflow model.

Parameter:

OrganizationId (STRING): ID of the organization assigned to the workflow

WorkflowId (STRING): ID of the workflow model

UserId (STRING): User ID

State (LONG): new status of the workflow model to be set

Return:

(INT): 0 = job successful, otherwise error code

Return values:

State (LONG): new status of the workflow model

Note:

Status of the workflow model

§ 1 = The model is in use, i.e. new processes can be started with it.

§ 2 = The model is locked for editing.

§ 3 = The model is still being edited, but is not locked.

§ 4 = The model is available for testing.

§ 5 = The model has been deleted, but is still contained in the database.

§ 6 = The model is available, but is not yet in use. New processes cannot be started with this model, active ones are being terminated.

See also:

wfm.ValidateWorkflow

wfm.CopyWorkflow

Description:

This job creates a copy of a workflow model including workflow forms and events.

Parameter:

UserId (STRING): ID of the executing user

SourceOrganizationId (STRING): ID of the organization to be copied from

SourceWorkflowId (STRING): ID of the workflow model to be copied

TargetOrganizationId (STRING): ID of the organization to be copied to

TargetFamilyId (STRING): ID of the workflow family to be copied to

TargetWorkflowName (STRING): name of the new workflow

Return:

(INT): 0 = job successful, otherwise error code

Return values:

WorkflowId (STRING): ID of the new workflow model

wfm.DeleteWorkflow

Description:

This job deletes a workflow model and the corresponding history entries from the database and notifies all other servers.

Parameter:

OrganizationId (STRING): ID of the organization of the workflow model

WorkflowId (STRING): ID of the workflow model

Return:

(INT): 0 = job successful, otherwise error code

wfm.GetWorkflow

Description:

This job returns a workflow model from the server.

Parameter:

WorkflowId (STRING): ID of the workflow model

ProcessId (STRING): ID of the process (the process has to be active); can be optionally used to replace the parameters 'FamilyId' and 'WorkflowId'; only valid for Action = 1 valid

OrganizationId (STRING): ID of the organization of the workflow model

UserId (STRING): ID of the DRT user

FamilyId (STRING): ID of the workflow family of the workflow model

Action (INT): action to be executed

§ 1 = the workflow model is queried for reading

§ 2 = the workflow model will be edited

§ 3 = an empty workflow model is queried

Return:

(INT): 0 = job successful, otherwise error code

Return values:

NewId (STRING): new ID of the requested workflow, if created (see input parameter 'Action')

State (INT): status of the model

§ 1 = The model is in use, i.e. new processes can be started with it.

§ 2 = The model is locked for editing.

§ 3 = The model is still being edited, but is not locked.

§ 4 = The model is available for testing.

§ 5 = The model has been deleted, but is still contained in the database.

§ 6 = The model is available, but is not yet in use. New processes cannot be started with this model, active ones are being terminated.

FamilyId (STRING): ID of the workflow's family

CreationTime (INT): workflow creation time

Version (INT): workflow version

File list: name and path of the file containing the XML package with the description of the model in XML format

See also:

wfm.StoreWorkflow

wfm.GetWorkflowData

Description:

This job returns status information of a workflow model for a workflow ID

Parameter:

OrganizationId (STRING): ID of the organization of the workflow model

WorkflowId (STRING): ID of the workflow model

Return:

(INT): 0 = job successful, otherwise error code

Return values:

WorkflowId (STRING): ID of the workflow model

WorkflowName (STRING): name of the workflow model

WorkflowState (INT): status of the workflow model

WorkflowCreator (STRING): ID of the workflow model creator

WorkflowCreationTime (INT): creation time of the workflow model

(STRING): version of the workflow model

WorkflowLockId (STRING): ID of the user who has locked the workflow model

WorkflowLockName (STRING): name of the user who has locked the workflow model

WorkflowLockTime (INT): lock time of the workflow model

WorkflowDescription (STRING): short description of the workflow model

WorkflowIconId (INT): icon ID of the workflow model

Note:

Status of the workflow model

§ 1 = The model is in use, i.e. new processes can be started with it.

§ 2 = The model is locked for editing.

§ 3 = The model is still being edited, but is not locked.

§ 4 = The model is available for testing.

§ 5 = The model has been deleted, but is still contained in the database.

§ 6 = The model is available, but is not yet in use. New processes cannot be started with this model, active ones are being terminated.

wfm.GetWorkflowInfo

Description:

This job returns the input parameters (workflow variables) of the the specified workflow family for the active workflow model (status = 1).

Parameter:

OrganizationId (STRING): ID of the organization of the workflow model

FamilyId (STRING): ID of the family of the active workflow model

Return:

(INT): 0 = job successful, otherwise error code

Return values:

InputParams (BASE64): input parameters of the model in XML format

Example:

Structure of InputParams

```
<InputParams>
<InputParam Id="" Name=""><![CDATA[]]></InputParam>
<InputParam Id="" Name=""><![CDATA[]]></InputParam>
</InputParams>
```

Note:

Detailed description of InputParams

- InputParam
  - ID (STRING): ID of the input parameter
  - Name (STRING): Name of the input parameter
  - CDATA: structure of the input parameter

See also:

wfm.GetWorkflowList

wfm.GetWorkflowList

Description:

This job returns a list of all startable workflows for the specified user.

Parameter:

OrganizationId (STRING): Organization ID

UserId (STRING): User ID

Flags (INT): indication parameter, currently only value 48 is valid

ClientTypeId (STRING): ID of the used client type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Workflows (BASE64) list containing data of the queried workflows in XML format

Example:

Structure of Workflows

```
<Workflows>
<Workflow FamilyId="" ModelName ="" Name="" Id="" Description="" IconId=""/>
<Workflow FamilyId="" ModelName="" Name="" Id="" Description="" IconId=""/>
</Workflows>
```

Note:

Detailed description of Workflows

- Workflow
  - FamilyId (STRING): ID of the workflow family of the workflow
  - ID (STRING): workflow ID
  - Name (STRING): name of the workflow (instance name)
  - Description (STRING): workflow model description
  - IconId (INT): icon ID of the workflow model
  - ModelName (STRING): name of the workflow model

See also:

wfm.GetOrganizations, wfm.GetOrganizationObjects, wfm.CreateProcessInstance

wfm.GetWorkflowListByFamily

Description:

This job returns all contained workflows for a workflow family. Within a workflow family only one workflow can have the status = 1.

Parameter:

OrganizationId (STRING): Organization ID

FamilyId (STRING): ID of the workflow family

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Workflows (BASE64): Workflow list in XML format

Example:

Structure of Workflows

```
<Workflows>
<Workflow Id="" Name="" State="" Creator="" CreationTime=""
Version="" LockId="" LockName ="" LockTime="" Description=""
IconId=""/>
<Workflow Id="" Name="" State="" Creator="" CreationTime=""
Version="" LockId="" LockName ="" LockTime="" Description=""
IconId=""/>
</Workflows>
```

Note:

Detailed description of Workflows

Workflow: structure containing information on a workflow

- ID (STRING): workflow ID
- Name (STRING): name
- State (INT): workflows status
  - 1 = The model is in use, i.e. new processes can be started with it.
  - 2 = The model is locked for editing.
  - 3 = The model is still being edited, but is not locked.
  - 4 = The model is available for testing.
  - 5 = The model has been deleted, but is still contained in the database.
  - 6 = The model is available, but is not yet in use. New processes cannot be started with this model, active ones are being terminated.
- Creator (STRING): Creator
- CreationTime (INT): Creation time
- Version: workflow version number
- LockId (STRING): ID of the user who has locked the workflow
- LockName (STRING): Name of the user who has locked the workflow
- LockTime (INT): lock time
- Description (STRING):workflow description
- IconId (INT): icon ID of the workflow model

wfm.StoreWorkflow

Description:

This job changes/saves a workflow model.

Parameter:

UserId (STRING): User ID

WorkflowId (STRING): ID of the workflow model

OrganizationId (STRING): Organization ID

FamilyId (STRING): ID of the workflow family

Flags (INT): 1=internal 2=external (the model is imported from an external source, a new ID is created for it)

Input file: path and name of the file containing workflow information in XML format

Return:

(INT): 0 = job successful, otherwise error code

Return values:

WorkflowId (STRING): workflow ID

Name (STRING): workflow name

State (INT): workflows status

CreatorId (STRING): ID of the workflow creator

CreationTime (INT): workflow creation time

Version (STRING): workflow version

LockId (STRING): ID of the user who has locked the workflow

LockName (STRING): name of the user who has locked the workflow

LockTime (INT): time of locking of the workflow

Description (STRING): short description of the workflow

IconId (INT): icon ID of the workflow model

wfm.ValidateWorkflow

Description:

This job checks whether a workflow model is allowed.

Parameter:

OrganizationId (STRING): ID of the organization

Input file: path and name of the file with description of the workflow model in XML format

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Valid (INT): flag which indicates whether the model is permissible (1=yes, 2=no)

ErrorCount (INT): number of errors found in the model

WarningCount (INT): number of warnings related to the model

Errors (BASE64): information on the found errors in XML format

Warnings (BASE64): information on the warnings in XML format

Example:

Structure of Errors

```
<Errors>
<SchemaValidationError>SchemaValidationError!</SchemaValidationError>
<MissingTypeDeclRecordMembers>
<MissingTypeDeclRecordMember TypeDeclId="" TypeDeclName="TestList"/>
<MissingTypeDeclRecordMember TypeDeclId="" TypeDeclName="RecFu"/>
</MissingTypeDeclRecordMembers>
<MissingVariableRecordMembers>
<MissingVariableRecordMember VariableId="" VariableName=" TestList1"/>
</MissingVariableRecordMembers>
<MissingTypeDeclarations>
<MissingTypeDeclaration Id="12346798134567891345678900001">
<ReferencingDataFields>
<ReferencingDataField Id="1234671345678913456789001" Name="abc"/>
<ReferencingDataField Id="126789134567890000" Name="xyz"/>
</ReferencingDataFields>
<ReferencingTypeDeclarations>
<ReferencingTypeDeclaration Id="123467981345" Name=" TestList"/>
</ReferencingTypeDeclarations>
</MissingTypeDeclaration>
```

```
<MissingTypeDeclaration Id="1234679813456789134567890000 02">
<ReferencingTypeDeclarations>
<ReferencingTypeDeclaration Id="" Name="recBla"/>
</ReferencingTypeDeclarations>
</MissingTypeDeclaration>
</MissingTypeDeclarations>

<MissingActivityParticipants>
<MissingActivityParticipant Id="12346798134567891345678900006">
<ReferencingActivities>
<ReferencingActivity Id="" Name="step1"/>
<ReferencingActivity Id="" Name="step2"/>
</ReferencingActivities>
</MissingActivityParticipant>
<MissingActivityParticipant Id="12346798134567891345678900007">
<ReferencingActivities>
<ReferencingActivity Id="" Name="step1"/>
</ReferencingActivities>
</MissingActivityParticipant>
</MissingActivityParticipants>
<NoParticipantsInStartActivity/>
<MissingToolIds>
<MissingToolId ActivityId="" ActivityName="stepLoop1"/>
<MissingToolId ActivityId="" ActivityName="stepLoop2"/>
</MissingToolIds>
<MissingApplicationMasks>
<MissingApplicationMask ApplicationId="" ApplicationName="Application
Step1" MaskId="12346798134567891345678900008"/>
</MissingApplicationMasks>
<MissingApplicationMaskIds>
<MissingApplicationMaskId ActivityId="" ActivityName="Act Step1" ApplicationId=""
ApplicationName="App Step 1"/>
</MissingApplicationMaskIds>
<MissingActivityApplications>
<MissingActivityApplication ActivityId="" ActivityName="step2"
ApplicationId="12346798134567891345678900010"/>
</MissingActivityApplications>
<InvalidApplicationMaskFieldIds>
<InvalidApplicationMaskFieldId MaskFieldId="123" ApplicationId=""
ApplicationName="AppTest"/>
</InvalidApplicationMaskFieldIds>
<MissingActivityVariables>
<MissingActivityVariable VariableId="12346798134567891345678900012">
<ReferencingActivities>
<ReferencingActivity Id="" Name="step2"/>
</ReferencingActivities>
</MissingActivityVariable>
<MissingActivityVariable VariableId="12346798134567891345678900013">
<ReferencingActivities>
<ReferencingActivity Id="" Name="step2"/>
</ReferencingActivities>
</MissingActivityVariable>
</MissingActivityVariables>
<InvalidParameterListCounts>
<InvalidParameterListCount ActivityId="" ActivityName="step1"
ApplicationId="" ApplicationName=" App Step1"/>
<InvalidParameterListCount ActivityId="" ActivityName="step3" ApplicationId=""
ApplicationName="App Step3"/>
</InvalidParameterListCounts>
<ParamWithoutVariableActivities>
<ParamWithoutVariableActivity ActivityId="" ActivityName="step1" ApplicationId=""
ApplicationName="App Step1"/>/>
<ParamWithoutVariableActivity ActivityId="" ActivityName="step2 ApplicationId=""
ApplicationName="App Step2"/>"/>
```

```
</ParamWithoutVariableActivities>
<MissingTakeOverActivities>
<MissingTakeOverActivity ActivityId="12346798134567891345678900014">
<ReferencingActivities>
<ReferencingActivity Id="step1" Name="step1"/>
</ReferencingActivities>
</MissingTakeOverActivity>
<MissingTakeOverActivity ActivityId="12346798134567891345678900015">
<ReferencingActivities>
<ReferencingActivity Id="step2" Name="step2"/>
</ReferencingActivities>
</MissingTakeOverActivity>
</MissingTakeOverActivities>
<MissingTakeOverVariables>
<MissingTakeOverVariable VariableId="12346798134567891345678900016">
<ReferencingActivities>
<ReferencingActivity Id="" Name="step1"/>
</ReferencingActivities>
</MissingTakeOverVariable>
</MissingTakeOverVariables>
<MissingFromActivities>
<MissingFromActivity FromActivityId="12346798134567891345678900017"
TransitionId="12346798134567891345678900018"/>
<MissingFromActivity FromActivityId="12346798134567891345678900019"
TransitionId="12346798134567891345678900020"/>
</MissingFromActivities>
<MissingToActivities>
<MissingToActivity ToActivityId="12346798134567891345678900021"
TransitionId="12346798134567891345678900022"/>
</MissingToActivities>
<RedundantTransitions>
<RedundantTransition FromActivityId="12346798134567891345678900021"
FromActivityName="stepX" ToActivityId="12346798134567891345678900021"
ToActivityName="stepY" TransitionId="12346798134567891345678900021"/>
</RedundantTransitions>
<InvalidFromLoopTransitions>
<InvalidFromLoopTransition
FromActivityId="1234679889134567890021" FromActivityName="stepX"
TransitionId="1234679891345678900021"/>
</InvalidFromLoopTransitions>
<InvalidToLoopTransitions>
<InvalidToLoopTransition ToActivityId="12346798134567891345678900021"
ToActivityName="stepX" TransitionId="12346798134567891345678900021"/>
</InvalidToLoopTransitions>
<InvalidFromOrToLoops>
<InvalidFromOrToLoop LoopActivityId="12346798134567891345678900021"
LoopActivityName="loopX"/>
</InvalidFromOrToLoops>
<CycleActivities>
<CycleActivity Id="12346798134567891345678900021" Name="step58"/>
<CycleActivity Id="12346798134567891345678900021" Name="step42"/>
</CycleActivities>
<InvalidLoopTransitions>
<InvalidLoopTransition TransitionId="12346798134567891345678900021"
FromActivityId="12346798134567891345678900021" FromActivityName="stepX"
ToActivityId="12346798134567891345678900021" ToActivityName="stepY"/>
</InvalidLoopTransitions>
<InvalidToLoopNumActivities>
<InvalidToLoopNumActivity Id="12367891345678900021" Name="step56"/>
<InvalidToLoopNumActivity Id="81345678913456789001" Name="step57"/>
</InvalidToLoopNumActivities>
<InvalidFromLoopNumActivities>
<InvalidFromLoopNumActivity Id="1234678900021" Name="step59"/>
<InvalidFromLoopNumActivity Id="3467981345678" Name="step60"/>
```

```
</InvalidFromLoopNumActivities>
<MissingLoopConditionActivities>
<MissingLoopConditionActivity Id="12891345678900021" Name="step61"/>
</MissingLoopConditionActivities>
<NoTerminationActivities>
<NoTerminationActivity Id="" Name="stepBla"/>
<NoTerminationActivity Id="" Name="stepFoo"/>
</NoTerminationActivities>
<NoLoopTerminationActivities>
<NoLoopTerminationActivity Id="" Name="stepLoop1"/>
<NoLoopTerminationActivity Id="" Name="stepLoop2"/>
</NoLoopTerminationActivities>
<MissingDefActIds>
<MissingDefActId Id="" Name="stepLoop1"/>
<MissingDefActId Id="" Name="stepLoop2"/>
</MissingDefActIds>
<AllClientTypesActivities>
<AllClientTypesActivitiy ActivityId="" ActivityName="XX" ApplicationId=""
ApplicationName="x1"/>
<AllClientTypesActivitiy ActivityId="" ActivityName="YY" ApplicationId=""
ApplicationName="y1"/>
</AllClientTypesActivities>
<AmbiguousActAppClientTypes>
<AmbiguousActAppClientType ActivityId="" ActivityName="ABC" ClientTypeId=""
ClientTypeName="CTX"/>
<AmbiguousActAppClientType ActivityId="" ActivityName="DEF" ClientTypeId=""
ClientTypeName="CTY"/>
</AmbiguousActAppClientTypes>
<ActAppInvalidClientTypes>
<ActAppInvalidClientType ActivityId="" ActivityName="AAB" ApplicationId=""
ApplicationName="v1"/>
<ActAppInvalidClientType ActivityId="" ActivityName="BBX" ApplicationId=""
ApplicationName="w2"/>
</ActAppInvalidClientTypes>
<ActEvtAllClientTypes>
<ActEvtAllClientType ActivityId="" ActivityName="ABC" EventTypeId="100"
EventTypeName="BeforeBlaEvent"/>
<ActEvtAllClientType ActivityId="" ActivityName="DEF" EventTypeId="100"
EventTypeName="AfterFuEvent"/>
</ActEvtAllClientTypes>
<AmbiguousActEvtClientTypes>
<AmbiguousActEvtClientType ActivityId="" ActivityName="ABCDE" EventTypeId="100"
EventTypeName="Before123Event" ClientTypeId="" ClientTypeName="CTX11"/>
<AmbiguousActEvtClientType ActivityId="" ActivityName="DEFGH" EventTypeId="100"
EventTypeName="After456Event" ClientTypeId="" ClientTypeName="CTY22"/>
</AmbiguousActEvtClientTypes>
<AmbiguousGlobalEvtClientTypes>
<AmbiguousGlobalEvtClientType ClientTypeId="" ClientTypeName="CT89"/>
<AmbiguousGlobalEvtClientType ClientTypeId="" ClientTypeName="CT90"/>
</AmbiguousGlobalEvtClientTypes>
<NoProcessStartClientType/>
<InvalidModParamVarIds>
<InvalidModParamVarId VarId="12891345678900021">
<InvalidModParamVarIds/>
<AdhocActWithoutDefaultSubActs>
<AdhocActWithoutDefaultSubAct Id="" Name="AdhocActWithoutDefaultSubAct 1"/>
<AdhocActWithoutDefaultSubAct Id="" Name="AdhocActWithoutDefaultSubAct 2"/>
</AdhocActWithoutDefaultSubActs>
<AdhocActWithUnknownDefaultSubActs>
<AdhocActWithUnknownDefaultSubAct Id="" Name="AdhocActWithUnknownDefaultSubAct 1"/>
<AdhocActWithUnknownDefaultSubAct Id="" Name="AdhocActWithUnknownDefaultSubAct 2"/>
</AdhocActWithUnknownDefaultSubActs>
<AdhocActDefaultActIsNotAdhocSubActs>
```

```
<AdhocActDefaultActIsNotAdhocSubAct Id="" Name="AdhocActDefaultActIsNotAdhocSubAct
1"/>
<AdhocActDefaultActIsNotAdhocSubAct Id="" Name="AdhocActDefaultActIsNotAdhocSubAct
2"/>
</AdhocActDefaultActIsNotAdhocSubActs>
<AdhocActDefaultActIsNotWorkitems>
<AdhocActDefaultActIsNotWorkitem Id="" Name="AdhocActDefaultActIsNotWorkitem 1"/>
<AdhocActDefaultActIsNotWorkitem Id="" Name="AdhocActDefaultActIsNotWorkitem 2"/>
</AdhocActDefaultActIsNotWorkitems>
</Errors>
```

Note:

Detailed description of Errors

- SchemaValidationError: schema validation error message
- MissingTypeDeclRecordMember: structure containing type declaration which contains a record without a member
  - TypeDeclId (STRING): ID of the type declaration
  - TypeDeclName (STRING): name of the type declaration
- MissingVariableRecordMember: structure containing a variable which contains a record without a member
  - VariableId (STRING): variable ID
  - VariableName (STRING): name of the variables
- MissingTypeDeclarations
  - MissingTypeDeclaration: structure that contains used but not defined type declaration
    - ID (STRING): ID used to reference missing type declaration
- ReferencingDataFields
  - ReferencingDataField: structure containing a DataField (workflow variable), which contains a non-defined type declaration
    - ID (STRING): ID of the data field
    - Name (STRING): name of the data field
- ReferencingTypeDeclarations
  - ReferencingTypeDeclaration: structure containing a type declaration, which uses an undefined type declaration
    - ID (STRING): ID of the type declaration
    - Name (STRING): name of the type declaration
- MissingActivityParticipants
  - MissingActivityParticipant: structure containing an activity participant who is not a workflow participant
    - ID (STRING): ID used to reference missing workflow participants
- ReferencingActivities
  - ReferencingActivity: structure containing an activity which references a parent structure
    - ID (STRING): Activity ID
    - Name (STRING): Activity name

- NoParticipantsInStartActivity: tag is available, if no participants have been assigned to the start activity
- MissingToolIds
  - MissingToolId: structure containing an activity which is missing the Tool ID
    - ActivityId (STRING): Activity type
    - ActivityName (STRING): Activity name
- MissingApplicationMasks
  - MissingApplicationMask: structure containing an application which references a non-existent mask
    - ApplicationId (STRING): application ID
    - ApplicationName (STRING): application name
    - MaskId (STRING): ID used to reference non-existent mask
- MissingApplicationMaskIds
  - MissingApplicationMaskId: structure containing an activity and an assigned application which has not been assigned a mask
    - ActivityId (STRING): Activity type
    - ActivityName (STRING): Activity name
    - ApplicationId (STRING): application ID
    - ApplicationName (STRING): Application name
- MissingActivityApplications
  - MissingActivityApplication: structure containing an activity which references a non-existent application
    - ActivityId (STRING): Activity type
    - ActivityName (STRING): Activity name
    - ApplicationId (STRING): ID used to reference a non-existent application
- InvalidApplicationMaskFieldId: structure contains invalid reference to mask fields
  - MaskFieldId (STRING): ID of the missing mask field
  - ApplicationId (STRING): application ID
  - ApplicationName (STRING): Application name
- MissingActivityVariables
  - MissingActivityVariable: structure containing an activity variable, which does not exist in the workflow as data field (workflow variable)
    - VariableId (STRING): ID used to reference a non-existent workflow variable
    - ReferencingActivityApplications
    - ReferencingActivityApplication: structure containing an activity application assignment for which the problem exists
    - ActivityId (STRING): Activity type
    - ActivityName (STRING): Activity name
    - ApplicationId (STRING): application ID

    - ApplicationName (STRING): Application name
- InvalidParameterListCounts
  - InvalidParameterListCount: structure containing an activity with an associated application with a disproportionate number of parameters
    - ActivityId (STRING): Activity type
    - ActivityName (STRING): Activity name
    - ApplicationId (STRING): application ID
    - ApplicationName (STRING): Application name
- ParamWithoutVariableActivities
  - ParamWithoutVariableActivity: structure containing an activity, which contains application parameters without a variable assignment
    - ActivityId (STRING): Activity type
    - ActivityName (STRING): Activity name
    - ApplicationId (STRING): application ID
    - ApplicationName (STRING): Application name
- MissingTakeOverActivities
  - MissingTakeOverActivity: structure containing an activity from which variables are to be taken over but where this activity does not exist in the workflow
    - ActivityId (STRING): ID which references a non-existent activity
- MissingTakeOverVariables
  - MissingTakeOverVariable: structure containing a variable to be transferred to an activity but this the variable (DataField) does not exist in the workflow
    - VariableId (STRING): ID used to reference a non-existent variable
- MissingFromActivities
  - MissingFromActivity: structure containing a transition that uses a From Activity which is not part of the workflow
    - FromActivityId (STRING): ID which references a non-existent activity
    - TransitionId (STRING): ID of the transition
- MissingToActivities
  - MissingToActivity: structure containing a transition that uses a To Activity which is not part of the workflow
    - ToActivityId (STRING): ID which references a non-existing activity
    - TransitionId (STRING): ID of the transition
- RedundantTransitions
  - RedundantTransition: structure containing a transition that is "too much" because this transition already exists with another ID
    - FromActivityId (STRING): ID of the FromActivity
    - FromActivityName (STRING): From activity name
    - ToActivityId (STRING): ID of the To activity

    - ToActivityName (STRING): To activity name
    - TransitionId (STRING): ID of the transition
- InvalidFromLoopTransitions
  - InvalidFromLoopTransition: structure containing a transition of the 'FROM LOOP' type where the From activity is not a loop activity
    - FromActivityId (STRING): ID of the FromActivity
    - FromActivityName (STRING): From activity name
    - TransitionId (STRING): ID of the transition
- InvalidToLoopTransitions
  - InvalidToLoopTransition: structure containing a transition of the 'TO LOOP' type where the To activity is not a loop activity
    - ToActivityId (STRING): ID of the To activity
    - ToActivityName (STRING): To activity name
    - TransitionId (STRING): ID of the transition
- InvalidFromOrToLoops
  - InvalidFromOrToLoop: structure containing a loop activity with an invalid combination of FROM and TO LOOP transitions
    - LoopActivityId (STRING): ID of the loop activity
    - LoopActivityName: (STRING) name of the loop activity
- CycleActivities
  - CycleActivity: structure containing an activity in which a forbidden cycle merges in the a workflow graph structure
    - ID (STRING): Activity ID
    - Name (STRING): Activity name
- InvalidLoopTransitions
  - InvalidLoopTransition: structure containing a transition which leads from one loop to another, i.e. an activity located in several partial graphs (loops)
    - FromActivityId (STRING): ID of the FromActivity
    - FromActivityName (STRING): From activity name
    - ToActivityId (STRING): ID of the To activity
    - ToActivityName (STRING): To activity name
    - TransitionId (STRING): ID of the transition
- InvalidToLoopNumActivities
  - InvalidToLoopNumActivity: structure containing a loop activity that does not have precisely one transition of the 'TO LOOP' type
    - ID (STRING): ID of the loop activity
    - Name (STRING): name of the loop activity
- InvalidFromLoopNumActivities

    - InvalidFromLoopNumActivity: Structure containing a loop activity that does not have precisely one transition of the 'FROM LOOP' type
        - ID (STRING): ID of the loop activity
        - Name (STRING): name of the loop activity
- MissingLoopConditionActivities
    - MissingLoopConditionActivity: structure containing a loop activity for which no condition is defined
        - ID (STRING): ID of the loop activity
        - Name (STRING): name of the loop activity
- NoTerminationActivities
    - NoTerminationActivity: structure containing an activity from which the termination activity cannot be reached
        - ID (STRING): Activity ID
        - Name (STRING): Activity name
- NoLoopTerminationActivities
    - NoLoopTerminationActivity: structure containing an activity within a loop without any path leading away, which leads back to the loop activity via a TOLOOP transition
        - ID (STRING): Activity ID
        - Name (STRING): Activity name
- MissingDefActIds
    - MissingDefActId: structure containing an activity, which has not been assigned a default activity for variable application
        - ID (STRING): Activity ID
        - Name (STRING): Activity name
- AllClientTypesActivities
    - AllClientTypesActivity: structure containing an activity for which the client application assignment is ambiguous (an assignment exists for all clients)
        - ActivityId (STRING): Activity type
        - ActivityName (STRING): Activity name
        - ApplicationId (STRING): application ID
        - ApplicationName (STRING): Application name
- AmbiguousActAppClientTypes
    - AmbiguousActAppClientType: structure containing an activity for which the client application assignment is ambiguous
        - ActivityId (STRING): Activity type
        - ActivityName (STRING): Activity name
        - ClientTypeId (STRING): ID of the client type
        - ClientTypeName (STRING): ClientType name
- ActAppInvalidClientTypes

  - ActAppInvalidClientType: structure containing an activity that contains an application assignment for an invalid client
    - ActivityId (STRING): Activity type
    - ActivityName (STRING): Activity name
    - ApplicationId (STRING): application ID
    - ApplicationName (STRING): Application name
- ActEvtAllClientTypes
  - ActEvtAllClientType: structure containing an activity for which the client event assignment is ambiguous (an assignment exists for all clients)
    - ActivityId (STRING): Activity type
    - ActivityName (STRING): Activity name
    - EventTypeId (STRING): EventType ID
    - EventTypeName (STRING): EventType name
- AmbiguousActEvtClientTypes
  - AmbiguousActEvtClientType: structure containing an activity for which the client event assignment is ambiguous
    - ActivityId (STRING): Activity type
    - ActivityName (STRING): Activity name
    - EventTypeId (STRING): ID of the client type
    - EventTypeName (STRING): ClientType name
    - ClientTypeId (STRING): ID of the client type
    - ClientTypeName (STRING): ClientType name
- AmbiguousGlobalEvtClientTypes
  - AmbiguousGlobalEvtClientType: structure containing a client type for which the assignment to the global client event is ambiguous
    - ClientTypeId (STRING): ID of the client type
    - ClientTypeName (STRING): ClientType name
- AdhocActWithoutDefaultSubActs
- AdhocActWithoutDefaultSubAct: structure containing ad hoc activities to which no default activity has been assigned
- ID (STRING): ID of the AdhocActivity
    - Name (STRING): adhoc activity name
- AdhocActWithUnknownDefaultSubActs
- AdhocActWithUnknownDefaultSubAct: structure containing ad hoc activities whose default activity does not exist in the model
- ID (STRING): ID of the AdhocActivity
- Name (STRING): adhoc activity name
- AdhocActDefaultActIsNotAdhocSubActs

- AdhocActDefaultActIsNotAdhocSubAct: structure containing ad hoc activities whose default activity is an invalid routing list activity for the respective ad hoc activity
- ID (STRING): ID of the AdhocActivity
- Name (STRING): adhoc activity name
- AdhocActDefaultActIsNotWorkitems
- AdhocActDefaultActIsNotWorkitem: structure containing ad hoc activities whose default activity is not a process step
- ID (STRING): ID of the AdhocActivity
- Name (STRING): adhoc activity name

Example:

Structure of Warnings

```
<Warnings>
<MissingWFParticipants>
<MissingWFParticipant Id="1234670003" Name="Dilbert" IsResponsible="1"/>
<MissingWFParticipant Id="1234679814" Name="Dogbert" IsResponsible="0"
IsFileResponsible="0">
<ReferencingActivities>
<ReferencingActivity Id="" Name="step1"/>
<ReferencingActivity Id="" Name="step2"/>
</ReferencingActivities>
</MissingWFParticipant>
<MissingWFParticipant Id="813456780005" Name="Catbert" IsResponsible="1">
<ReferencingActivities>
<ReferencingActivity Id="" Name="step1"/>
</ReferencingActivities>
</MissingWFParticipant>
</MissingWFParticipants>
<WFParticipantsWithoutASUser>
<WFParticipantWithoutASUser Id="1234670003" Name="Dilbert"/>
<WFParticipantWithoutASUser Id="1234679814" Name="Dogbert">
</WFParticipantsWithoutASUser>
<InvalidResponsibleIds>
<InvalidResponsibleId Id="1234567..."/>
<InvalidResponsibleId Id="2345678..."/>
</InvalidResponsibleIds>
<UnconnectedActivities>
<UnconnectedActivity Id="12346798134567891345678900021" Name="step7"/>
<UnconnectedActivity Id="12346798134567891345678900021" Name="step9"/>
</UnconnectedActivities>
<InvalidDefActIds>
<InvalidDefActId Id="" Name="stepLoop3">
<ValidDefaultActivities>
<ValidDefaultActivity Id="" Name="bla"/>
</ValidDefaultActivities>
</InvalidDefActId>
</InvalidDefActIds>
<NoOutVarsInLoopActConditions>
<NoOutVarsInLoopActCondition Id="1234679813454567890002l" Name="loopX"/>
<NoOutVarsInLoopActCondition Id="12346798134567891345671" Name="loopY"/>
</NoOutVarsInLoopActConditions>
<NoOutVarsInLoopActs>
<NoOutVarsInLoopAct Id="12346798134567891345678900021" Name="loopXA"/>
<NoOutVarsInLoopAct Id="12346798134567891345678900021" Name="loopYB"/>
</NoOutVarsInLoopActs>
<UnknownActAppClientTypes>
```

```
<UnknownActAppClientType ActivityId="" ActivityName="AAA"
ClientTypeId="1234679813456789134567890000A"/>
<UnknownActAppClientType ActivityId="" ActivityName="BBB"
ClientTypeId="1234679813456789134567890000B"/>
</UnknownActAppClientTypes>
<UnknownActEvtClientTypes>
<UnknownActEvtClientType ActivityId="" ActivityName="ABCDE" EventTypeId="100"
EventTypeName="Before123Event" ClientTypeId="ABC.."/>
<UnknownActEvtClientType ActivityId="" ActivityName="DEFGH" EventTypeId="100"
EventTypeName="After456Event" ClientTypeId="XYZ.."/>
</UnknownActEvtClientTypes>
<UnknownGlobalEvtClientTypes>
<UnknownGlobalEvtClientType ClientTypeId="123456...."/>
<UnknownGlobalEvtClientType ClientTypeId="1234567..."/>
</UnknownGlobalEvtClientTypes>
<AppToolsWithoutClientType>
<AppToolWithoutClientType ActivityId="" ActivityName="Activity 42" ApplicationId=""
ApplicationName="Application 43"/>
<AppToolWithoutClientType ActivityId="" ActivityName="Activity 44" ApplicationId=""
ApplicationName="Application 45"/>
<AppToolWithoutClientType ActivityId="" ActivityName="Activity 46" ApplicationId=""
ApplicationName="Application 47"/>
</AppToolsWithoutClientType>
<NoOrInvalidFileResponsibleId/>
<MasksWithoutFields>
<MaskWithoutFields MaskId="1234567" MaskName="Mask1"/>
<MasksWithoutFields/>
</Warnings>
```

Note:

Detailed description of Warnings

- MissingWFParticipants
  - MissingWFParticipant: structure containing a workflow participant who does not exist in the organization
    - ID (STRING): ID of the participant
    - Name (STRING): Name of the participant
    - IsResponsible (INT): indicates whether the participant is the person in charge of the process (yes = 1, no = 0)
    - IsFileResponsible (INT): indicates whether the participant is in charge of the files (yes = 1, no = 0)
- ReferencingActivities
  - ReferencingActivity: structure containing an activity to which users are assigned who do not exist in the organization
    - ID (STRING): Activity ID
    - Name (STRING): Activity name
- WFParticipantsWithoutASUser
  - WFParticipantWithoutASUser: structure containing a user who is assigned to no or a non-existent AS user
    - ID (STRING): User ID
    - Name (STRING): User name
- InvalidResponsibleIds

    - InvalidResponsibleId: structure containing a workflow process owner who does not exist in the list of participants
        - ID (STRING): process owner ID
- UnconnectedActivities
    - UnconnectedActivity: structure containing an activity which cannot be reached from the StartActivity
        - ID (STRING): Activity ID
        - Name (STRING): Activity name
- InvalidDefActIds: structure containing activities to which an invalid default activity has been assigned for variable application
        - ID (STRING): Activity ID
        - Name (STRING): Activity name
- ValidDefaultActivities: structure containing activities that represent valid default activities for the parent InvalidDefActId
        - ID (STRING): Activity ID
        - Name (STRING): Activity name
- NoOutVarsInLoopActConditions: structure with loop activities for which no variable application has been defined
        - ID (STRING): Activity ID
        - Name (STRING): Activity name
- UnknownActAppClientTypes: structure with activities for which an application assignment exists for an unknown client
        - ActivityId (STRING): Activity type
        - ActivityName (STRING): Activity name
        - ClientTypeId (STRING): ID of the client type
- UnknownActEvtClientTypes: structure with activities for which an event assignment exists for an unknown client
        - ActivityId (STRING): Activity type
        - ActivityName (STRING): Activity name
        - EventTypeId (STRING): EventType ID
        - EventTypeName (STRING): EventType name
        - ClientTypeId (STRING): ID of the client type
- UnknownGlobalEvtClientTypes: structure with unknown client types created for a global client event
        - ClientTypeId (STRING): ID of the client type
- AppToolsWithoutClientType: structure with activities for which an application assignment exists without client specification
        - ActivityId (STRING): Activity type
        - ActivityName (STRING): Activity name

  - § ApplicationId (STRING): application ID
  - § ApplicationName (STRING): Application name
- § MasksWithoutFields: structure with masks for which no mask fields exist
  - § MaskId (STRING): ID of the mask
  - § MaskName (STRING): mask name

Workflow Process and Process Step

wfm.CancelWorkItem

Description:

This job depersonalizes a process step. Afterwards a process step can again be seen by all participants and it can again be assigned to a person.

Parameter:

UserId (STRING): ID of the user to date

WorkItemId (STRING): instance ID of the activity

OrganizationId (STRING): Organization ID

ClientTypeId (String): ID of the used client type

Return:

(INT): 0 = job successful, otherwise error code

wfm.CompleteWorkItem

Description:

This job passes the editing data (variables, file) of a process step to the server and forwards the process step corresponding to the 'ActionType' parameter.

Parameter:

UserId (STRING): User ID

WorkItemId (STRING): instance of the activity

Parameters (BASE64): XML list of workflow variables

ActionType (STRING): flag that indicates what is to happen with the activity

§ SEND_BUTTON: Forward work item

§ STOREONLY: changes are saved but the process step is not forwarded

SendTo (STRING): is no longer supported -> an empty string is passed

File (BASE64): contains documents of the workflow file in XML format

DocsDeleted (STRING): comma-separated list of document IDs, which are to be deleted from the WF file

ClientTypeId (String): ID of the used client type

RoutingList (BASE64): routing list. This parameter is optional.

Return:

(INT): 0 = job successful, otherwise error code

Example:

Structure of Parameters

```
<Parameters>
<Parameter Name="WF_EDITOR_1" DataField=
"9FC5D03089E843F7B2D64F1CC2421418"><![CData[Schulze]]></Parameter>

</Parameters>
```

Note:

Detailed description of Parameters

§ Parameter: Workflow variable

  § DataField (STRING): parameter ID

  § Name (STRING): parameter name

  § CDATA: data specifying the content of the parameter

Example:

Structure of File

```
<File>
<Docs>
  <Doc Id="45" Type="23" Location="1" Workspace="0" New="1" Deleteable="1"
Moveable="2" UseActiveVariant="1"/>
</Docs>
</File>
```

Note:

Detailed description of File

- Docs: list of parameters (Doc) with the following structure
  - Id(STRING): ID of the document
  - Type (LONG): Document type
  - Location (INT): indicates whether the document is located in the SDREL (location = '1', SDREL is the database table Root-Document Relation) or in the system tray (location = '2')
  - Workspace (INT): indicates whether the object is in the info area (0) or in the workspace (1)
  - New (INT): indicates whether the object was newly added to the file (New = '1')
  - Deleteable (INT): indicates whether it is allowed to delete the document from the file (0 = no, 1 = yes)
  - Moveable (INT): indicates whether the document can be moved in the file (0 = no, 1 = yes)
  - UseActiveVariant (INT): indicates whether the active variant is to be used for this document object (0 = no, 1= yes)

Example:

Structure of RoutingList

```
<RoutingList Id="3294B433BFF6454D9C861B86B5A8AD5D"
ProcessId="BA16C21BB96D46D099E72070BCB644CC"
ActivityId="3294B433BFF6454D9C861B86B5A8AD5D" Expandable="1">
<Entries>
<Entry Nr="203" Expandable="1">
<Item Id="99825B18A8334987935684FDA3D6A40D"
ActivityId="6EE4490A48164A0FA6DC34A80099AF66" ActivityName="Create invoice"
ModelActivityName="Create invoice" Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
</Entry>
<Entry Nr="253" Expandable="1">
<Item Id="E15594D692C14FDA9AFDE8FA0A43F6E4"
ActivityId="6EE4490A48164A0FA6DC34A80099AF67" ActivityName="Approve invoice BL"
ModelActivityName="Approve invoice"  Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
<Item Id="C6DA9503CD874D69A9B703D0E06A52E8"
ActivityId="6EE4490A48164A0FA6DC34A80099AF67" ActivityName="Approve invoice GF"
ModelActivityName="Approve invoice" Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
</Entry>
</Entries>
</RoutingList>
```

Note:

Detailed description of RoutingList

- RoutingList: routing list with the following structure (or subsets of it)
- ID (String): routing list ID. The value is set by the server and must not be changed.
- ProcessId (String): Process ID
- ActivityId (String): activity ID
- Expandable (Int): 0: routing list cannot be expanded, 1: routing list can be expanded

- Entries: the structure combines entries of the routing list. An entry consists of multiple elements which can be executed simultaneously.
- Entry: describes an entry in the routing list.
- No (Int): for relative sorting of entries within the routing list. The absolute values do not have any influence on the client.
- Expandable (Int): 0: entry cannot be expanded, 1: entry can be expanded
- Item: describes an element of the routing list. This can be an activity, an executing person or a deadline.
- ID (STRING): for identification This ID must not be changed and must be identically sent for all jobs. If an item was created by the client, the ID must be stated here.
- ActivityId (String): ID of the activity in the workflow model
- ActivityName (String): activity name (does not necessarily have to match the name in the workflow model).
- ActivityModelName (String): ID of the activity in the workflow model
- TimerId(String): ID of a reminder time
- TimerDuration(Int): timer duration
- TimerDurationType(Int): 0: no period, 1: relative, 2: absolute
- Changeable(Int): 0: no change possible, 1:The element can be changed by the client.
- Deleteable(Int): 0: deletion not allowed, 1: element can be deleted
- Remark (String): note on editing (Text)
- ObjectsIds (String): list of editors' GUIDS (roles or persons), separated by comma

wfm.CreateProcessInstance

Description:

This job creates a process of the specified workflow family if the specified user has the right to execute a workflow. The created process instance can then be started using the wfm.StartProcess job.

Parameter:

UserId (STRING): User ID

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow family

ClientTypeId (STRING): ID of the used client type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ProcessId (STRING): ID of the newly created process

See also:

wfm.GetWorkflowList, wfm.StartProcess

wfm.GetActivityPerformers

Description:

This job returns all users/roles which are assigned to the activity as participants

Parameter:

RActivityId (STRING): instance ID of the activity

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ObjectIds (BASE64): XML list of roles/users

OrganizationId (STRING): ID of the organization to which the roles/users belong

Example:

Structure of the Object Ids

```
<Objects>
<Object Id="E95EF24F6C6AE1A40F" Absent="0" Substitute="0" Flag="1"/>
<Object Id="8FDD6BCB06CE478699" Absent="0" Substitute="0" Flag="1"/>
<Object Id="A7286EA0463F382057" Absent="0" Substitute="0" Flag="1"/>
</Objects>
```

Note:

Detailed description of ObjectIds

Object: structure with the following contents

§ ID (STRING): object ID

§ Absent (LONG): this flag indicates whether the person is absent = 1 or present = 0

§ Substitute (LONG): this flag indicates whether the person sees the activity in substitution = 1

§ Flag (LONG):

  § 1 = user is directly assigned to the process step (not through a role assignment)

  § 2 = the process step goes to the user's inbox

  § 4 = process step is personalized by this user

See also:

wfm.SetActivityPerformers

wfm.GetProcessFile

Description:

This job supplies the records for a process.

Parameter:

ProcessId (String): Process ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

File (BASE64): The file to the process in XML format

Example:

Structure of File

```
<File>
<Docs>
  <Doc Id="45" Type="23" Location="1" Workspace="0" New="1" Deleteable="1"
Moveable="2" UseActiveVariant="0" OriginalId="42" Display="1"  />
</Docs>
</File>
```

Note:

Detailed description of Workspace

- Docs: list of parameters (Doc) with the following structure
    - ID (STRING): ID of the document
    - Type (LONG): Document type
    - Location (INT): indicates whether the document is located in the SDREL (location = '1', SDREL is the database table Root-Document Relation) or in the system tray (location = '2')
    - Workspace (INT): indicates whether the object is in the info area (0) or in the workspace (1)
    - New (INT): indicates whether the object was newly added to the file (New = '1')
    - Deleteable (INT): indicates whether it is allowed to delete the document from the file (0 = no, 1 = yes)
    - Moveable (INT): indicates whether the document can be moved in the file (0 = no, 1 = yes)
    - UseActiveVariant (INT): indicates whether the active variant is to be used for the object (0 = no, 1= yes)
    - OriginalId (INT): indicates which document was originally dragged into the file (which ID this document had)
    - Display (INT): indicates whether this document is to be displayed in the preview (0 = no, 1= yes)

wfm.GetProcessList

Description:

This job returns a list of processes, the status of which can be specified.

Parameter:

Flags (INT): combined flag indicating the possible status of the processes to be queried

- 1 = CSPROCESS_INIT
- 2 = CSPROCESS_RUNNING
- 4 = CSPROCESS_SUSPENDED
- 8 = CSPROCESS_ACTIVE
- 16 = CSPROCESS_TERMINATED
- 32 = CSPROCESS_COMPLETED

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Processes (BASE64): information on the requested processes in XML format

Example:

Structure of processes

```
<Processes>
<Process>
<Id>9E813BD6D4054B6ABD9385A804FEA398</Id>
<Name>test 231</Name>
<Subject>Test (€42)</Subject>
<State>2</State>
<CreatorId>8FDD6BCB06CE467FAE8885E81F078699</CreatorId>
<CreationTime>1077888972</CreationTime>
</Process>
</Processes>
```

Note:

Detailed description of Process

- § ID (STRING): Process ID
- § Name (STRING): Process name
- § State (INT): Process status
- § CreatorId (STRING): ID of the process creator
- § CreationTime (INT): process creation time

wfm.GetProcessListByObject

Description:

This job returns a list of processes which contain a specified object in their files.

Parameter:

- § OrganizationId (STRING): Organization ID
- § ObjectId (INT): ID of the object
- § UserId (STRING, optional): user ID in the organizational structure
- § AllProcesses(INT, optional, default =0): If 1, then completed processes are determined as well. Otherwise only running processes are returned.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Processes (BASE64): information on the requested processes in XML format

Example:

Structure of processes

```
<Processes>
<Process
Id ="9E813BD6D4054B6ABD9385A804FEA398"
Name="test 231"
Subject ="Test (€42)
State="2"
```

```
CreatorId="8FDD6BCB06CE467FAE8885E81F078699"
CreationTime="1077888972"
      ProcessResponsible="0"
</Process>
</Processes>
```

Note:

Detailed description of Process

§ ID (STRING): Process ID

§ Name (STRING): Process name

§ State (INT): Process status

§ CreatorId (STRING): ID of the process creator

§ CreationTime (INT): process creation time

§ ProcessResponsible (INT): 1 = the querying user is the process owner for this process; 0 = otherwise

wfm.GetProcessProtocol

Description:

This job returns the log file for a process.

Parameter:

ProcessId (STRING): Process ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

File list: name and path of the log file

wfm.GetProcessResponsibles

Description:

This job returns the supervisor for the specified process.

Parameter:

ProcessId (STRING): Process ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ProcessId (STRING): ID of the organization of the process

Responsibles (STRING): comma-separated list of IDs of process supervisors

See also:

wfm.GetProcessResponsibles

wfm.GetRunningActivities

Description:

This job returns all activities which have to be performed for a specified user.

Parameter:

OrganizationId (STRING): Organization ID

UserId (STRING): ID of the enaio® user name

ClientTypeId (String): ID of the used client type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

RunningActivities (BASE64): data list of all the user's running activities in XML format

Example:

Structure of RunningActivities

```
RunningActivity
<RunningActivity>
<Activity Id="" RActivityId="" Name="" State="" ClosureTime="" OverTime=""
ReminderTime="" CanCancel =""/>
<User Name=""/>
<Process Id="" Name="" WorkflowId="" CreationTime="" IconId="" ObjectId=""/>
<Columns>
<Column DisplayName="" Value="" Position="">
<Column DisplayName="" Value="" Position="">
</Columns>
</RunningActivity>
<RunningActivity>
<Activity Id="" RActivityId="" Name="" State="" ClosureTime="" OverTime=""
ReminderTime="" CanCancel =""/>
<User Name=""/>
<Process Id="" Name="" WorkflowId="" Subject="" CreationTime="" IconId=""/>
<Columns/>
</RunningActivity>
</RunningActivities>
```

Note:

Detailed description of the return value RunningActivities

- Activity: structure describes a running activity
  - ID (STRING): ID of the activity in the model
  - RActivityId (STRING): instance ID of the activity
  - Name (STRING): Activity name
  - State (INT): Activity status
  - ClosureTime (INT): closure time indicates how long the activity remains closed
  - OverTime (INT): flag indicates whether the activity should already have been finished (1)
  - ReminderTime (INT): reminder time indicating when the activity should be finished
  - CanCancel (INT): no longer supported -> always 0
- User:

  - Name (STRING): name of the personalized user
- Process: describes the associated process
  - ID (STRING): Process ID
  - Name (STRING): Process name
  - WorkflowId (STRING): workflow ID
  - Subject (STRING): Process subject
  - CreationTime (INT): process creation time
  - IconId (INT): icon ID of the workflow model
  - ObjectId(String): ID of the document to be displayed by the clients in the preview.
- Columns: list of elements of the 'Column' type
  - Column: used to display workflow variables
  - DisplayName (STRING): the variable will be displayed under this name
  - Value: variable value
  - Position (INT): dictates the order of elements

wfm.GetWorkItemList

Description:

This job returns a list of all process steps for which the indicated user is configured as participant and which have not been personalized by another participant.

Parameter:

OrganizationId (STRING): Organization ID

UserId (STRING): User ID

ClientTypeId (String): ID of the used client type

Flags (INT): the queried process steps can be narrowed down with flags.

- 1 = INITIATED
- 2 = RUNNING
- 4 = SUSPENDED
- 16 = TERMINATED
- 32 = COMPLETED
- 64 = INUSE

Return:

(INT): 0 = job successful, otherwise error code

Return values:

WorkItems (BASE64): list with requested process steps in XML format

Example:

Structure of WorkItems

```
<WorkItems>
<WorkItem Id="" State="" Personalized="" ProcessId="" ProcessName=""
ActivityId="" ActivityName="" WarningTime="" OverTime=""
CreationTime="" WorkflowId="" Substitute="" IconId="" WorkflowType="2"
WorkflowVersion="5" ObjecId="32">
<Columns>
<Column DisplayName="" Value="" Position="">
<Column DisplayName="" Value="" Position="">
</Columns>
</WorkItem>
<WorkItem Id="" State="" Personalized="" ProcessId="" ProcessName=""
ProcessSubject="" ActivityId="" ActivityName="" WarningTime=""
OverTime="" CreationTime="" WorkflowId="" Substitute="" IconId="" WorkflowType="1"
WorkflowVersion="42" ObjecId="52">
<Columns>
<Column DisplayName="" Value="" Position="">
<Column DisplayName="" Value="" Position="">
</Columns>
</WorkItem>
</WorkItems>
```

Note:

Detailed description of WorkItems

- WorkItem: describes a process step
  - ID (STRING): ID of the process step
  - State (INT): status of the process step
    - 1 = INITIATED
    - 2 = RUNNING
    - 4 = SUSPENDED
    - 16 = TERMINATED
    - 32 = COMPLETED
    - 64 = INUSE
  - Personalized (STRING): name of the user who has personalized this step
  - ProcessId (STRING): Process ID
  - ProcessName (STRING): Process name
  - ProcessSubject (STRING): process subject
  - ActivityId (STRING): instance ID of the activity
  - ActivityName (STRING): Activity name
  - WarningTime (INT): Reminder time
  - OverTime (INT): flag indicating whether the step should already have been finished (1)
  - CreationTime (INT): creation time of the activity
  - WorkflowId (STRING): Workflowid
  - IconId (INT): icon ID of the workflow model
  - Substitute (INT): 1 = user receives the process step as a substitute, otherwise 0
  - WorkflowType (INT): 1 = ProductionWorkflow, 2 = Adhoc Workflow
  - WorkflowVersion (INT): returns the version number of the workflow model

- § ObjectId (String): the ID of the document to be displayed by the clients in the preview.
- § Columns: list of elements of the 'Column' type
  - § Column: used to display workflow variables
  - § DisplayName (STRING): the variable will be displayed under this name
  - § Value: variable value
  - § Position (INT): dictates the order of elements

See also:

wfm.GetWorkItem, wfm.StartWorkItem, wfm.GetWorkItemParams

wfm.GetWorkItemParams

Description:

This job determines all parameters of a process step for the user who has personalized the process step. All workflow variables, parameters for the input mask, contents of the workflow file and additional parameters (e.g. a password has to be entered for forwarding) are returned. This job was replaced by wfm.GetWorkItem.

Parameter:

WorkItemId (STRING): instance ID of the activity

UserId (STRING): User ID

ClientTypeId (String): ID of the used client type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Parameters (BASE64): XML list of parameters for the data mask

ExtendedAttributes (BASE64): XML list with data for the parameters ('attributes')

File (BASE64): XML list with documents of the info area/workspace of the WF file

RoutingList (Base64): routing list. This parameter is optional.

Example:

Structure of Parameters

```
<Parameters>
<Parameter FormField="" DataField="" Name="" Mode="" Selection=""
InfoText="" ListType="" ><![CDATA[ ]]></Parameter>
<Parameter FormField="" DataField="" Name="" Mode="" Selection=""
InfoText="" ListType="" ><![CDATA[ ]]></Parameter>
</Parameters>
```

Note:

Detailed description of Parameters

- § Parameters: list of formal parameters with the following structure (or subsets of it)
  - § FormField (STRING): ID of the field on a form to which the workflow variable is assigned, if there is no assignment name of the workflow variable
  - § DataField (STRING): ID of the workflow variable

- Name (STRING): name of the workflow variable
- Mode (INT): mode of the workflow variable
  - 1 = input parameter
  - 2 = output parameter
  - 3 = input/output parameter
- Selection (STRING): selection type for workflow variables in list form (single or multi:x)
- InfoText (string): information text Infotext for workflow variables in list form
- ListType (STRING): list type
  - ProcessList
  - UserList
  - UserDefList
- CDATA: structure and data of the workflow variable

Example:

Structure of ExtendedAttributes

```
<ExtendedAttributes>
<ExtendedAttribute Name="MASKID" Value=""/>
<ExtendedAttribute Name="SEND_BUTTON" Value="0"/>
<ExtendedAttribute Name="SENDTO_BUTTON" Value="0"/>
<ExtendedAttribute Name="END_BUTTON" Value="0"/>
<ExtendedAttribute Name="SIGN_ACTIVITY" Value=""/>
<ExtendedAttribute Name="CHECK_PASSWORD" Value=""/>
</ExtendedAttributes>
```

Note:

Detailed description of ExtendedAttributes

- ExtendedAttributes: list of parameters ('attributes') with the following structure
  - Name (STRING): attribute name
    - MASKID: GUID of the workflow mask
    - SEND_BUTTON:
    - END_BUTTON:
    - SIGN_ACTIVITY: 1 = digital signature required, otherwise 0
    - CHECK_PASSWORD: 1 = a password must be entered for forwarding, otherwise 0
  - Value: attribute value

Example:

Structure of RoutingList

```
<RoutingList Id="3294B433BFF6454D9C861B86B5A8AD5D"
ProcessId="BA16C21BB96D46D099E72070BCB644CC"
ActivityId="3294B433BFF6454D9C861B86B5A8AD5D" Expandable="1">
<Entries>
<Entry Nr="203" Expandable="1">
<Item Id="99825B18A8334987935684FDA3D6A40D"
ActivityId="6EE4490A48164A0FA6DC34A80099AF66" ActivityName="Create invoice"
ModelActivityName="Create invoice" Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
```

```
</Item>
</Entry>
<Entry Nr="253" Expandable="1">
<Item Id="E15594D692C14FDA9AFDE8FA0A43F6E4"
ActivityId="6EE4490A48164A0FA6DC34A80099AF67" ActivityName="Approve invoice BL"
ModelActivityName="Approve invoice"  Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
<Item Id="C6DA9503CD874D69A9B703D0E06A52E8"
ActivityId="6EE4490A48164A0FA6DC34A80099AF67" ActivityName="Approve invoice GF"
ModelActivityName="Approve invoice" Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
</Entry>
</Entries>
</RoutingList>
```

- RoutingList: routing list with the following structure (or subsets of it)
- ID (String): routing list ID. The value is set by the server and must not be changed.
- ProcessId (String): Process ID
- ActivityId (String): activity ID
- Expandable (Int): 0: routing list cannot be expanded, 1: routing list can be expanded
  - Entries: the structure combines entries of the routing list. An entry consists of multiple elements which can be executed simultaneously.
  - Entry: describes an entry in the routing list.
  - No (Int): for relative sorting of entries within the routing list. The absolute values do not have any influence on the client.
  - Expandable (Int): 0: entry cannot be expanded, 1: entry can be expanded
  - Item: describes an element of the routing list. This can be an activity, an executing person or a deadline.
  - ID (STRING): for identification This ID must not be changed and must be identically sent for all jobs. If an item was created by the client, the ID must be stated here.
  - ActivityId (String): ID of the activity in the workflow model
  - ActivityName (String): activity name (does not necessarily have to match the name in the workflow model).
  - ActivityModelName (String): ID of the activity in the workflow model
  - TimerId(String): ID of a reminder time
  - TimerDuration(Int): timer duration
  - TimerDurationType(Int): 0: no period, 1: relative, 2: absolute
  - Changeable(Int): 0: no change possible, 1:The element can be changed by the client.
  - Deleteable(Int): 0: deletion not allowed, 1: element can be deleted
  - Remark (String): note on editing (Text)
  - ObjectsIds (String): list of editors' GUIDS (roles or persons), separated by comma

Example:

Structure of File

```
<File>
<Docs>
  <Doc Id="45" Type="23" Location="1" Workspace="0" New="1" Deleteable="1"
Moveable="2" UseActiveVariant="0" OriginalId="42" Display="1"  />
</Docs>
</File>
```

Note:

Detailed description of Workspace

- Docs: list of parameters (Doc) with the following structure
  - ID (STRING): ID of the document
  - Type (LONG): Document type
  - Location (INT): indicates whether the document is located in the SDREL (location = '1', SDREL is the database table Root-Document Relation) or in the system tray (location = '2')
  - Workspace (INT): indicates whether the object is in the info area (0) or in the workspace (1)
  - New (INT): indicates whether the object was newly added to the file (New = '1')
  - Deleteable (INT): indicates whether it is allowed to delete the document from the file (0 = no, 1 = yes)
  - Moveable (INT): indicates whether the document can be moved in the file (0 = no, 1 = yes)
  - UseActiveVariant (INT): indicates whether the active variant is to be used for the object (0 = no, 1= yes)
  - OriginalId (INT): indicates which document was originally dragged into the file (which ID this document had)
  - Display (INT): indicates whether this document is to be displayed in the preview (0 = no, 1= yes)
  -

wfm.SetActivityPerformers

Description:

This job sets the participants for an activity. Roles and users can be chosen as participants. Note that old settings will be overwritten.

Parameter:

OrganizationId (STRING): ID of the organization to which the roles/users belong

UserId (STRING): ID of the executing user

RActivityId (STRING): instance ID of the activity

ObjectIds (String): GUIDs of the roles/users, separated by commas

Return:

(INT): 0 = job successful, otherwise error code

See also:

wfm.GetActivityPerformers

wfm.SetProcessResponsibles

Description:

This job sets the persons who are responsible for a process.

Parameter:

ProcessId (STRING): Process ID

OrganizationId (STRING): Organization ID

Responsibles (STRING): comma-separated list of IDs of process supervisors

Return:

(INT): 0 = job successful, otherwise error code

wfm.StartProcess

Description:

This job starts a workflow process. It is verified whether the specified user is authorized to start the process. The start activity of the process is executed. In order to be able to use this job, a process instance has to be created using the job wfm.CreateProcessInstance. Documents passed to the process are always located in the workspace of the workflow file.

Parameter:

UserId (STRING): User ID

ProcessId (STRING): Process ID

Workspace (BASE64): contains documents in XML format

DataFields (BASE64): contains the structure and values of the input variables in XML format

Return:

(INT): 0 = job successful, otherwise error code

Example:

Workspace structure:

```
<Workspace>
<Docs>
<Doc Id ="" Type="" Location="" Moveable="" Deleteable="" Workspace=""/>
<Doc Id ="" Type="" Location="" Moveable="" Deleteable="" Workspace=""/>
</Docs>
</Workspace>
```

Note:

Detailed description of Workspace

- Doc structure which encapsulates information for a document
  - Id (INT): ID of the document
  - Type (INT): Document type
  - Location (INT): indicates whether the document is located in the SDREL (location = '1', SDREL is the database table Root-Document Relation) or in the system tray (location = '2')
  - Moveable (INT): indicates whether the document can be moved from the info to the workspace (and the other way round), moveable = 1, otherwise 0

- § Deletable (INT): indicates whether the document can be deleted from the file (deletable = 1), otherwise 0
- § `Workspace` (INT): indicates whether the object has to be in the info area (0) or in the workspace (1)

Example:

Structure of DataFields

```
<DataFields>
<DataField Id="iDiscount">
<![CDATA[<WFVar><String>0</String></WFVar>]]>
</DataField>
<DataField Id="iDiscountable">
<![CDATA[<WFVar><String>0</String></WFVar>]]>
</DataField>
<DataField Id="lPositions">
<![CDATA[
<List TypeId="920C3899284B424EACBF881EE3A714C0">
<ListItem Id="00000000000000000000000000000001" Selection="0">
<Record>
<Member Name="iPosition"><STRING>1</STRING></Member>
<Member Name="strName"><STRING>Tisch</STRING></Member>
</Record>
</ListItem>
</List>
]]>
</DataField>
</DataFields>
```

Note:

Detailed description of DataFields

- § DataField: Workflow variable
  - § Id (STRING): name of the workflow variable
  - § CDATA: structure of the workflow variable

See also:

wfm.GetOrganizationObjects, wfm.CreateProcessInstance,

wfm.StartWorkItem

Description:

This job starts a process step. The process step is personalized for the indicated user. This job was replaced by wfm.GetWorkItem.

Parameter:

UserId (STRING): User ID

WorkItemId (STRING): instance ID of the activity

ClientTypeId (String): ID of the used client type

Return:

(INT): 0 = job successful, otherwise error code

See also:

wfm.GetWorkItemList, wfm.CancelWorkItem, wfm.CompleteWorkItem

wfm.GetWorkItem

Description:

This job starts a process step. The process step is personalized for the indicated user. Additionally all required data (form, file and workflow variables) will be returned for the client.

Parameter:

UserId (STRING): User ID

WorkItemId (STRING): instance ID of the activity

ClientTypeId (String): ID of the used client type

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ExtendedAttributes (BASE64): XML list with data for the parameters ('attributes')

Format

File (BASE64): XML list with documents of the info area/workspace of the WF file

Masks (BASE64): mask data in XML format

Parameters (BASE64): XML list of parameters for the data mask

RoutingList (Base64): routing list. This parameter is optional.

See also:

wfm.GetWorkItemList, wfm.CancelWorkItem, wfm.CompleteWorkItem

Example:

Structure of ExtendedAttributes

```
<ExtendedAttributes>
<ExtendedAttribute Name="MASKID" Value=""/>
<ExtendedAttribute Name="SEND_BUTTON" Value="0"/>
<ExtendedAttribute Name="SENDTO_BUTTON" Value="0"/>
<ExtendedAttribute Name="END_BUTTON" Value="0"/>
<ExtendedAttribute Name="SIGN_ACTIVITY" Value=""/>
<ExtendedAttribute Name="CHECK_PASSWORD" Value=""/>
</ExtendedAttributes>
```

Note:

Detailed description of ExtendedAttributes

- ExtendedAttributes: list of parameters ('attributes') with the following structure
  - Name (STRING): attribute name
    - MASKID: GUID of the workflow mask
    - SEND_BUTTON:
    - END_BUTTON:
    - SIGN_ACTIVITY: 1 = digital signature required, otherwise 0
    - CHECK_PASSWORD: 1 = a password must be entered for forwarding, otherwise 0

- Value: attribute value

Example:

Structure of File

```
<File>
<Docs>
<Doc Id ="" Type="" Rights="" Location="" Workspace="" Deleteable="0" Moveable ="1"
UseActiveVariant="" OriginalId="" Display=""/>
<Doc Id ="" Type="" Rights="" Location="" Workspace="" Deleteable="1" Moveable="1"
UseActiveVariant="" OriginalId="" Display=""/>
</Docs>
</File>
```

Note:

Detailed description of File

- File: encapsulates the parameters Workspace and Infospace
    - Docs: a list of document parameters ('Doc') with this structure (or a subset of it)
    - Id (INT): ID of the DMS document
    - Type (INT): Document type
    - Rights (INT): access rights

      0 = accessDenied
      1 = accessView
      2 = accessEdit
      4 = accessDelete
      8 = accessEditDataSheet
      15 = accessAll
    - Location (INT): location of the document (1: SDREL, 2: WF filing tray (document does not yet have a location))
    - Workspace (INT): indicates whether the object is in the info area (0) or in the workspace (1)
  - Deleteable (INT): indicates whether it is allowed to delete the document from the file (0 = no, 1 = yes)
  - Moveable (INT): indicates whether the document can be moved in the file (0 = no, 1 = yes)
  - UseActiveVariant (INT): indicates whether the active variant is to be used for the object (0 = no, 1= yes)
  - OriginalId (INT): indicates which document was originally dragged into the file (which ID this document had)
  - Display (INT): indicates whether this document is to be displayed in the preview (0 = no, 1= yes)

Example:

Structure of masks

```
<Masks>
<Mask Id="" Name="" Flags="" FrameWidth="" FrameHeight="">
<MaskField Id="" Name="" InternalName="" FieldName="" TabOrder=""
DataType="" InpLen="" Init="" Flags="" Flags1="" Flags2=""
InpLeft="" InpTop="" InpRight="" InpBottom="" FieldLeft=""
FieldTop="" FieldRight="" FieldBottom="" ToolTip="" ValuesId="">
<MaskFieldVal><![CDATA[ ]]></MaskFieldVal>
```

```
</MaskField>



<!Structure for Listcontrols -->
<MaskField Id="" Name="" InternalName="" TabOrder=""
DataType="" InpLen="" Init="" Flags="" Flags1="" Flags2=""
InpLeft="" InpTop="" InpRight="" InpBottom="" FieldLeft=""
FieldTop="" FieldRight="" FieldBottom="" ToolTip="" ValuesId="">
<MaskListCtrls>
<MaskListCtrl ColPos="" Name="" Type="" Length=""
ColWidth="" Color="" TextAlign="" ValuesId=""/>
<MaskListCtrlVal><![CDATA[ ]]></MaskListCtrlVal>
</MaskListCtrls>
</MaskField>


<!Structure for Pagecontrols -->
<MaskField Id="" Name="" InternalName="" TabOrder=""
DataType="" InpLen="" Init="" Flags="" Flags1="" Flags2=""
InpLeft="" InpTop="" InpRight="" InpBottom="" FieldLeft=""
FieldTop="" FieldRight="" FieldBottom="" ToolTip="" ValuesId="">
<Page Id="" Name="" Number="" IconId=""/>
<MaskFields>
<MaskField Id="" Name="" InternalName=""
TabOrder="" DataType="" InpLen="" Init="" Flags="" Flags1=""
Flags2="" InpLeft="" InpTop="" InpRight="" InpBottom=""
FieldLeft="" FieldTop="" FieldRight="" FieldBottom=""
ToolTip="" ValuesId=""/>
</MaskFields>
</Page>
</MaskField>
</Mask>
</Masks>
```

Note:

Detailed description of Masks

- Masks: list of masks, the elements of this list are of the 'Mask' type
  - Form structure which also contains a list of form fields of the 'MaskField' type
    - ID (STRING): ID of the mask
    - Name (STRING): mask name
    - Flags (INT): Flags
    - FrameWidth (INT): width of the mask
    - FrameHeight (INT): height of the mask
    - MaskField: structure containing the information about a mask field, including either the value of the mask field ('MaskfieldVal') or a list of form field controls ('MaskListCtrls'):
- ID (STRING): ID of the form field
- Name (STRING): name
- InternalName (STRING): internal name
- TabOrder (INT): tabulator order
- DataType (INT?): Data type
- InpLen (INT): Input length

- Init (STRING): initialization value
- Flags (INT): Flags
- Flags1 (INT): other flags
- Flags2 (INT): other flags
- InpLeft (INT): X position of the input field
- InpTop (INT): Y position of the input field
- InpRight (INT): width of the input field
- InpBottom (INT): height of the input field
- FieldLeft (INT): X of the field label
- FieldTop (INT): Y of the field label
- FieldRight (INT): width of the field label in pixels
- FieldBottom (INT): height of the field label in pixels
- ToolTip (INT): Tooltip
- ValuesId (INT): reference to list fields
- MaskFieldVal: form field value as CDATA
- MaskListCtrl: structure containing information and data for a form field control
- ColPos (INT): column position
- Name (STRING): name
- Type (STRING): Type
- Length (INT): Length
- ColWidth (INT): column width
- Color (INT): Color
- TextAlign (INT): text alignment
- ValuesId (STRING): reference to list fields
- MaskListCtrlVal: form field control value as CDATA
- Page: Structure containing the information about a page control (then again contains MaskFields)
- Id (STRING): pagecontrol ID
- Name (STRING): Pagecontrol name
- Number (INT): indicates the position ('page number') of a page
- IconId (INT): ID of the icon (from the DB table Osicons) which will be displayed on the pagecontrol

Example:

Structure of Parameters

```
<Parameters>
<Parameter FormField="" DataField="" Name="" Mode="" Selection=""
InfoText="" ListType="" ><![CDATA[ ]]></Parameter>
<Parameter FormField="" DataField="" Name="" Mode="" Selection=""
InfoText="" ListType="" ><![CDATA[ ]]></Parameter>
</Parameters>
```

Note:

Detailed description of Parameters

- Parameters: list of formal parameters with the following structure (or subsets of it)
  - FormField (STRING): ID of the field on a form to which the workflow variable is assigned, if there is no assignment name of the workflow variable
  - DataField (STRING): ID of the workflow variable
  - Name (STRING): name of the workflow variable
  - Mode (INT): mode of the workflow variable
    - 1 = input parameter
    - 2 = output parameter
    - 3 = input/output parameter
  - Selection (STRING): selection type for workflow variables in list form (single or multi:x)
  - InfoText (string): information text Infotext for workflow variables in list form
  - ListType (STRING): list type
    - ProcessList
    - UserList
    - UserDefList
  - CDATA: structure and data of the workflow variable

Example:

Structure of RoutingList

```
<RoutingList Id="3294B433BFF6454D9C861B86B5A8AD5D"
ProcessId="BA16C21BB96D46D099E72070BCB644CC"
ActivityId="3294B433BFF6454D9C861B86B5A8AD5D" Expandable="1">
<Entries>
<Entry Nr="203" Expandable="1">
<Item Id="99825B18A8334987935684FDA3D6A40D"
ActivityId="6EE4490A48164A0FA6DC34A80099AF66" ActivityName="Create invoice"
ModelActivityName="Create invoice" Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
</Entry>
<Entry Nr="253" Expandable="1">
<Item Id="E15594D692C14FDA9AFDE8FA0A43F6E4"
ActivityId="6EE4490A48164A0FA6DC34A80099AF67" ActivityName="Approve invoice BL"
ModelActivityName="Approve invoice"  Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
<Item Id="C6DA9503CD874D69A9B703D0E06A52E8"
ActivityId="6EE4490A48164A0FA6DC34A80099AF67" ActivityName="Approve invoice GF"
ModelActivityName="Approve invoice" Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
</Entry>
</Entries>
</RoutingList>
```

Warning, will be extended!!!

- RoutingList: routing list with the following structure (or subsets of it)
- ID (String): routing list ID. The value is set by the server and must not be changed.
- ProcessId (String): Process ID
- ActivityId (String): activity ID
- Expandable (Int): 0: routing list cannot be expanded, 1: routing list can be expanded
  - Entries: the structure combines entries of the routing list. An entry consists of multiple elements which can be executed simultaneously.
  - Entry: describes an entry in the routing list.
  - No (Int): for relative sorting of entries within the routing list. The absolute values do not have any influence on the client.
  - Expandable (Int): 0: entry cannot be expanded, 1: entry can be expanded
  - Item: describes an element of the routing list. This can be an activity, an executing person or a deadline.
  - ID (STRING): for identification This ID must not be changed and must be identically sent for all jobs. If an item was created by the client, the ID must be stated here.
  - ActivityId (String): ID of the activity in the workflow model
  - ActivityName (String): activity name (does not necessarily have to match the name in the workflow model).
  - ActivityModelName (String): ID of the activity in the workflow model
  - TimerId(String): ID of a reminder time
  - TimerDuration(Int): timer duration
  - TimerDurationType(Int): 0: no period, 1: relative, 2: absolute
  - Changeable(Int): 0: no change possible, 1:The element can be changed by the client.
  - Deleteable(Int): 0: deletion not allowed, 1: element can be deleted
  - Remark (String): note on editing (Text)
  - ObjectsIds (String): list of editors' GUIDS (roles or persons), separated by comma

Workflow Form, Event and Script

- wfm.DeleteEvent
- wfm.DeleteMasks
- wfm.DeleteScript
- wfm.GetEvents
- wfm.GetEventTypes
- wfm.GetGlobalScripts
- wfm.LoadMasks
- wfm.LoadScript
- wfm.SaveEvent
- wfm.SaveMasks
- wfm.SaveScript

§ wfm.SetEventScriptRelation

wfm.DeleteEvent

Description:

This job deletes one or more workflow events of a specific workflow model.

Parameter:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow model

EventIds (STRING): comma-separated GUID list of events to be deleted

ClientTypeId (String): ID of the used client type

Return:

(INT): 0 = job successful, otherwise error code

See also:

wfm.GetEvents

wfm.DeleteMasks

Description:

This job deletes one or more workflow forms of a specific workflow model.

Parameter:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow model

MaskIds (STRING): comma-separated GUID list of the masks to be deleted

Return:

(INT): 0 = job successful, otherwise error code

Return values:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow model

MaskIds (STRING): comma-separated GUID list of the masks that could not be deleted

wfm.DeleteScript

Description:

This job deletes a script belonging to a workflow event.

Parameter:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow

ScriptId (STRING): ID of the script

Return:

(INT): 0 = job successful, otherwise error code

wfm.GetEvents

Description:

This job returns a list of all set up events for an activity or the entire workflow model.

Parameter:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow model

ActivityId (STRING): ID of the activity. If the parameter is left blank, all events of the workflow model are returned.

ClientTypeId (String): ID of used client type. If the parameter is left blank, all defined client type events are returned.

EventTypeGroups (INT): combinable flag, indicates which events are requested

- 1 – all Client events are returned
- 2 – the global Client script is returned
- 4 – all server events are returned
- 8 – the global Client script is returned
- 15 – all events are returned

Code (INT): if code = 0 no script code is transferred, if code = 1 a script code is transferred

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Events (BASE64): contains the requested event list in XML format

Example:

Structure of events

```
<Events>
<Event>
<Id></Id>
<EventType></EventType>
  <ActivityId></ActivityId>
<Params></Params>
<Desription></Description>
  <ClientTypeId></ClientTypeId>
<Script Id="" Name="" Time="" Description="" ScriptLanguage="1">Script
</Script>
</Event>
</Events>
```

Note:

Detailed description of Events:

- Event: contains further elements and the 'Script' structure
  - ID (STRING): event ID
  - EventType (LONG): event type

    - 1 = BeforeForward
    - 2 = AfterForward
    - 3 = BeforeForwardTo
    - 4 = ButtonClick
    - 5 = BeforeOpen
    - 6 = AfterSignature
    - 7 = BeforeCancel
    - 8 = SimulateMaskEdit
    - 10000 = StartActivity
    - 10001 = EndActivity
    - 10003 = PersonalizeWorkItem
    - 10004 = GetWorkItemParams
    - 1000000 = global server script
    - 1000001= global client script
- Params: ID of the button for the 'ButtonClick' event type
- Description (STRING): not currently supported
- ActivityId (STRING): ID of the activity for which the event was created
- ClientTypeId (STRING): ID of the client type for which the event was created
- Script: structure containing the script:
    - ID (STRING): ID of the script
    - Name (STRING): script name
    - Time (LONG): creation time of the script
    - Description (STRING): not currently supported
    - ScriptLanguage (LONG): script language (1 = VB script, 2 = J script)
    - CDATA: data (contains the script code)

See also:

wfm.SaveEvent, wfm.DeleteEvent

wfm.GetEventTypes

Description:

This job returns all available event types.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

EventTypes (BASE64): contains all event types in XML format

Example:

Structure of event types

```
<EventTypes>
<EventType Id="1" Name="BeforeForward"/>
<EventType Id="2" Name="AfterForward"/>
<EventType Id="3" Name="BeforeForwardTo"/>
<EventType Id="4" Name="ButtonClick"/>
<EventType Id="5" Name="BeforeOpen"/>
<EventType Id="6" Name="AfterSignature"/>
<EventType Id="7" Name="BeforeCancel"/>
<EventType Id="8" Name="SimulateMaskEdit"/>
<EventType Id="10000" Name="StartActivity"/>
<EventType Id="10001" Name="EndActivity"/>
<EventType Id="10002" Name="BeforeStartSubProc"/>
<EventType Id="10003" Name="PersonalizeWorkItem"/>
<EventType Id="10004" Name="GetWorkItemParams"/>
        <EventType Id="10005" Name="CancelWorkItem"/>
</EventTypes>
```

Note:

Detailed description of EventTypes

- EvenType: structure characterizing an event type
  - ID (STRING): EventType ID
  - Name (STRING): EventType name

wfm.GetGlobalScripts

Description:

This job returns the global scripts for a workflow model.

Parameter:

WorkflowId (STRING): ID of the workflow model

OrganizationId (STRING): ID of the organization

Code (LONG): 0 = no script code is returned, 1 = script code is returned

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Scripts (BASE64): contains the global scripts in XML format

Example:

Structure of scripts

```
<Scripts>
<Script Id ="" Type="2"><![CDATA[...]]></Script>
<Script Id ="" Type="3"><![CDATA[...]]></Script>
</Scripts>
```

Note:

Detailed description of Script

- Script: structure characterizing a global script
  - Id (STRING): script ID
  - Type (LONG): indicates whether it is a server script (2) or a client script (3)

wfm.LoadMasks

Description:

This job returns all specified forms or all forms for a workflow model with substructure (fields, ListCtrlCols, catalogs).

Parameter:

OrganizationId (STRING): ID of the organization where the workflow and the masks are located

WorkflowId (STRING): ID of the workflow incl. masks

MaskIds (STRING): IDs of the requested masks (comma-separated); blank= all masks of the workflow model are loaded

Return:

(INT): 0 = job successful, otherwise error code

Return values:

OrganizationId (STRING): ID of the organization where the workflow and the masks are located

WorkflowId (STRING): ID of the workflow incl. masks

Masks (BASE64): mask data in XML format

Example:

Structure of masks

```
<Masks>
<Mask Id="" Name="" Flags="" FrameWidth="" FrameHeight="">
<MaskField Id="" Name="" InternalName="" FieldName="" TabOrder=""
DataType="" InpLen="" Init="" Flags="" Flags1="" Flags2=""
InpLeft="" InpTop="" InpRight="" InpBottom="" FieldLeft=""
FieldTop="" FieldRight="" FieldBottom="" ToolTip="" ValuesId="">
<MaskFieldVal><![CDATA[ ]]></MaskFieldVal>
</MaskField>



<!Structure for Listcontrols -->
<MaskField Id="" Name="" InternalName="" TabOrder=""
DataType="" InpLen="" Init="" Flags="" Flags1="" Flags2=""
InpLeft="" InpTop="" InpRight="" InpBottom="" FieldLeft=""
FieldTop="" FieldRight="" FieldBottom="" ToolTip="" ValuesId="">
<MaskListCtrls>
<MaskListCtrl ColPos="" Name="" Type="" Length=""
ColWidth="" Color="" TextAlign="" ValuesId=""/>
<MaskListCtrlVal><![CDATA[ ]]></MaskListCtrlVal>
</MaskListCtrls>
</MaskField>


<!Structure for Pagecontrols -->
<MaskField Id="" Name="" InternalName="" TabOrder=""
DataType="" InpLen="" Init="" Flags="" Flags1="" Flags2=""
InpLeft="" InpTop="" InpRight="" InpBottom="" FieldLeft=""
FieldTop="" FieldRight="" FieldBottom="" ToolTip="" ValuesId="">
<Page Id="" Name="" Number="" IconId=""/>
<MaskFields>
<MaskField Id="" Name="" InternalName=""
TabOrder="" DataType="" InpLen="" Init="" Flags="" Flags1=""
Flags2="" InpLeft="" InpTop="" InpRight="" InpBottom=""
```

```
FieldLeft="" FieldTop="" FieldRight="" FieldBottom=""
ToolTip="" ValuesId=""/>
</MaskFields>
</Page>
</MaskField>
</Mask>
</Masks>
```

Note:

Detailed description of Masks

- Masks: list of masks, the elements of this list are of the 'Mask' type
  - Form structure which also contains a list of form fields of the 'MaskField' type
    - ID (STRING): ID of the mask
    - Name (STRING): mask name
    - Flags (INT): Flags
    - FrameWidth (INT): width of the mask
    - FrameHeight (INT): height of the mask
    - MaskField: structure containing the information about a mask field, including either the value of the mask field ('MaskfieldVal') or a list of form field controls ('MaskListCtrls'):
- ID (STRING): ID of the form field
- Name (STRING): name
- InternalName (STRING): internal name
- TabOrder (INT): tabulator order
- DataType (INT?): Data type
- InpLen (INT): Input length
- Init (STRING): initialization value
- Flags (INT): Flags
- Flags1 (INT): other flags
- Flags2 (INT): other flags
- InpLeft (INT): X position of the input field
- InpTop (INT): Y position of the input field
- InpRight (INT): width of the input field
- InpBottom (INT): height of the input field
- FieldLeft (INT): X of the field label
- FieldTop (INT): Y of the field label
- FieldRight (INT): width of the field label in pixels
- FieldBottom (INT): height of the field label in pixels
- ToolTip (INT): Tooltip
- ValuesId (INT): reference to list fields
- MaskFieldVal: form field value as CDATA
- MaskListCtrl: structure containing information and data for a form field control

- § ColPos (INT): column position
- § Name (STRING): name
- § Type (STRING): Type
- § Length (INT): Length
- § ColWidth (INT): column width
- § Color (INT): Color
- § TextAlign (INT): text alignment
- § ValuesId (STRING): reference to list fields
- § MaskListCtrlVal: form field control value as CDATA
- § Page: structure containing the information about a page control (then again contains MaskFields)
- § ID (STRING): pagecontrol ID
- § Name (STRING): Pagecontrol name
- § Number (INT): indicates the position ('page number') of a page
- § IconId (INT): ID of the icon (from the DB table Osicons) which will be displayed on the pagecontrol

See also:

wfm.SaveMasks

wfm.LoadScript

Description:

This job returns a script from the database.

Parameter:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow

ScriptId (STRING): ID of the script

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ScriptCode (STRING): the requested script code

See also:

wfm.SaveScript

wfm.SaveEvent

Description:

This job creates an event for an activity. If the parameter EventId is set, the values Params and Description are reset for an existing event.

Parameter:

EventId (STRING): ID of an event (must be empty if it is a new event)

WorkflowId (STRING): ID of the workflow model

ActivityId (STRING): activity ID

EventType (INT): event type

Params (STRING): the ID of the button is transferred here for the event type 'ButtonClick'

Description (STRING): event description

OrganizationId (STRING): organization ID

ClientTypeId (String): ID of the used client type

Return:

(INT): 0: job successful, otherwise error code

Return values:

EventId (STRING): event ID

Note:

Event types

§ 1 = BeforeForward

§ 2 = AfterForward

§ 3 = BeforeForwardTo

§ 4 = ButtonClick

§ 5 = BeforeOpen

§ 6 = AfterSignature

§ 7 = BeforeCancel

§ 8 = SimulateMaskEdit

§ 10000 = StartActivity

§ 10001 = EndActivity

§ 10002 = BeforeStartSubProc

§ 10003 = PersonalizeWorkItem

§ 10004 = GetWorkItemParams

See also:

wfm.GetEvents, wfm.DeleteEvent, wfm.SaveScript, wfm.SetEventScriptRelation

wfm.SaveMasks

Description:

This job saves changes or multiple forms including substructures (fields, ListCtrlCols, catalogs).

Parameter:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow model

Masks (BASE64): Structure containing the masks to be saved (XML format)

Return:

(INT): 0 = job successful, otherwise error code

Return values:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow model

MaskIds (STRING): comma-separated ID list of masks that could not be saved

Example:

Structure of masks

```
<Masks>
<Mask Id="" ModState="" />
<!--for 0=MODSTATE_UNMODIFIED, 3=MODSTATE_DELETED -->
<!--or-->
<Mask Id="" ModState="" Name="" Flags="" FrameWidth="" FrameHeight="">
<!--for 1=MODSTATE_CHANGED, 2=MODSTATE_NEW-->
<MaskField Id="" ModState="" />
<!--for 0=MODSTATE_UNMODIFIED, 3=MODSTATE_DELETED-->
<MaskField Id="" ModState="" Name="" InternalName="" FieldName=""
TabOrder="" DataType="" InpLen="" Init="" Flags="" Flags1=""
Flags2="" InpLeft="" InpTop="" InpRight="" InpBottom=""
FieldLeft="" FieldTop="" FieldRight="" FieldBottom="" ToolTip=""
ValuesId="">
<!--for 1=MODSTATE_CHANGED, 2=MODSTATE_NEW-->
<MaskFieldVal Id="" ModState="">
<![CDATA[  ]]>
</MaskFieldVal>
</MaskField>
</Mask>
</Masks>
```

wfm.SaveScript

Description:

This job saves a script or creates a new one (Action = 2).

Parameter:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow

ID (STRING): ID of the script

Name (STRING): script Name

Description (STRING): short description of the script

Action (INT): action to be executed

§ 1 = the existing script will be overwritten

§ 2 = a new script will be created

ScriptCode (STRING): script code

Type (INT): script type

§ 1 = event script

§ 2 = global server script

§ 3 = global client script

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ScriptId (STRING): ID of the script

See also:

wfm.SaveEvent, wfm.SetEventScriptRelation

wfm.SetEventScriptRelation

Description:

This job links an event to a script.

Parameter:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow

EventId (STRING): event ID

ScriptId (STRING): Script ID

Action (INT): action to be executed with the specified parameters

§ 1 = create link

§ 2 = delete link

Return:

(INT): 0 = job successful, otherwise error code

See also:

wfm.SaveScript, wfm.SaveEvent

Administration and History Administration

Administration

§ wfm.AdminDeleteStatisticReportConfigs

§ wfm.AdminDeleteStatisticReports

§ wfm.AdminDeleteProcesses

§ wfm.AdminGetActivityVariables

§ wfm.AdminGetLockInfo

§ wfm.AdminGetProcessActivities

§ wfm.AdminGetProcessList

§ wfm.AdminGetProcessListByRole

§ wfm.AdminGetProcessListByUser

§ wfm.AdminGetProcessLocks

§ wfm.AdminGetProcessReport

- Wfm.AdminGetStatisticReportConfigs
- wfm.AdminGetStatisticReportData
- wfm.AdminGetStatisticReports
- wfm.AdminGetRoleProcesses
- wfm.AdminGetUserProcesses
- wfm.AdminGetWorkerqueue
- wfm.AdminGetWorkflowList
- wfm.AdminReleaseLock
- wfm.AdminRequestStatisticReport
- wfm.AdminResumeActivity
- wfm.AdminResumeProcess
- wfm.AdminRollbackProcess
- wfm.AdminSaveActivityVariables
- wfm.AdminSaveStatisticReportConfig
- wfm.AdminSuspendActivity
- wfm.AdminSuspendProcess
- wfm.AdminTerminateActivity
- wfm.AdminTerminateProcess

wfm.AdminDeleteStatisticReportConfigs

Description:

This job deletes all specified report configurations.

Parameter:

OrganizationId (STRING): ID of the organization in which the report configurations are located

ConfigIds (STRING): comma-separated list of report configuration GUIDs to be deleted

Return:

(INT): 0 = job successful, otherwise error code

Return values:

none

- **See also:**
- Wfm.AdminGetStatisticReportConfigs, wfm.AdminSaveStatisticReportConfig

wfm.AdminDeleteStatisticReports

Description:

This job deletes all specified statistics reports.

Parameter:

ReportIds (STRING): comma-separated list of report GUIDs to be deleted

Return:

(INT): 0 = job successful, otherwise error code

Return values:

none

wfm.AdminDeleteProcesses

Description:

This job deletes all specified workflow processes. If documents which were created during the workflow process and not yet in the enaio® system are contained in the workflow file, they are inserted in the filing tray of the enaio® client of the job executor. Before a process can be deleted, it must be stopped using the job wfm.AdminSuspendProcess.

Parameter:

OrganizationId (STRING): GUID of the organization to which the processes belong

Processes (STRING): comma-separated list of process GUIDs, which will be deleted

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Processes (STRING): comma-separated list of process GUIDs which could not be deleted

See also:

wfm.AdminGetProcessList, wfm.GetOrganizations,

wfm.AdminSuspendProcess

wfm.AdminGetActivityVariables

Description:

This job returns the IDs, names, structure and values of all workflow variables for an activity in XML format.

Parameter:

RActivityId (STRING): instance ID of an activity

Return:

(INT): 0 = job successful, otherwise error code

Return values:

DataFields (BASE64): list with information on activity variables in XML format

Example:

Structure of DataFields

```
<DataFields>
<DataField Id="72FA13878B7C4744BB33C58B5AAFF5F0" Name="wfProtocol">
<![CDATA[
<WFVar>
<List TypeId="75FCEF515EDE4CF5A1BA8DE94FD26023"></List>
<Types>
<Type Id="75FCEF515EDE4CF5A1BA8DE94FD26023">
<Record>
<Member Name="date"><STRING/></Member>
<Member Name="time"><STRING/></Member>
<Member Name="activity"><STRING/></Member>
<Member Name="user"><STRING/></Member>
<Member Name="log"><STRING/></Member>
</Record>
</Type>
</Types>
</WFVar>
]]>
</DataField>
<DataField Id="4170579B168642B0E8A172BC73459" Name="Testvariable">
<![CDATA[
<WFVar>
<String>Here is a string.</String>
<Types></Types>
</WFVar>
]]>
</DataField>
</DataFields>
```

Note:

Detailed description of DataFields

§ DataField

  § ID (STRING): GUID of the variables

  § Name (STRING): name of the variables

  § CDATA: structure and values of variables

See also:

wfm.AdminGetProcessActivities, wfm.AdminSaveActivityVariables

wfm.AdminGetProcessActivities

Description:

This job returns all activities for a process.

Parameter:

ProcessId (STRING): GUID of the process

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Activities (BASE64): contains requested information in XML format

Example:

Structure of Activities

```
<Activities>
<Activity>
<Name></Name>
<Id></Id>
<RActivityId></RActivityId>
<CreationTime></CreationTime>
```

```
<Owner></Owner>
<OwnerId></OwnerId>
<AccessTime></AccessTime>
<EndTime></EndTime>
<ReminderTime></ReminderTime>
<ReminderState></ReminderState>
<State></State>
<ClosureTime></ClosureTime>
<WorkItem></WorkItem>
            <LoopCount></LoopCount>
            <ExecutionPoints></ExecutionPoints>
<RLoopId></RLoopId>
      <ActivityNo></ActivityNo>
</Activity>
</Activities>
```

Note:

Detailed description of Activities

- Activity: structure containing all information on the activity
  - Name (STRING): Activity name
  - Id (STRING): GUID of the activity (from the workflow model)
  - RActivityId (STRING): instance ID of the activity
  - CreationTime (LONG): creation time of the activity on the server
  - Owner (STRING): owner who has personalized the activity
  - OwnerId (STRING): owner ID
  - AccessTime (LONG): time when the activity was last accessed
  - EndTime (LONG): time when the activity ended
  - ReminderTime (LONG): Reminder time
  - ReminderState (LONG): reminder status (1 = reminder time exceeded, otherwise 0)
  - State (LONG): Activity status
    - 0x1 = the activity has been initialized
    - 0x2 = the activity has been started (e.g. variables have been created).
    - 0x4 = the start activity event has been executed
    - 0x8 = the end activity event has been executed
    - 0x10=only with loops: the loop condition has been checked.
    - 0x20=only with loops: the loop body is executed.
    - 0x40=only with process steps: the process step is provided in the inboxes.
    - 0x80=only with process steps: the process step is personalized.
    - 0x100 = waiting until a closure is expired
    - 0x400 = the activity has been executed, e.g. a process step has been forwarded or a loop has been fully executed.
    - 0x800 = the following activities have been calculated and possibly also been created
    - 0x1000 = the activity is completed, no follow-up activities have been initiated
    - 0x2000 = the activity has been stopped by a user

    - § 0x4000 = the activity is completed
    - § 0x8000=only with multi-instance activities: the activity has been created.
    - § 0x10000=only with ad hoc activities: the ad hoc activity has been created.
    - § 0x20000 = activity canceled
    - § 0x40000 = only with ad hoc activities: the ad hoc activity is executed.
    - § 0x10000000 = the activity has been stopped by the system due to an error
- § ClosureTime (LONG): closure time for the activity
- § WorkItem (LONG): 1 -> activity visible in inboxes, otherwise 0
- § LoopCount (LONG): if it is a loop activity, the loop count is indicated here, otherwise 0
- § ExecutionPoints (LONG): Execution points for job wfm.AdminRollbackProcess
    - § 100 = Activity is created
    - § 200 = Activity finished
- § RLoopId (STRING): instance ID of the surrounding loop, if none exists, the parameter is empty
- § ActivityNo (LONG): indicates at which position the activity in the process was created

See also:

wfm.AdminGetProcessList, wfm.AdminGetActivityVariables

wfm.AdminGetProcessList

Description:

This job returns a list of all active processes for a workflow model.

Parameter:

OrganizationId (STRING): ID of the organization

WorkflowId (STRING): ID of the workflow model

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Processes (BASE64): list of all running processes in XML format

Example:

Structure of processes

```
<Processes>
<Process Id="" Name="" Subject="" State="" SuspendedActivity="">
<Creation UserId="" UserName="" Time=""/>
<LastActivity ExecTime="" Name="" Id="" UserId="" UserName=""/>
</Process>
<Process Id="" Name="" Subject="" State="" SuspendedActivity="">
<Creation UserId="" UserName="" Time=""/>
<LastActivity ExecTime="" Name="" Id="" UserId="" UserName=""/>
</Process>
</Processes>
```

Note:

Detailed description of Processes

- ID (STRING): ID of the process
- Name (STRING): name of a process
- Subject (STRING): Process name
- State (LONG): state of a process
  - 1 = INIT (process has been initialized)
  - 2 = RUNNING (process is running)
  - 4 = SUSPENDED (process was stopped -> is not supported, yet)
  - 8 = ACTIVE (process is running and at least one activity is personalized)
  - 16 = TERMINATED (process was canceled -> is not supported, yet)
  - 32 = COMPLETED (process successfully completed)
  - 64 = SYSSUSPENDED (process was stopped by engine e.g. due to an error in the event script)
- SuspendedActivity (LONG): 1 – at least one activity of the process has been stopped, otherwise 0
- Creation: structure which encapsulates information on the creation of the respective process
  - UserId (STRING): user ID of the creator
  - UserName (STRING): user name of the creator
  - Time (LONG): Creation time
- LastActivity: structure which encapsulates information on the last use of the process
  - ExecTime (LONG): last execution time
  - Name (STRING): name of the last executed activity
  - ID (STRING): ID of the last executed activity
  - UserId (STRING): executor ID of the last activity
  - UserName (STRING): executor name of the last activity

See also:

wfm.AdminGetWorkflowList, wfm.GetOrganizations, wfm.AdminDeleteProcesses

wfm.AdminGetProcessListByRole

Description:

This job returns all processes for a role ID which currently have a corresponding process step in the inbox.

Parameter:

OrganizationId (STRING): ID of the organization

RoleId (STRING): ID of the role

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Processes (BASE64): list of all process steps in the inbox in XML format

Example:

Structure of processes

```
<Processes>
<Process Id="" Name="" Subject="" State="">
<Creation UserId="" UserName="" Time=""/>
<Activity CreationTime="" Name="" Id="" Owner="" OwnerId=""
ReminderTime="" OwnerTime="" WICreationTime="" State =""/>
</Process>
</Processes>
```

Note:

Detailed description of Processes

§ ID (STRING): Process ID

§ Name (STRING): Process name

§ Subject (STRING): Process subject

§ State (LONG): state of a process

  § 1 = INIT (process has been initialized)

  § 2 = RUNNING (process is running)

  § 4 = SUSPENDED (process was stopped -> is not supported, yet)

  § 8 = ACTIVE (process is running and at least one activity is personalized)

  § 16 = TERMINATED (process was canceled -> is not supported, yet)

  § 32 = COMPLETED (process successfully completed)

  § 64 = SYSSUSPENDED (process was stopped by engine e.g. due to an error in the event script)

§ Creation: structure which encapsulates information on the creation of the respective process

  § UserId (STRING): user ID of the creator

  § UserName (STRING): user name of the creator

  § Time (LONG): process creation time

§ Activity: structure which encapsulates information on the activity of the process

  § CreationTime (LONG): Creation time

  § Name (STRING): Activity name

  § ID (STRING): Activity ID

  § ReminderTime (LONG): reminder time (if 0, then no reminder time)

  § Owner (STRING): name of the user who has personalized the process step

  § OwnerId (STRING): ID of the user who has personalized the process step

  § OwnerTime (LONG): time when the process step was personalized

  § WICreationTime (LONG): time when the process step was created in the inbox

  § State (LONG): Activity status

    § 0x1 = the activity has been initialized

    § 0x2 = the activity has been started (e.g. variables have been created).

    § 0x4 = the start activity event has been executed

    § 0x8 = the end activity event has been executed

- 0x10=only with loops: the loop condition has been checked.
- 0x20=only with loops: the loop body is executed.
- 0x40=only with process steps: the process step is provided in the inboxes.
- 0x80=only with process steps: the process step is personalized.
- 0x100 = waiting until a closure is expired
- 0x400 = the activity has been executed, e.g. a process step has been forwarded or a loop has been fully executed.
- 0x800 = the following activities have been calculated and possibly also been created
- 0x1000 = the activity is completed, no follow-up activities have been initiated
- 0x2000 = the activity has been stopped by a user
- 0x4000 = the activity is completed
- 0x8000=only with multi-instance activities: the activity has been created.
- 0x10000=only with ad hoc activities: the ad hoc activity has been created.
- 0x20000 = activity canceled
- 0x40000 = only with ad hoc activities: the ad hoc activity is executed.
- 0x10000000 = the activity has been stopped by the system due to an error

See also:

wfm.GetOrganizations, wfm.AdminGetRoleProcesses, wfm.AdminGetProcessActivities, wfm.AdminDeleteProcesses

wfm.AdminGetProcessListByUser

Description:

This job returns all processes for a user ID which currently have a corresponding process step in the inbox.

Parameter:

OrganizationId (STRING): Organization ID

UserId (STRING): User ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Processes (BASE64): contains requested information in XML format

Example:

Structure of processes

```
<Processes>
<Process Id="" Name="" Subject="" State="">
<Creation UserId="" UserName="" Time=""/>
<Activity CreationTime="" Name="" Id="" ReminderTime=""
OwnerTime="" WICreationTime="" State=""/>
</Process>
</Processes>
```

Note:

Detailed description of Processes

- ID (STRING): Process ID
- Name (STRING): Process name
- Subject (STRING): Process subject
- State (LONG): state of a process
  - 1 = INIT (process has been initialized)
  - 2 = RUNNING (process is running)
  - 4 = SUSPENDED (process was stopped -> is not supported, yet)
  - 8 = ACTIVE (process is running and at least one activity is personalized)
  - 16 = TERMINATED (process was canceled -> is not supported, yet)
  - 32 = COMPLETED (process successfully completed)
  - 64 = SYSSUSPENDED (process was stopped by engine e.g. due to an error in the event script)

- Creation: structure which encapsulates information on the creation of the respective process
  - UserId (STRING): user ID of the creator
  - UserName (STRING): user name of the creator
  - Time (LONG): process creation time
- Activity: structure which encapsulates information on the activity of the process
  - CreationTime (LONG): Creation time
  - Name (STRING): Activity name
  - ID (STRING): Activity ID
  - ReminderTime (LONG): reminder time (if 0, then no reminder time)
  - Owner (STRING): name of the user who has personalized the process step
  - OwnerId (STRING): ID of the user who has personalized the process step
  - OwnerTime (LONG): time when the process step was personalized
  - WICreationTime (LONG): time when the process step was created in the inbox
  - State (LONG): Activity status
    - 0x1 = the activity has been initialized
    - 0x2 = the activity has been started (e.g. variables have been created).
    - 0x4 = the start activity event has been executed
    - 0x8 = the end activity event has been executed
    - 0x10=only with loops: the loop condition has been checked.
    - 0x20=only with loops: the loop body is executed.
    - 0x40=only with process steps: the process step is provided in the inboxes.
    - 0x80=only with process steps: the process step is personalized.
    - 0x100 = waiting until a closure is expired

  - 0x400 = the activity has been executed, e.g. a process step has been forwarded or a loop has been fully executed.
  - 0x800 = the following activities have been calculated and possibly also been created
  - 0x1000 = the activity is completed, no follow-up activities have been initiated
  - 0x2000 = the activity has been stopped by a user
  - 0x4000 = the activity is completed
  - 0x8000=only with multi-instance activities: the activity has been created.
  - 0x10000=only with ad hoc activities: the ad hoc activity has been created.
  - 0x20000 = activity canceled
  - 0x40000 = only with ad hoc activities: the ad hoc activity is executed.
  - 0x10000000 = the activity has been stopped by the system due to an error

See also:

wfm.GetOrganizations, wfm.AdminGetUserProcesses, wfm.AdminGetProcessActivities, wfm.AdminDeleteProcesses

wfm.AdminGetRoleProcesses

Description:

This job returns all roles of a specific organization for which process steps exist in inboxes.

Parameter:

OrganizationId (STRING): organization ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Roles (BASE64): list of all roles in XML format

Example:

Role structure

```
<Roles>
<Role Id="" Name="" ProcessCount=""/>
<Role Id="" Name="" ProcessCount=""/>
</Roles>
```

Note:

Detailed description of Roles

- Role: structure which merges information on roles and number of processes
  - ID (STRING): ID of the role
  - Name (STRING): role name
  - ProcessCount (LONG): number of processes for which process steps exist in inboxes

See also:

wfm.GetOrganizations

wfm.AdminGetUserProcesses

Description:

This job returns all users of a specific organization for which process steps exist in inboxes.

Parameter:

OrganizationId (STRING): organization ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Users (BASE64): list of all users in XML format

Example:

Structure of Users

```
<Users>
<User Id="" Name="" ProcessCount=""/>
<User Id="" Name="" ProcessCount=""/>
</Users>
```

Note:

Detailed description of users

- § User: structure which merges information on a user and the number of processes
  - § ID (STRING): User ID
  - § Name (STRING): User name
  - § ProcessCount (LONG): number of processes of the user for which process steps exist in the inbox

See also:

wfm.GetOrganizations

wfm.AdminGetWorkflowList

Description:

This job returns all workflow models which are used (status ACTIVE/INUSE) and the number of running processes in an organization.

Parameter:

OrganizationId (STRING): organization ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Workflows (BASE64): contains requested information in XML format

Example:

Structure of Workflows

```
<Workflows>
```

```
<Workflow Id="" Name="" ProcessCount=""/>
<Workflow Id="" Name="" ProcessCount=""/>
</Workflows>
```

Note:

Detailed description of Workflows

- Workflow: structure which encapsulates information for a workflow model
  - Id (STRING): ID of the workflow model
  - Name (STRING): name of the workflow model
  - ProcessCount (LONG): number of running processes for this workflow model

See also:

wfm.GetOrganizations, wfm.AdminGetProcessList

wfm.AdminRequestStatisticReport

description:

This job requires the creation of a new report for a given statistics report configuration.

Parameter:

OrganizationId (STRING): Organization ID

UserId (STRING): ID of the user who executes the action

ConfigId (STRING): ID of the statistics report configuration

CompileTime (INT): earliest time at which the report is to be generated

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ReportId (STRING): report ID

CreatorId (STRING): ID of the user who has requested the report

CreatorName (STRING): name of the person object of the user

CreationTime (INT): timestamp (when has be report been created), if state = 0 or 1 it is the time when the report has been requested, otherwise it is the time when it has been started to create the report

- State (INT): report status

0 – report is requested

- 1 – report is currently being generated
- 2 – report generated

wfm.AdminResumeActivity

Description:

This job releases the activity for processing after it has been stopped.

Parameter:

RActivityId (STRING): instance ID of the activity

UserId (STRING): ID of the user who executes the action

Return:

(INT): 0 = job successful, otherwise error code

Return values:

State (INT): Activity status after resume

See also:

wfm.AdminSuspendActivity

wfm.AdminResumeProcess

Description:

This job releases a process for processing after it has been stopped. If only certain activities have been stopped, they will also be continued.

Parameter:

ProcessId (STRING): ID of the process

UserId (STRING): ID of the user who executes the action

Return:

(INT): 0 = job successful, otherwise error code

Return values:

State (INT): Process status after resume

See also:

wfm.AdminSuspendProcess

wfm.AdminRollbackProcess

Description:

This job resets a process to the specified activity and returns all process activities, which will be deleted. With the 'ExecutionPoint', the reset point for the activity will be specified. It can be reset to the activity before it has been created or before it is ended. Before a process can be reset, it must be stopped with the job wfm.AdminSuspendProcess.

Parameter:

RActivityId (STRING): InstanceId of the activity to which the process is to be reset

ExecutionPoint (INT): start point in the activity

§ 100 = Activity is created

§ 200 = Activity finished

DoRollback (INT): 0 = reset is not done; 1 = process is reset (RunningActivities is filled in both options)

Return:

(INT): 0 = job successful, otherwise error code

Return values:

RunningActivities (BASE64): contains all activities that were deleted after reset (DoRollback = 1) or would be deleted (DoRollback = 0)

Example:

Structure of RunningActivities

```
RunningActivity
<RunningActivity Id="" RActivityId="" Name="" State="" CreationTime=""/>
<RunningActivity Id="" RActivityId="" Name="" State="" CreationTime=""/>
</RunningActivities>
```

Note:

Detailed description of RunningActivities

- RunningActivity
  - ID (STRING): Activity ID from the workflow model
  - RactivityId (STRING): instance ID of the activity
  - Name (STRING): Activity name
  - State (LONG): Activity status
  - CreationTime (LONG): creation time of the activity (timestamp)

See also:

wfm.AdminSuspendProcess

wfm.AdminSaveActivityVariables

Description:

This job saves workflow variables for a given instance ID of an activity in the database.

Parameter:

UserId (STRING): ID of the user who has performed the modification

RActivityId (STRING): instance ID of an activity

DataFields (BASE64): list with information on variables in XML format

Return:

(INT): 0 = job successful, otherwise error code

Example:

Structure of DataFields

```
<DataFields>
<DataField Id="" ><![CData[]]></DataField>
<DataField Id="" ><![CData[]]></DataField>
</DataFields>
```

Note:

Detailed description of DataFields

- DataField: structure which encapsulates information about a workflow variable

- ID (STRING): variable ID
- CData: variable value

See also:

wfm.AdminGetProcessActivities, wfm.AdminGetActivityVariables

wfm.AdminSaveReportConfig

Description:

This job saves ('insert' or 'update') a report configuration.

Parameter:

UserId (STRING): ID of the user who has performed the modification

ConfigId (STRING): configuration ID. If the ID is not known to the server, a new configuration will be created, if the ID exists already, the corresponding configuration will be changed. Only ConfigName, FamilyIDs and ConfigData are changed with this update.

ConfigName (STRING): configuration name

OrganizationId (STRING): ID of the organization in which the configuration is located

ConfigType (LONG): configuration type

0 – statistics processes

1 – process details

ConfigData (STRING): XML description of the configuration

Return:

(INT): 0 = job successful, otherwise error code

Return values:

SaveType (INT): indicates type of save

0 – insert new / Insert

1 – change existing / Update

- If SaveType = 0 other output parameters exist:
- CreationTime (INT): creation time

CreatorId (STRING): ID of the user who has made the last change

See also:

wfm.AdminDeleteStatisticReportConfigs, Wfm.AdminGetStatisticReportConfigs

wfm.AdminSuspendActivity

Description:

This job stops an activity. The activity cannot be processed.

Parameter:

RActivityId (STRING): instance ID of the activity

UserId (STRING): ID of the user who executes the action

Return:

(INT): 0 = job successful, otherwise error code

Return values:

State (LONG): Activity status before the pause

- 0x1 = the activity has been initialized
- 0x2 = the activity has been started (e.g. variables have been created).
- 0x4 = the start activity event has been executed
- 0x8 = the end activity event has been executed
- 0x10=only with loops: the loop condition has been checked.
- 0x20=only with loops: the loop body is executed.
- 0x40=only with process steps: the process step is provided in the inboxes.
- 0x80=only with process steps: the process step is personalized.
- 0x100 = waiting until a closure is expired
- 0x400 = the activity has been executed, e.g. a process step has been forwarded or a loop has been fully executed.
- 0x800 = the following activities have been calculated and possibly also been created
- 0x1000 = the activity is completed, no follow-up activities have been initiated
- 0x2000 = the activity has been stopped by a user
- 0x4000 = the activity is completed
- 0x8000=only with multi-instance activities: the activity has been created.
- 0x10000=only with ad hoc activities: the ad hoc activity has been created.
- 0x20000 = activity canceled
- 0x40000 = only with ad hoc activities: the ad hoc activity is executed.
- 0x10000000 = the activity has been stopped by the system due to an error

See also:

wfm.AdminResumeActivity

wfm.AdminSuspendProcess

Description:

This job stops a process. No activities of the process can be processed.

Parameter:

ProcessId (STRING): Process ID

UserId (STRING): ID of the user who executes the action

Return:

(INT): 0 = job successful, otherwise error code

Return values:

State (INT): process status before the stop

- 0 – neutral value
- 1 – process has been initialized
- 2 – process running
- 3 – process stopped
- 4 – process active
- 5 – process is terminated
- 6 – process finished
- 7 – the process was stopped by the system (an error occurred)

See also:

wfm.AdminResumeProcess

wfm.AdminTerminateActivity

Description:

This job cancels an activity. Until now only multi instances can be canceled. If this is the last instance of a multi-instance activity, the process will be resumed.

Parameter:

RActivityId (STRING): Activity ID

UserId (STRING): ID of the user who executes the action

Return:

none

wfm.AdminTerminateProcess

Description:

This job cancels a process. The process is deleted from the runtime tables of the workflow but remains in the history and is marked as canceled there.

Parameter:

ProcessId (STRING): Process ID

UserId (STRING): ID of the user who executes the action

Return:

none

wfm.AdminGetLockInfo

Description:

This job returns information on workflow database tables that were locked by the system.

Parameter:

OrganizationId (STRING): ID of the organization

Return:

(INT): 0 = job successful, otherwise error code

Return values:

LockItems (BASE64): information about workflow database tables

Example:

Structure of LockItems

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<LockItems>
<LockItem Id="" Name="" State="" LockTime="" ServerId="" ThreadId=""/>
<LockItem Id="" Name="" State="" LockTime="" ServerId="" ThreadId=""/>
</LockItems>
```

Note:

Detailed description of LockItems

§ Id (LONG): database table ID.

§ Name (STRING): database table name.

§ State (LONG): 1 = database table is locked, otherwise 0

§ LockTime (LONG): timestamp (when the table was locked)

§ ServerId (LONG): ID of the server that locked the table

§ ThreadId (LONG): ID of the thread that locked the table

See also:

wfm.AdminReleaseLock

wfm.AdminGetWorkerqueue

Description:

This job returns information about elements which are currently in the worker queue.

Parameter:

OrganizationId (STRING): ID of the organization

Return:

(INT): 0 = job successful, otherwise error code

Return values:

WorkerqueueItems (BASE64): information on worker queue elements in XML format

Example:

Structure of WorkerqueueItems

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<WorkerqueueItems>
<WorkerqueueItem Id="" ActivityName="" ProcessName="" State=""
TargetState="" CreationTime="" LockState="" LockTime="" ServerId=""/>
<WorkerqueueItem Id="" ActivityName="" ProcessName="" State=""
TargetState="" CreationTime="" LockState="" LockTime="" ServerId=""/>
</LockItems>
```

Note:

Detailed description of WorkerqueueItems

- ID (STRING): instance ID of the activity in the worker queue
- ActivityName (STRING): activity name
- ProcessName (STRING): name of the process to which the activity belongs
- State (LONG): current status of the activity
- TargetState (LONG): this status should be reached after processing by the Workerqueue
- CreationTime (LONG): timestamp (when element was included in the workerqueue)
- LockState (LONG): 1 – worker queue element is locked, otherwise 0
- LockTime (LONG): timestamp (when worker queue element was locked)
- ServerId (LONG): ID of the server that has locked worker queue element

See also:

wfm.AdminReleaseLock

wfm.AdminGetProcessLocks

Description:

This job returns information about locked workflow processes.

Parameter:

OrganizationId (STRING): ID of the organization

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Processes (BASE64): information on the locked workflow processes in XML format

Example:

Structure of processes

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<Processes>
<Process Id="" Name="" State="" LockTime="" ServerId="" ThreadId=""/>
<Process Id="" Name="" State="" LockTime="" ServerId="" ThreadId=""/>
</LockItems>
```

Note:

Detailed description of Processes

- ID (STRING): Process ID
- Name (STRING): Process name
- LockTime (LONG): timestamp (when has the process been locked)
- ServerId (LONG): ID of the server that has locked the process
- ThreadId (LONG): ID of the thread that has locked the process

See also:

wfm.AdminReleaseLock

wfm.AdminReleaseLock

Description:

This job unlocks the database of the specified type.

Parameter:

Locktype (LONG): indicates the lock type

- 1 – locked processes will be released (LockId must contain the corresponding process ID)
- 2 – locked worker queue element will be released (LockID must contain the corresponding worker queue item ID)
- 3 – locked database table will be unlocked (LockID then has to contain the corresponding table ID)

LockId (STRING): see Locktype

Return:

(INT): 0 = job successful, otherwise error code

See also:

wfm.AdminGetWorkerqueue, wfm.AdminGetProcessLocks,

wfm.AdminGetLockInfo

wfm.AdminGetProcessReport

Description:

This job provides information on the indicated workflow processes. Not to be confused with the statistic process reports!

Parameter:

Processes (STRING): comma-separated list of process IDs

File (LONG): 1 = information about the WF file is also returned, otherwise 0

GlobalDataFields (LONG): 1 = information about global variables is also returned, otherwise 0

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Processes (BASE64): information on workflow processes in XML format

Example:

Structure of processes

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Processes>
<Process>
<Id>B216B6ACC46B4A33AFB9D6852D928A33</Id>
<Name>Global Variables Test - Parallel Process 2</Name>
<CreationTime>1143128816</CreationTime>
<EndTime>0</EndTime>
<UserName>Peter Mustermann</UserName>
<DataFields>
<DataField>
<Id>79FBB5E2F38B434DBF6C7507614CF740</Id>
<Name>TextVar02</Name>
```

```
<Value><WFVar><String>Value 1</String><Types></Types></WFVar></Value>
</DataField>
</DataFields>
<Docs>
<Doc>
<Id>194</Id>
<Type>196608</Type>
</Doc>
</Docs>
</Process>
</Processes>
```

Note:

Detailed description of Processes

§ ID (STRING): Process ID

§ Name (STRING): Process name

§ CreationTime (LONG): timestamp (when the process was created)

§ EndTime (LONG): timestamp (when the process was completed)

§ UserName (STRING): process creator name

§ DataFields: information on global variables

 § ID (STRING): variable ID

 § Name (STRING): variable name

 § Value: structure and value of variable

§ Docs: objects of the workflow file

 § ID (STRING): ID of the ECM object

 § Type (LONG): Object type

wfm.AdminGetStatisticReportConfigs

Description:

This job returns configurations for process reports (statistics etc.)

Parameter:

OrganizationId (STRING): ID of the organization where the requested report configurations are located

CreatorId (STRING): ID of the configuration creator. Can be empty. If the ID has been set, only configurations with this CreatorID will be returned.

UserId (STRING): ID of the calling user. Only configurations to which the user has access rights will be returned.

ConfigTypes (STRING): comma-separated list of configuration types. Can be empty. If the parameter has been set, only configurations with the indicated types will be returned.

0 – process statistics

1 – process details

FamilyIds (STRING): comma-separated list with IDs of workflow families. Can be empty. If the parameter is not empty, only configurations will be returned which are at least assigned to one of the indicated families.

RespectRights (LONG): the flag indicates whether only report configurations are returned, whose reports the user can request and view.

0 – user receives report configuration independently of permissions

1 – user receives report configuration according to permissions

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ReportConfigs (BASE64): information about the requested report configurations in XML format

Example:

Structure of ReportConfigs

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<ReportConfigs>
<ReportConfig ConfigId="" ConfigName="" CreationTime="" CreatorId="" CreatorName=""
ConfigType="">
<ConfigData><![CDATA[]]</CondigData>
</ReportConfig>
<ReportConfig ConfigId="" ConfigName="" CreationTime="" CreatorId="" CreatorName=""
ConfigType="">
<ConfigData><![CDATA[]]</CondigData>
</ReportConfig>
</ReportConfigs>
```

Note:

Detailed description of ReportConfig

§ ConfigId (BSTR): configuration ID

§ ConfigName (STRING): configuration name

§ CreationTime (LONG): timestamp (when the configuration was created)

§ CreatorId (STRING): user ID of the configuration creator

§ CreatorName (STRING): name of the person object corresponding to the user

§ ConfigType (LONG): configuration type

0 – statistics processes

1 – process details

§ ConfigData CDATA section (XML): configuration as XML description

§ **See also:**

§ wfm.AdminDeleteStatisticReportConfigs, wfm.AdminSaveStatisticReportConfig

wfm.AdminGetStatisticReportData

Description:

This job returns data for an already generated statistics report.

Parameter:

ReportId (STRING): report ID

UserId (STRING): ID of the calling user. Only data to which the user has the appropriate rights will be returned.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

File list: name and path of the XML file with the data of the statistics report

wfm.AdminGetStatisticReports

description:

Returns the available reports for a statistics report configuration.

Parameter:

OrganizationId (STRING): ID of the organization

ConfigId (STRING): ID of the statistics report configuration

UserId (STRING): ID of the calling user. Only data to which the user has the appropriate rights will be returned.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Reports (BASE64): information on the requested reports in XML format

Example:

Report structure

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Reports>
<Report ReportId="" CreationTime="" CreatorId="" CreatorName="" State=""/>
<Report ReportId="" CreationTime="" CreatorId="" CreatorName="" State=""/>
</Reports>
```

Note:

Detailed description of Report

- ReportId (STRING): report ID
- CreationTime (LONG): timestamp (when has the report been created), if state = 0 or 1 it is the time when the report has been requested, otherwise it is the time when the creation has been started
- CreatorId (STRING): userID of the report creator
- CreatorName (STRING): name of the person object corresponding to the user
- State (LONG): report status

0 – report is requested

- 1 – report is currently being generated
- 2 – report generated

History Administration

wfm.GetHistActivitiesByProcess

Description:

This job determines all activities for a historic process.

Parameter:

ProcessId (STRING): history process ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Activities (BASE64): contains the requested activities in XML format

Example:

Structure of Activities

```
<Activities>
<Activity RActivityId="3A25AFD5CBD9B246" Name="Step1" EntryNo="10"/>
<Activity RActivityId="51BBA8B5AAA1D471" Name="StartActivity" EntryNo="5"/>
<Activity RActivityId="1A8E439C131AADF2B" Name="EndActivity" EntryNo="21"/>
</Activities>
```

Note:

Detailed description of Activity

§ Activity: structure characterizing an activity instance

  § RActivityId (STRING): Instance ID of the activity

  § Name (STRING): Activity name

  § EntryNo (LONG): reflects the chronological order

See also:

wfm.GetHistProcessList, wfm.GetHistEntries

wfm.GetHistEntries

Description:

This job returns all performed actions for a history activity or a history process. When calling a job it has to be guaranteed that only one of the two parameters is set.

Parameter:

ProcessId (STRING): hist. process ID

RActivityId (STRING): hist. instance ID of the activity

Return:

(INT): 0 = job successful, otherwise error code

Return values:

HistoryEntries (BASE64): contains the requested actions in XML format

Example:

Structure of HistoryEntries

```
<HistoryEntries>
<HistoryEntry HistId=".." EntryNr=".." ProcessId=".." RActivityId=".."
HistType=".." Time=".." OrganisationId=".." UserId=".." UserName=".."
ServerId=".." ServerName=".."/>
<HistoryEntry HistId=".." EntryNr=".." ProcessId=".." RActivityId=".."
HistType=".." Time=".." OrganisationId=".." UserId=".." UserName=".."
ServerId=".." ServerName=".."/>
</HistoryEntries>
```

Note:

Detailed description of HistoryEntries

§ HistId (STRING): ID of the hist. entry

§ EntryNo (LONG): entry number reflects the chronological order.

§ ProcessId (STRING): process ID

§ RActivityId (STRING): instance ID of the activity (only set when called via parameter RActivityID)

§ HistType (LONG): type of the hist. entry

- § 1 = PREPAREPROCESS
- § 2 = STARTPROCESS
- § 3 = ENDPROCESS
- § 4 = PREPAREACTIVITY
- § 5 = STARTACTIVITY
- § 6 = ENDACTIVITY
- § 7 = COPYACTIVITYVARIABLES
- § 8 = HALTACTIVITY
- § 9 = REACTIVATEACTIVITY
- § 10 = STARTWORKITEM
- § 11 = PERSONALIZED
- § 12 = DEPERSONALIZED

  - 13 = SAVEWORKITEM
  - 14 = ENDWORKITEM
  - 15 = CREATETIMER
  - 16 = CANCELTIMER
  - 17 = REMIND
  - 18 = DELAYED
  - 19 = STARTSCRIPT
  - 20 = ENDSCRIPT
  - 21 = SYSSUSPEND
  - 22 = SETACTIVITYPERFORMER
- Time (LONG)
- OrganisationId (STRING)
- UserId (STRING)
- UserName (STRING)
- ServerId (STRING)
- ServerName (STRING)

See also:

wfm.GetHistActivitiesByProcess, wfm.GetHistProcessList, wfm.GetHistVariablesByHistEntry

wfm.GetHistProcessList

Description:

This job determines all processes for a historic workflow model.

Parameter:

OrganizationId (STRING): ID of the organization

HistWorkflowId (STRING): history ID of the model

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Processes (BASE64): contains the requested processes in XML format

Example:

Structure of processes

```
<Processes>
<Process Id=".." Name=".." FinalSubject=".." UserName=".." CreationTime=".."
EndTime=".."/>
<Process Id=".." Name=".." FinalSubject=".." UserName=".." CreationTime=".."
EndTime=".."/>
</Processes>
```

Note:

Detailed description of Processes

- § ID (STRING): Process ID
- § Name (STRING): Process name
- § FinalSubject (STRING): final subject of the process
- § UserName (STRING): process creator name
- § CreationTime (STRING): creation time of the process
- § EndTime (STRING): end time of the process

See also:

wfm.GetHistWorkflowList, wfm.GetHistActivitiesByProcess, wfm.GetHistEntries, wfm.GetHistTimersByProcess

wfm.GetHistTimerEntries

Description:

This job returns all actions (historic entries) for a reminder/closure period.

Parameter:

TimerId (STRING): timer ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

TimerEntries (BASE64): contains the requested actions in XML format

Example:

structure of TimerEntries:

```
<TimerEntries>
<TimerEntry HistId=".." EntryNr=".." ProcessId=".." RActivityId=".."
HistType=".." Time=".." OrganisationId=".." UserId=".." UserName=".."
ServerId=".." ServerName=".."/>
</TimerEntries>
```

Note:

Detailed description of TimerEntries

- § HistId (STRING): hist. ID
- § EntryNo (LONG): entry number (reflects the chronological order)
- § ProcessId (STRING): process ID
- § RActivityId (STRING): instance ID of the activity
- § HistType(LONG): type of the hist. entry
  - § 15 = CREATETIMER
  - § 16 = CANCELTIMER
  - § 17 = REMIND
  - § 18 = DELAYED
- § Time (LONG): entry creation time
- § OrganizationId (STRING): Organization ID

§ UserId (STRING): User ID

§ UserName (STRING): name of the user who has personalized the activity

§ ServerId (STRING): server ID

§ ServerName (STRING): Name of the server

See also:

wfm.GetHistTimersByProcess

wfm.GetHistTimersByProcess

Description:

This job returns all reminder/closure periods for a historic process.

Parameter:

ProcessId (STRING): hist. process ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Timers (BASE64): contains the requested reminder/closure periods in XML format

Example:

Structure of Timers

```
<Timers>
<Timer TimerId=".." ProcessId=".." FromActivityId=".." FromActivityName=".."
ToActivityId=".." ToActivityName=".." TimerType=".." DestinationType=".."
DestinationTime=".." LoopType=".."/>
</Timers>
```

Note:

Detailed description of Timers

§ TimerId (STRING): ID of the reminder/closure period

§ ProcessId (STRING): process ID

§ FromActivityId (STRING): ID of the activity (the period is valid starting with this activity)

§ FromActivityName (STRING): name of the activity (the period is valid starting with this activity)

§ ToActivityId (STRING): ID of the activity (the period is valid until this activity)

§ ToActivityName (STRING): name of the activity (the period is valid until this activity)

§ TimerType (LONG): Period type

  § 0 = closure period

  § 1 = reminder time

§ DestinationType (LONG):

  § 0 = TIMER_REFERENCES_START_OF_ACTIVITY

  § 1 = TIMER_REFERENCES_END_OF_ACTIVITY

§ DestinationTime (LONG): indicates when the target has to be reached

§ LoopType (LONG): 0 is always returned

See also:

wfm.GetHistProcessList, wfm.GetHistTimerEntries

wfm.GetHistVariablesByHistEntry

Description:

This job returns the workflow variables for a history entry.

Parameter:

HistId (STRING): ID of the history entry

Return:

(INT): 0 = job successful, otherwise error code

Return values:

DataFields (BASE64): contains the requested history variables in XML format

Example:

Structure of DataFields

```
<DataFields>
<DataField Id="123.." Name="strTest">
<![CDATA[I am a string]]>
</DataField>
<DataField Id="" Name="">
<![CDATA[]]>
</DataField>
</DataFields>
```

Note:

Detailed description of DataFields

§ ID (STRING): variable ID

§ Name (STRING): name of the variables

§ CDATA: value and structure of the variables

See also:

wfm.GetHistEntries

wfm.GetHistWorkflowList

Description:

This job returns all workflow models in the history administration and the number of processes started by the workflow model.

Parameter:

OrganizationId (STRING): organization ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Workflows (BASE64): contains the requested actions in XML format

Example:

Structure of Workflows

```
<Workflows>
<Workflow Id="18268F5E1F532" HistWorkfowId="04DC0D15A62A4" Name="test"
ProcessCount="3" Version="70" FamilyId="0774DC0D15A62A4"
FamilyName="testfam"/>
<Workflow Id="39418268F5532" HistWorkfowId="0BFB9FF2D225E7" Name="test"
ProcessCount="1" Version="83" FamilyId="T0774DC0D15A62"
FamilyName="testfamily"/>
</Workflows>
```

Note:

Detailed description of Workflows

§ ID (STRING): ID of the workflow model

§ HistWorkfowId (STRING): history ID of the workflow model

§ Name (STRING): name of the workflow model

§ ProcessCount (LONG): indicates how often the workflow model was started

§ Version (LONG): version number of the workflow model

§ FamilyId (STRING): GUID of the associated workflow family

§ FamilyId Name(STRING): GUID of the associated workflow family

See also:

wfm.GetOrganizations, wfm.GetHistProcessList

wfm.GetHistWorkItemRelActivitiesByProcess

Description:

This job returns all activities which were put into an inbox for a historical process, i.e. all activities which were edited by a user.

Parameter:

ProcessId (STRING): hist. process ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Activities (BASE64): contains the requested activities in XML format

Example:

Structure of Activities

```
<Activities>
<Activity RActivityId="2D0131F8C0" Name="Check request" EntryNo="17"/>
<Activity RActivityId="32BAE" Name="Forward request" EntryNo="9"/>
</Activities>
```

Note:

Detailed description of Activities

§ RActivityId (STRING): instance ID of the activity

- Name (STRING): Activity name
- EntryNo (LONG): entry number reflects the chronological order.

See also:

wfm.GetHistProcessList, wfm.GetHistWorkItemRelEntriesByActivity

wfm.GetHistWorkItemRelEntriesByActivity

Description:

This job returns all history entries for the specified process step.

Parameter:

RActivityId (STRING): instance ID of the activity

Return values:

HistWorkItemRelEntries (BASE64): contains the requested entries in XML format

Example:

Structure of HistWorkItemRelEntries

```
<HistWorkItemRelEntries>
<HistWorkItemRelEntry RActivityId="" ActivityName="" OrganizationId=""
UserId="" UserName="" Reason="" HistType="" HistId="" EntryNo="" Time=""/>
</HistWorkItemRelEntries>
```

Note:

Detailed description of HistWorkItemRelEntries

- RActivityId (STRING): instance ID of the activity
- ActivityName (STRING): Activity name
- OrganizationId (STRING): Organization ID
- UserId (STRING): ID of the user who had the activity in his inbox
- UserName (STRING): User name
- Reason (LONG): reason why the user had the activity in his inbox
    - 0 = set up as participant
    - 1 = was assigned to the user based on a reminder time
    - 2 = assigned (HistType = 11) / removed (HistType = 12) by the administrator
    - 3 = assigned (HistType = 11) / removed (HistType = 12) by a script
- HistType (LONG): type of the hist. entry
    - 10 = STARTWORKITEM
    - 11 = PERSONALIZED
    - 12 = DEPERSONALIZED
- HistId (STRING): ID of the hist. entry
- EntryNo (LONG): reflects the chronological order
- Time (LONG): creation time of the entry

See also:

wfm.GetHistWorkItemRelActivitiesByProcess

wfm.GetHistWorkItemRelUsersByProcess

Description:

This job returns all users resp. roles for which process steps of the specified history process were inserted into their inboxes.

Parameter:

ProcessId (STRING): hist. process ID

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Users (BASE64): contains the requested user in XML format

Example:

Structure of Users

```
<Users>
<User UserId=".." UserName=".."/>
<User UserId=".." UserName=".."/>
</Users>
```

Note:

Detailed description of Workflows

§ UserId (STRING): ID of the user/role

§ UserName (STRING): name of the user/role

See also:

wfm.GetHistProcessList

wfm.GetHistWorkItemRelEntriesByUser

Description:

This job returns all activities processed by the specified user and the respective history entries of the specified history process.

Parameter:

ProcessId (STRING): hist. process ID

UserId (STRING): User ID

Return values:

HistWorkItemRelEntries (BASE64): contains the requested information in XML format

Example:

Structure of HistWorkItemRelEntries

```
<HistWorkItemRelEntries>
<HistWorkItemRelEntry RActivityId="" ActivityName="" OrganizationId=""
UserId="" UserName="" Reason="" HistType="" HistId="" EntryNo="" Time=""/>
</HistWorkItemRelEntries>
```

Note:

Detailed description of HistWorkItemRelEntries

- RActivityId (STRING): instance ID of the activity
- ActivityName (STRING): Activity name
- OrganizationId (STRING): Organization ID
- UserId (STRING): ID of the user who had the activity in his inbox
- UserName (STRING): User name
- Reason (LONG): reason why the user had the activity in his inbox
    - 0 = set up as participant
    - 1 = was assigned to the user based on a reminder time
    - 2 = assigned (HistType = 11) / removed (HistType = 12) by the administrator
    - 3 = assigned (HistType = 11) / removed (HistType = 12) by a script
- HistType (LONG): type of the hist. entry
    - 10 = STARTWORKITEM
    - 11 = PERSONALIZED
    - 12 = DEPERSONALIZED
- HistId (STRING): ID of the hist. entry
- EntryNo (LONG): reflects the chronological order
- Time (LONG): creation time of the entry

See also:

wfm.GetHistProcessList

Other jobs

- wfm.ConvertExportFile
- wfm.DeleteSysClienttypes
- wfm.Export
- wfm.GetSysClienttypes
- wfm.GetVersionInfo
- wfm.GetWFMInfo
- wfm.Import
- wfm.InsertSysClienttypes
- wfm.GetProjectList
- AdhocConfigTemplate

wfm.ConvertExportFile

Description:

This job converts an export file into the current format (e.g. from 4.20 to 4.50). The export file must be included with the job. The result file will then be returned along with the response.

Parameter:

File list: name and path of the workflow export file to be converted

Return:

(INT): 0 = job successful, otherwise error code

Return values:

File list: name and path of the converted workflow export file

wfm.Export

Description:

This job exports the organizational structure, workflow projects and/or workflows.

Parameter:

OrganizationId (STRING): organization ID from which the export is to take place

OrgDoExport (INT): flag indicates whether organizational data will be exported, too (1=yes, 0=no)

WFProjectTree (BASE64): the structure indicates which projects and workflow models are to be exported (XML format)

ReportConfigDoExport (INT): flag indicates whether the report configuration of the organization are to be exported

ReportConfigIds (String): comma-separated list of report configuration IDs which will be exported. If the parameter ReportConfigDoExport = 1 and this list is empty, all configurations of the organization are exported.

AdhocRoutingListTemplateDoExport (INT): flag indicates if routing list templates of the organization are to be exported

AdhocRoutingListTemplateIds (String): comma-separated list of IDs of routing list templates to be exported. If the parameter AdhocRoutingListTemplateDoExport = 1 and this list is empty, all routing templates of the organization are exported.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

File list: name and path of the export file

Example:

Structure of WFProjectTree

```
<WFProjectTree>
<WorkflowProject Id="" CompleteExport=0>
<WorkflowProject Id="">
<WorkflowProject Id="" CompleteExport=0>
</Workflow Id="">
</Workflow Id="">
</WorkflowProject>
</WorkflowProject Id="" CompleteExport=1>
</WorkflowProject>
</WorkflowProject Id="" CompleteExport=1>
```

```
</WorkflowProject>
</WorkflowProject Id="" CompleteExport=1>
</Workflow Id="">
</Workflow Id="">
</WFProjectTree>
```

Note:

Detailed description of WFProjectTree

- WorkflowProject: structure (may be cascaded) that indicates which workflow project (can also be a workflow family) is to be exported
  - ID (STRING): ID of the workflow project
  - CompleteExport (LONG): this flag indicates whether all projects or workflows connected to the project are to be exported. If it is set, the substructure does not need to be specified any further.
- Workflow: structure indicating which workflow (which workflows) will be exported
  - ID (STRING): workflow ID

See also:

wfm.Import

wfm.GetVersionInfo

Description:

This job returns information on the version of the workflow engine.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

EngineVersion (INT): specifies the major version of the engine e.g. 450

EngineSubVersion (INT): specifies the sub version of the engine e.g. 3

wfm.GetWFMInfo

Description:

This job returns information on a user (e.g. absence, WF user ID) through the DRT user ID.

Parameter:

Requests (BASE64): contains a list of search requests in XML format

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Requests (BASE64): contains responses to search requests in XML format

Example:

Structure of the input parameter Requests

```
<Requests>
```

```
<Request UserId="" RequestType=""/>
<Request UserId="" RequestType=""/>
</Requests>
```

Note:

Detailed description of the input parameter Requests

- UserId (STRING): ID of the DRT user
- RequestType (INT): flag for the search request type
  - 1 = determine the ID of the active organization
  - 2 = determine if the DRT user if part of the active organization
  - 3 = determine the ID of the WF user for a DRT user ID
  - 4 = determine if the specified user is set to absent

Example:

Structure of the output parameter Requests

```
<Requests>
<Request UserId="" RequestType="" Value=""/>
<Request UserId="" RequestType="" Value=""/>
</Requests>
```

Note:

Detailed description of the output parameter Requests

- UserId (STRING): ID of the DRT user
- RequestType (INT): flag for the search request type
  - 1 = determine the ID of the active organization
  - 2 = determine if the DRT user if part of the active organization
  - 3 = determine the ID of the WF user for a DRT user ID
  - 4 = determine if the specified user is set to absent
- Value (INT): search request response
  - with RequestType = 2: 1 = DRT user contained in the organization, otherwise 0
  - with RequestType = 4: 1 = DRT user set to absent, otherwise 0

wfm.Import

Description:

This job imports an organization. Therefore a file is added to the job.

Parameter:

Input file: name and path of the file to be imported.

DoImportOrganization (INT): indicates whether organizational data is also to be imported (1=yes, 0=no)

DestOrganizationId (BASE64): if organizational data is to be imported, the target organization is specified here. The parameter is left blank if a new organization should be created.

WorkflowProjects (BASE64): specifies the workflow projects, which are to be imported, in XML format.

Workflows (BASE64): specifies the workflow models, which are to be imported, in XML format.

ReportConfigs (BASE64): specifies the report configurations, which are to be imported, in XML format.

Templates (BASE64): specifies the routing list templates, which are to be imported, in XML format.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ChangedOrganizations (STRING): comma-separated ID list of modified organizations

Example:

Structure of WorkflowProjects

```
<WorkflowProjects>
</WorkflowProject Id="" DestPrjId="" DestOrgId="" Overwrite=""
CompleteImport="" OldParent="">
</WorkflowProject Id="" DestPrjId="" DestOrgId="" Overwrite=""
CompleteImport="" OldParent="">
</WorkflowProject Id="" DestPrjId="" DestOrgId="" Overwrite=""
CompleteImport="" OldParent="">
</WorkflowProjects>
```

Note:

Detailed description of WorkflowProjects

§ ID (STRING): ID of the project in the import file

§ DestPrjId (STRING): ID of the parent project

§ DestOrgId (STRING): ID of the target organization. This attribute is not specified if a new organization should be created on import and if the project will have this new organization as its target.

§ Overwrite (INT): this flag indicates if an existing project with the same ID is to be overwritten. Otherwise a new project with a new ID is created.

§ CompleteImport (INT): this flag indicates whether the entire substructure (projects/workflows) of the project is to be imported. This does not then need to be listed.

§ OldParent (INT): this flag is only required if the parent workflow projects were copied (i.e. not overwritten). If the flag was set, the project is written into the old WF project (whereby the overwrite flag is taken into account) – if not, it is written into the created (copied) WF project. This flag must always be set if the parent WF project is supposed to be overwritten.

Example:

Structure of Workflows

```
<Workflows>
</Workflow Id="" DestFamId="" DestOrgId="" Overwrite="" OldFamily="">
</Workflow Id="" DestFamId="" DestOrgId="" Overwrite="" OldFamily="">
</Workflow Id="" DestFamId="" DestOrgId="" Overwrite="" OldFamily="">
</Workflows>
```

Note:

Detailed description of Workflows

§ ID (STRING): ID of the workflow in the import file

- DestFamId (STRING): ID of the target WFFamily
- DestOrgId (STRING): ID of the target organization. This attribute is not specified if a new organization should be created on import (see node description: organization) and if the model will have this new organization as its target.
- Overwrite (INT): this flag indicates whether an existing workflow with the same ID is to be overwritten. Otherwise a new workflow with a new ID is created.
- OldFamily (INT): this flag is only required if the parent workflow families was copied (i.e. not overwritten). If the flag was set, the model is written into the old WF family (whereby the overwrite flag is taken into account) – if not, it is written into the created (copied) WF family. This flag must always be set if the parent WF family is supposed to be overwritten.

Example:

Structure of ReportConfigs

```
<ReportConfigs>
<ReportConfig ConfigId="" Overwrite="" DestOrgId=""/>
<ReportConfig ConfigId="" Overwrite="" DestOrgId=""/>
<ReportConfig ConfigId="" Overwrite="" DestOrgId=""/>
</ReportConfigs >
```

Note:

Detailed description of Workflows

- ConfigId (STRING): ID of the configuration in the import file
- DestOrgId (STRING): ID of the target organization.
- Overwrite (INT): this flag indicates whether an existing configuration with the same ID is to be overwritten. Otherwise a new configuration with a new ID is created.

See also:

wfm.Export

wfm.GetSysClienttypes

Description:

This job returns all client types defined for the system.

Return:

(INT): 0 = job successful, otherwise error code

Return values:

ClientTypes (Base64): list of all defined client types

Example:

Structure of Clienttypes

```
<ClientTypes>
<ClientType Id="" Name=""/>
        <ClientType Id="" Name=""/>
</Clienttypes>
```

Note:

Detailed description of client types

- ID (STRING): GUID of the client type

§ Name (STRING): name of the client type

See also:

wfm.InsertSysClienttypes

wfm.InsertSysClienttypes

Description:

This job has not been implemented yet. This job defines new client types.

Parameter:

Clienttypes (Base64): list of all defined client types

Return:

(INT): 0 = job successful, otherwise error code

Example:

Structure of Clienttypes

```
<Clienttypes>
<Clienttype Id="" Name=""/>
        <Clienttype Id="" Name=""/>
</Clienttypes>
```

Note:

Detailed description of Clienttypes

§ ID (STRING): GUID of the client type

§ Name (STRING): name of the client type

See also:

wfm.GetSysClienttypes

wfm.DeleteSysClienttypes

Description:

This job has not been implemented yet. This job deletes the specified client types.

Parameter:

ClientTypeIds (STRING): comma-separated GUIDs of the client types which will be deleted

Return:

(INT): 0 = job successful, otherwise error code

See also:

wfm.GetSysClienttypes

wfm.GetProjectList

Description:

This job returns all workflow projects for an organization.

Parameter:

OrganizationId (STRING): GUID of the organization

Return:

(INT): 0 = job successful, otherwise error code

Return values:

Projects (Base64): list of all defined client types

Example:

Structure of Clienttypes

```
<Projects>
<Project Id="" Name="" Type="" CreatorId="" CreationTime="" Description=""/>
        <Project Id="" Name="" Type="" CreatorId="" CreationTime="" Description=""/>
</Projects>
```

Note:

Detailed description of Clienttypes

§ ID (STRING): project ID

§ Name (STRING): project name

§ Type (INT): 1 = root project, 2 = workflow project, 3 = workflow family

§ CreatorID (STRING): GUID of the creator

§ CreationTime (INT): Time of creation

§ Description (STRING): Description

wfm.AdhocConfigTemplate

Description:

This job is used to configure ad hoc templates for the workflow.

Parameter:

UserId (STRING): User ID

OrgId (STRING): instance of the activity

Action (Int): 1: saving a template, 2: deleting a template, 3: publishing a template, 4: personalizing a template

Depending on the particular action, other parameters may be required:

1: saving a template

TemplateId (String): template ID. Is empty if the template is saved for the first time.

TemplateName(String): template name

Public   (Int): 0: this is a private template, 1: the template is public

Template(BASE64):

2: deleting a template

TemplateId (String): template ID.

3: publishing a template

TemplateId (String): template ID

4: personalizing a template:

TemplateId (String): template ID.

Return:

The return value depends on the selected action (Action(Int)):

1: saving a template

TemplateId (String): template ID.

2: deleting a template

no return parameter

3: publishing a template

no return parameter

4: personalizing a template:

no return parameter

Example:

Structure of Template

```
<RoutingList Id="3294B433BFF6454D9C861B86B5A8AD5D"
ActivityId="3294B433BFF6454D9C861B86B5A8AD5D" Expandable="1">
<Entries>
<Entry Nr="203" Expandable="1">
<Item Id="99825B18A8334987935684FDA3D6A40D"
ActivityId="6EE4490A48164A0FA6DC34A80099AF66" ActivityName="Create invoice"
ModelActivityName="Create invoice" Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
</Entry>
<Entry Nr="253" Expandable="1">
<Item Id="E15594D692C14FDA9AFDE8FA0A43F6E4"
ActivityId="6EE4490A48164A0FA6DC34A80099AF67" ActivityName="Approve invoice BL"
ModelActivityName="Approve invoice"  Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
<Item Id="C6DA9503CD874D69A9B703D0E06A52E8"
ActivityId="6EE4490A48164A0FA6DC34A80099AF67" ActivityName="Approve invoice GF"
ModelActivityName="Approve invoice" Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
</Entry>
</Entries>
</RoutingList>
```

Note:

Detailed description of RoutingList

§ RoutingList: routing list with the following structure (or subsets of it)

§ ID (String): routing list ID. The value is set by the server and must not be changed.

§ ActivityId (String): activity ID

- Expandable (Int): 0: routing list cannot be expanded, 1: routing list can be expanded
    - Entries: the structure combines entries of the routing list. An entry consists of multiple elements which can be executed simultaneously.
    - Entry: describes an entry in the routing list.
    - No (Int): for relative sorting of entries within the routing list. The absolute values do not have any influence on the client.
    - Expandable (Int): 0: entry cannot be expanded, 1: entry can be expanded
    - Item: describes an element of the routing list. This can be an activity, an executing person or a deadline.
    - ID (STRING): for identification This ID must not be changed and must be identically sent for all jobs. If an item was created by the client, the ID must be stated here.
    - ActivityId (String): ID of the activity in the workflow model
    - ActivityName (String): activity name (does not necessarily have to match the name in the workflow model).
    - ActivityModelName (String): ID of the activity in the workflow model
    - TimerId(String): ID of a reminder time
    - TimerName(String): name of the reminder time
    - TimerDuration(Int): timer duration
    - TimerDurationType(Int): 0: no period, 1: relative, 2: absolute
    - Changeable(Int): 0: no change possible, 1:The element can be changed by the client.
    - Deleteable(Int): 0: deletion not allowed, 1: element can be deleted
    - Remark (String): note on editing (Text)
    - ObjectsIds (String): list of editors' GUIDS (roles or persons), separated by comma

wfm.AdhocGetTemplateList

Description:

Returns multiple ad hoc templates for the specified user.

Parameter:

UserId (STRING): User ID

OrgId (STRING): instance of the activity

TemplateId (String): template ID. If this parameter is empty, all ad hoc templates are determined which are visible for the user (= all public and personalized templates)

Return:

Templates (BASE64): list of the determined ad hoc templates

Template: describes an ad hoc template

TemplateId(String): ID of the ad hoc template

TemplateName(String): name of the ad hoc template

Public(int): 0: the template is not public, 1: the template is public

- RoutingList: routing list with the following structure (or subsets of it)
- ID (String): routing list ID. The value is set by the server and must not be changed.
- ActivityId (String): activity ID
- Expandable (Int): 0: routing list cannot be expanded, 1: routing list can be expanded
  - Entries: the structure combines entries of the routing list. An entry consists of multiple elements which can be executed simultaneously.
  - Entry: describes an entry in the routing list.
  - No (Int): for relative sorting of entries within the routing list. The absolute values do not have any influence on the client.
  - Expandable (Int): 0: entry cannot be expanded, 1: entry can be expanded
  - Item: describes an element of the routing list. This can be an activity, an executing person or a deadline.
  - ID (STRING): for identification This ID must not be changed and must be identically sent for all jobs. If an item was created by the client, the ID must be stated here.
  - ActivityId (String): ID of the activity in the workflow model
  - ActivityName (String): activity name (does not necessarily have to match the name in the workflow model).
  - ActivityModelName (String): ID of the activity in the workflow model
  - TimerId(String): ID of a reminder time
  - TimerName(String): name of the reminder time
  - TimerDuration(Int): timer duration
  - TimerDurationType(Int): 0: no period, 1: relative, 2: absolute
  - Changeable(Int): 0: no change possible, 1:The element can be changed by the client.
  - Deleteable(Int): 0: deletion not allowed, 1: element can be deleted
  - Remark (String): note on editing (Text)
  - ObjectsIds (String): list of editors' GUIDS (roles or persons), separated by comma

Example:

Structure of Templates

```
<Templates>
<Template TemplateId="" TemplateName="" Public="">
<RoutingList Id="3294B433BFF6454D9C861B86B5A8AD5D"
ActivityId="3294B433BFF6454D9C861B86B5A8AD5D" Expandable="1">
<Entries>
<Entry Nr="203" Expandable="1">
<Item Id="99825B18A8334987935684FDA3D6A40D"
ActivityId="6EE4490A48164A0FA6DC34A80099AF66" ActivityName="Create invoice"
ModelActivityName="Create invoice" Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
</Entry>
<Entry Nr="253" Expandable="1">
<Item Id="E15594D692C14FDA9AFDE8FA0A43F6E4"
ActivityId="6EE4490A48164A0FA6DC34A80099AF67" ActivityName="Approve invoice BL"
```

```
ModelActivityName="Approve invoice"  Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
<Item Id="C6DA9503CD874D69A9B703D0E06A52E8"
ActivityId="6EE4490A48164A0FA6DC34A80099AF67" ActivityName="Approve invoice GF"
ModelActivityName="Approve invoice" Remark="" TimerId="" TimerDuration=""
TimerDurationType="" Changeable="1" Deleteable="0">
<ObjectIds></ObjectsIds>
</Item>
</Entry>
</Entries>
</RoutingList>
</Template>
</Templates>
```

Server-internal Jobs

The jobs listed here are server-internal and are therefore generally not called from the outside.

wfm.DBCommands

Description:

This job executes cached database commands. This job is generally not executed from the outside.

Parameter:

DBCommandsId (STRING): GUID of the database command to be executed in the database command cache

Return:

(INT): 0 = job successful, otherwise error code

BatchJobs

The listed jobs are periodically executed by the workflow engine.

§ wfm.CheckJob

§ wfm.WorkerJob

§ wfm.WorkItemNoti

wfm.CheckJob

Description:

This job verifies whether dunning or retention periods have expired and reacts accordingly. Activities affected by retention periods are activated and the retention periods are removed from the database. Activities affected by dunning periods are marked as delayed and have their defined action (send e-mail message, forward to substitute) executed.

Return:

(INT): 0 = job successful, otherwise error code

wfm.WorkerJob

Description:

This job processes the activities queue. Activities are started or ended by this job.

Return:

(INT): 0 = job successful, otherwise error code

wfm.WorkItemNoti

Description:

This job sends the ServerJob 'ServerNotifyClients'. The job ServerNotifyClients informs all connected clients about changes in the inbox.

Return:

(INT): 0 = job successful, otherwise error code

ServerCommunicationJobs

Jobs listed here serve the communication between different servers. Contrary to other jobs these jobs are not executed by clients but by a (different) server.

§ wfm.ServerNotifyClients

§ wfm.ServerUpdateWorkflowModels

§ wfm.ServerUserAbsent

wfm.ServerNotifyClients

Description:

This job sends a message reporting the update of the inbox to the clients of all users who are contained in the list of user IDs. If this list is empty, a message is sent to all connected clients.

Parameter:

UserGUIDs (STRING): comma-separated list of user GUIDs

Return:

(INT): 0 = job successful, otherwise error code

wfm.ServerUpdateWorkflowModels

Description:

This job deletes all listed workflow models from the workflow engine cache and sends a notification to all attached operating systems: 4.DRT-Workflow_Editors.

Parameter:

OrganizationId (STRING): GUID of the organization from which the models originate

WorkflowIds (STRING): comma-separated list of workflow model GUIDs

Return:

(INT): 0 = job successful, otherwise error code

wfm.ServerUserAbsent

Description:

This job informs all attached operating systems 4.DRT-Workflow_Editors for which user the presence status has changed.

Parameter:

OrganizationId (STRING): GUID of the organization from which the models originate

AbsentIds (STRING): comma-separated list of user GUIDs, which are present

PresentIds (STRING): comma-separated list of user GUIDs, which are absent

Return:

(INT): 0 = job successful, otherwise error code

Core Services

Core Services contain functions related to administration, licensing, session management, engine administration and internal control. These are encapsulated by the application server core (axsvckrn.exe). This makes it possible to execute basic administrative functions without loading additional engines.

Namespaces

§ Administrations-Core-Services (Namespace adm)

§ Kernel-Core-Services (Namespace krn)

§ License-Core-Services (Namespace lic)

Administration Core Services (Namespace adm)

In the 'adm' namespace exist functions used to administer system files and a few configuration tasks at the server. This namespace is directly implemented by the kernel.

- adm.CleanUpConfig
- adm.CleanUpLog
- adm.EnumServerGroups
- adm.EnumServers
- adm.GetServerFamilyInfo
- adm.GetServersActivity
- adm.GetSystemFile
- adm.LogdirDeleteFiles
- adm.LogdirDownloadFiles
- adm.LogdirGetInfo
- adm.StoreSystemFile

adm.CleanUpConfig

Description:

This job looks for the specified configuration file and empties it.

Parameter:

Flags (INT): not currently supported

Types (INT): type of configuration file

- 1 = object definition file
- 2 = AS.cfg
- 4 = AsListen.dat
- 8 = AsImpExp.Cfg
- 16 = AsForm.Cfg
- 32 = AsCold.Cfg
- 4294967295 = all

Versions (INT): file version

Return:

(INT): 0 = job successful, otherwise error code

adm.CleanUpLog

Description:

This job looks for the specified log files and removes them.

Parameter:

Flags (INT): not currently supported

Type (INT): type of log files to be deleted

- 1 = FLW files
- 2 = ERR files
- 4 = LOG files
- 8 = REP files
- 4294967295 = all

Days (INT): Days

Path (STRING): log directory path

Return:

(INT): 0 = job successful, otherwise error code

adm.EnumServerGroups

Description:

This job returns a list of all server groups.

Parameter:

Flags (INT): 0 = return value is Group[00..nn]; 1 = return values are Info and InfoType

Return values:

[Info] (STRING): MIME encoded buffer with information on the server group

[InfoType] (STRING): semicolon-separated names of values returned in Info

[Group[00..nn]] (STRING): semicolon-separated information on server group

- ID of the server group
- Server group name
- Description of the server group

Return:

(INT): 0 = job successful, otherwise error code

adm.EnumServers

Description:

This job returns all servers of a server group

Parameter:

Flags (INT): 0 = return value is server[00..nn]; 1 = return values are Info and InfoType

[GroupID] (INT): ID of the server group

Return values:

[Info] (STRING): MIME encoded buffer with information on the server group

[InfoType] (STRING): semicolon-separated names of values returned in Info

[Server [00..nn]]: semicolon-separated information concerning server

- IP of the server

- Name of the server
- IP address
- Port
- Service name

Return:

(INT): 0 = job successful, otherwise error code

adm.GetServerFamilyInfo

Description:

This job returns the ID and the name of the server family.

Parameter:

Flags (INT): not currently supported

Return values:

GUID (STRING): ID of the server family

Name (STRING): name of the server family

Return:

(INT): 0 = job successful, otherwise error code

adm.GetServersActivity

Description:

This job returns all IDs of servers and their status.

Parameter:

Flags (INT): not currently supported

Return values:

Server[00..nn] (STRING): semicolon-separated information concerning server

- ID of the server
- Status of the server
  - 1 = Server running
  - 2 = Server crashed

Return:

(INT): 0 = job successful, otherwise error code

adm.GetSystemFile

Description:

The job transfers the system file (e.g. as.cfg) to the client and locks it. System files are written to the database table 'osresources' and have the resource type 1.

Parameter:

Flags (INT):

- § LOWORD(Flags) = 0: the system file is opened for reading
- § LOWORD(Flags) = 2: the system file is opened for writing
- § Version != 0 or HIWORD(Flags) != 0 or LOWORD(Flags) != 2
  - § HIWORD(Flags) = 1: LowDateTime and HighDateTime are returned
  - § HIWORD(Flags) = 2: LowDateTime is returned

FileName (STRING): name of the system file

Version (INT): Version (INT): version of the system file to be requested

Return values:

[FileCount] (INT): FileCount equals 1

[LowDateTime] (INT): LowDateTime

[HighDateTime] (INT): HighDateTime

[File list]: name and path of the system file

Return:

(INT): 0 = job successful, otherwise error code

See also:

adm.StoreSystemFile

adm.LogdirDeleteFiles

Description:

This job deletes the specified files inside the log directory of the server.

Parameter:

Flags (INT): not currently supported

Files (STRING): file names separated by '?'

Return values:

Deleted (INT): number of deleted files

Failed (INT): number of files that could not be deleted

Return:

(INT): 0 = job successful, otherwise error code

See also:

adm.LogdirGetInfo

adm.LogdirDownloadFiles

Description:

This job returns the specified files inside the log directory of the server. The files are returned in a compressed form.

Parameter:

Flags (INT): not currently supported

Files (STRING): file names separated by '?'

Return values:

Files (STRING): names of the returned files

File list: File list: name and path of the compressed file (contains all refer to requested files)

Return:

(INT): 0 = job successful, otherwise error code

See also:

adm.LogdirGetInfo

adm.LogdirGetInfo

Description:

This job returns a list of all files inside the log directory of the server.

Parameter:

Flags (INT): not currently supported

Return values:

FileInformation[00000000..nnnnnnnnn] (STRING): file information separated by '?'

§ File name

§ File size in Byte

§ Timestamp of the last modification

Return:

(INT): 0 = job successful, otherwise error code

See also:

adm.LogdirDownloadFiles

adm.StoreSystemFile

Description:

This job saves the system file received by the client.

Parameter:

Flags (INT):

§ LOWORD(Flags) = 0: save system file and write history file

§ LOWORD(Flags) = 1: undo locking of the file

§ HIWORD(Flags) = 2: save date stamp in LowDateTime format; otherwise use LowDateTime format and HighDateTime format

FileName (STRING): name of the system file

LowDateTime (INT): date stamp in LowDateTime format

HighDateTime (INT): date stamp in HighDateTime format

File list: name and path of the system file

Return:

(INT): 0 = job successful, otherwise error code

See also:

adm.GetSystemFile

Kernel Core Services (Namespace krn)

Jobs of the 'krn' kernel executor are used for the internal administration of the application server processes. These are mainly functions for batch management, server monitoring, registry administration and administration of loaded engines at runtime.

§ Registry administration

§ Batch administration

§ Server administration

§ Session administration

§ Engine administration

§ Other jobs

Registry Administration

These jobs are for registry administration. Registry entries can be read or changed.

§ krn.REBackup

§ krn.REGetCurrentSchema

§ krn.REGetRegValue

§ krn.RELoad

§ krn.RESave

§ krn.RESetRegValue

krn.REBackup

Description:

This job creates a backup of the current registry schema in XML format.

Parameter:

Flags (INT): not currently supported

FileName (STRING): Name of the destination file including path information; the file extension should be .xml

Return:

(INT): 0 = job successful, otherwise error code

krn.REGetCurrentSchema

Description:

This job returns the current registry schema in XML format.

Parameter:

Flags (INT): not currently supported

Return values:

Schema (STRING): contains the current schema in XML notation

Return:

(INT): 0 = job successful, otherwise error code

krn.REGetRegValue

Description:

This job returns the value of a specified registry entry.

Parameter:

Flags (INT): not currently supported

Name (STRING): name of the registry entry

Return values:

Value (STRING): Value of the registry entry

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.RESetRegValue

krn.RELoad

Description:

This job loads the current registry schema into memory.

Parameter:

Flags (INT): not currently supported

Return:

(INT): 0 = job successful, otherwise error code

krn.RESave

Description:

This job saves the current registry schema. The key is HKLM\SOFTWARE\OPTIMAL SYSTEMS\[service name of the application server]\Schemata \[version number(e.g. 4.0)].

Parameter:

Flags (INT): not currently supported

Return:

(INT): 0 = job successful, otherwise error code

krn.RESetRegValue

Description:

This job changes the value of a specified registry entry.

Parameter:

Flags (INT): not currently supported

Name (STRING): Name of the registry entry

Value (STRING): Value of the registry entry

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.REGetRegValue, krn.RESave

Batch administration

§ krn.BatchAdd

§ krn.BatchChange

§ krn.BatchEnum

§ krn.BatchGetStatistic

§ krn.BatchRemove

krn.BatchAdd

Description:

This job adds a batch to the registry.

Parameter:

Flags (INT): not currently supported

Registry (STRING): name in the registry

Name (STRING): name of the batch to be added

NameSpace (STRING): namespaces of the job to be executed

JobName (STRING): name of the job to be executed (without namespace)

Scheduling (STRING): time of execution

Period (INT): duration in ms between the job calls

Enabled (BOOL): 1 = batch is enabled, otherwise 0

DoLog (BOOL): 1 = job will be logged, otherwise 0

Parameters: name, data type, job value (MIME encoded buffer)

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.BatchChange, krn.BatchRemove

krn.BatchChange

Description:

This job changes an existing batch.

Parameter:

Flags (INT): not currently supported

Registry (STRING): name in the registry

Name (STRING): name of the batch to be added

NameSpace (STRING): namespaces of the job to be executed

JobName (STRING): name of the job to be executed (without namespace)

Scheduling (STRING): time of execution

Period (INT): duration in ms between the job calls

Enabled (BOOL): 1 = batch is enabled, otherwise 0

DoLog (BOOL): 1 = job will be logged, otherwise 0

Parameters: name, data type, job value (MIME encoded buffer)

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.BatchEnum

krn.BatchEnum

Description:

This job returns a list of existing batches.

Parameter:

Flags (INT): not currently supported

Return values:

Batch[1..n] (STRING): batch information

Return:

(INT): 0 = job successful, otherwise error code

Note:

returned batch information

§ Registry: name of the registry entry

§ Name: batch name

§ Period: duration in ms between the job calls

§ Scheduling:

§ JobName: name of the job to be executed (without namespace)

§ NameSpace: namespaces of the job to be executed

§ Enabled 1 = batch is enabled, otherwise 0

§ DoNotLog: 1 = job will be logged, otherwise 0

§ TimeCreated: batch creation time

§ TimeEnabled: activation time

§ TimeFired: time of last execution

§ Parameters: MIME encoded parameter of the job (name, data type, value)

See also:

krn.BatchAdd, krn.BatchChange, krn.BatchRemove, krn.BatchGetStatistic

krn.BatchGetStatistic

Description:

This job returns statistical parameters for the selected batch.

Parameter:

Flags (INT): not currently supported

Name (STRING): name of the batch for which the job parameters are output

Return values:

Ticks (STRING): number of job executions

PeriodAve (STRING): average time between ticks (in seconds)

PeriodLast (STRING): time between the last and the penultimate tick (in seconds)

DeviationAveP (STRING): average variance of intervals between ticks and the preset (in percent)

DeviationAveS (STRING): average variance of intervals between ticks and the preset (in seconds)

DeviationLastP (STRING): variance between the last two ticks and the preset (in percent)

DeviationLastS (STRING): variance between the last two ticks and the preset (in seconds)

DeviationMaxP (STRING): maximum variance of intervals between ticks and the preset (in percent)

DeviationMaxS (STRING): maximum variance of intervals between ticks and the preset (in seconds)

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.BatchEnum

krn.BatchRemove

Description:

This job removes the existing batch.

Parameter:

Flags (INT): not currently supported

Registry (STRING): Name of the registry entry

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.BatchEnum

Server Manager

These jobs serve the server administration.

- krn.AppsEventsEnum
- krn.AppsEventsSubscribe
- krn.CheckCrashedServers
- krn.CheckServerConnection
- krn.GetServerInfo
- krn.GetServerInfoEx
- krn.MakeBeatPing
- krn.RefillServerList
- krn.ShutDown

krn.AppsEventsEnum

Description:

This job returns all defined event types which are implemented in the server.

Parameter:

Flags (INT): not currently supported

Return values:

Info (STRING): semicolon-separated MIME-encoded buffer which receives event types

Infotype (STRING): contains the description of the data returned in the parameter info

Return:

(INT): 0 = job successful, otherwise error code

krn.AppsEventsSubscribe

Description:

This job makes the server send out notifications for the specified events to the executor of the job.

Parameter:

Flags (INT): not currently supported

Events (STRING): semicolon-separated events

§ krn.SessionLogin

§ krn.SessionLogout

§ krn.Connected

§ krn.Disconnected

§ krn.JobCall

§ krn.Executor

Return:

(INT): 0 = job successful, otherwise error code

krn.CheckCrashedServers

Description:

This job finds crashed servers and releases their resources. Such servers have their server status in the database table 'ospingtable' set to 2 = 'hung server'.

Parameter:

Flags (INT): not currently supported

Return:

(INT): 0 = job successful, otherwise error code

krn.MakeBeatPing

krn.CheckServerConnection

Description:

This job checks the server connection by incrementing a global counter variable for every job call by 1.

Parameter:

Flags (INT): not currently supported

Return values:

Callnumber (LONG): number of job calls

Return:

(INT): 0 = job successful, otherwise error code

krn.GetServerInfo

Description:

This job returns certain server information.

Parameter:

Flags (INT): not currently supported

Info (INT): information type (see Tab) which will be queried

Return values:

Info (INT): equals the input parameter

Name (STRING): string corresponding to the info

Value (STRING): outputs the queried server information

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.GetServerInfoEx

Info type	**Info string**	**Description**
1	ServerID	ID of the server
2	ComputerName	Name of the server
3	InstanceName	Name of the program instance
4	ComString	Server ComString (e.g. 127.0.0.1)
5	DataBaseSourceC	Database name (e.g. os400)
6	DataBaseParser	Database parser (e.g. oxorantl.dll)
7	ClientETC	Path to the client configuration files
8	StatusLine	String of the status bar
9	ServerType	0 = main server; 1 = secondary server
10	ValidDomains	
11	DataBaseModule	
12	DataBaseParsType	
13	DataBaseSchema	
14	DataBaseProvider	
15	DataBaseConString	
16	ETC	full path of the server directory 'etc'
17	DataBaseNewMethod	
18	DataBaseModuleS	
19	DataBaseModuleClient	
1000	FileVersionListUpdate	returns all file names and the time of creation of the server directory server\etc\update
1001	FileVersionListClient	returns all file names and the time of creation of the server directory server\etc\update\client
1002	FileVersionListAdmin	returns all file names and the time of creation of the server directory server\etc\update\admin
1003	FileVersionListIndex	returns all file names and the time of creation of the server directory server\etc\update\index

1004	FileVersionListTemplate	returns all file names and the time of creation of the server directory server\etc\templates

krn.GetServerInfoEx

Description:

This job returns the specified server information.

Parameter:

Flags (INT): not currently supported

Info (STRING): semicolon-separated parameters which will be output (;parameter;parameter;...)

§ ? -> the names of all parameters are returned as a list

§ empty string -> all parameters and their values are returned as a list

Return values:

[Param[000...NNN]] (STRING): Output only for Info = ?; names of all parameters which can be transferred via Info

[parameter name] (STRING): Server Information

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.GetServerInfo

krn.MakeBeatPing

Description:

This job performs a ping to the application server, the table 'ospingtable' will be updated with server status 1 = 'server active'.

Parameter:

Flags (INT): not currently supported

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.CheckCrashedServers

krn.RefillServerList

Description:

This job loads the server information from the database tables 'ospingtable' and 'sever' into memory.

Parameter:

Flags (INT): not currently supported

Return:

(INT): 0 = job successful, otherwise error code

krn.ShutDown

Description:

This job shuts down the server.

Parameter:

Flags (INT): not currently supported

Return:

(INT): 0 = job successful, otherwise error code

Session Administration

These jobs serve the session administration.

- krn.SessionAttach
- krn.SessionDeleteLost
- krn.SessionDrop
- krn.SessionDropDB
- krn.SessionEnum
- krn.SessionEnumDB
- krn.SessionEnumResourcesDB
- krn.SessionGetInfo
- krn.SessionLogin
- krn.SessionLogout
- krn.SessionPropertiesEnum
- krn.SessionPropertiesGet
- krn.SessionPropertiesSet
- krn.UserSessionCreate
- krn.UserSessionDelete

krn.SessionAttach

Description:

This job creates a new work session with the application server or resumes an existing work session.

Parameter:

Flags (INT): not currently supported

SessionGUID (STRING): GUID of the session which will be resumed; empty = a new session will be created

Return values:

SessionGUID (STRING): GUID of the new session or the existing session

Return:

(INT): 0 = job successful, otherwise error code

krn.SessionDeleteLost

Description:

This job deletes all sessions which are no longer online from the database.

Parameter:

Flags (INT): 1 = parameter AgeHours applies

AgeHours (INT): number of hours

Return values:

DeletedSessions (STRING): ID of the deleted sessions

Return:

(INT): 0 = job successful, otherwise error code

krn.SessionDrop

Description:

This job deletes the specified session.

Parameter:

Flags (INT): not currently supported

SessionGUID (STRING): ID of the session

Return:

(INT): 0 = job successful, otherwise error code

krn.SessionDropDB

Description:

This job deletes all entries of a session in the database (oslockedres, ossession).

Parameter:

Flags (INT): not currently supported

SessionGUID (STRING): ID of the session

Return:

(INT): 0 = job successful, otherwise error code

krn.SessionEnum

Description:

This job returns a list of all existing sessions connected to the server.

Parameter:

Flags (INT): not currently supported

Return values:

Sessions (STRING): semicolon-separated list of GUIDs of all active sessions

Return:

(INT): 0 = job successful, otherwise error code

krn.SessionEnumDB

Description:

This job returns a list of all database entries for the sessions.

Parameter:

Flags (INT): not currently supported

Return values:

SessionInfoType (STRING): contains the description of the data returned in the parameter sessions

Sessions (STRING): MIME encoded buffer with information on the session

Return:

(INT): 0 = job successful, otherwise error code

krn.SessionEnumResourcesDB

Description:

This job returns all used resources saved in the database for the specified session.

Parameter:

Flags (INT): not currently supported

SessionGUID (STRING): ID of the session

Return values:

Resources (STRING): semicolon-separated resources (format: GUID1=Locktime1,Name1;GUID2=Locktime2,Name2;...)

- ID of the resource
- time at which the resource was locked
- short description of the resource

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.SessionEnum

krn.SessionGetInfo

Description:

This job returns information on the sessions transferred in the GUID that exist at the server.

Parameter:

Flags (INT): not currently supported

Sessions (STRING): semicolon-separated IDs of the sessions

Return values:

SessionInfoType (STRING): information identifiers which are returned in the parameter SessionInfo

ConnectionInfoType (STRING): information identifiers which are returned in the parameter SessionInfo

SessionInfo (STRING): MIME encoded buffer with information on the session

Return:

(INT): 0 = job successful, otherwise error code

krn.SessionLogin

Description:

This job logs a user on to the active session. The user is identified by his name and the encrypted password.

Parameter:

Flags (INT): not currently supported

UserName (STRING): User name

UserPwd (STRING): encrypted password, information on password encryption can be obtained from OPTIMAL SYSTEMS GmbH

[EntMgr] (LONG): parameter should only be set for Enterprise Manager Start

Return values:

Description (STRING): description if an error occurs

Action (STRING): action which was executed

[UserGUID] (STRING): GUID of the user who was logged in (only with Enterprise Manager)

[UserID] (INT): ID of the user who was logged in (only with Enterprise Manager)

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.SessionAttach, krn.SessionLogout, krn.SessionPropertiesSet

krn.SessionLogout

Description:

This job ends the use of the active session.

Parameter:

Flags (INT): not currently supported

Return:

(INT): 0 = job successful, otherwise error code

krn.SessionPropertiesEnum

Description:

This job returns the names of all properties of a session with the server.

Parameter:

Flags (INT): not currently supported

Return values:

Names (STRING): semicolon-separated properties which will be set

Return:

(INT): 0 = job successful, otherwise error code

krn.SessionPropertiesGet

Description:

This job returns the specified properties of a specified session.

Parameter:

Flags (INT): not currently supported

Properties (STRING): semicolon-separated properties which will be displayed (an output parameter is created for each property which is specified here)

SessionGUID (STRING): GUID of the session

Return values:

The output parameters are dictated by the input parameter Properties.

Return:

(INT): 0 = job successful, otherwise error code

Note:

Properties of a session

§ Address: address of the application server

§ SessGUID: Session GUID

§ Statname: computer name of the workstation

§ instname: instance name

§ StatGUID: GUID of the workstation

§ UserGUID: GUID of the user

§ hasserveraccount: does the session have a server account?

§ loggedin: is a user logged in?

§ haschannel: does the session have a channel?

§ autologin: login through dialog or autologin

§ supervisor: name of the supervisor

§ LangID: ID of the language

See also:

krn.SessionPropertiesSet, krn.SessionPropertiesEnum

krn.SessionPropertiesSet

Description:

This job sets the properties of a session.

Parameter:

Flags (INT): not currently supported

Properties (STRING): semicolon-separated properties which will be set

[property name] (STRING): value of the property (an input parameter is generated for each property that is specified in the parameter properties)

Return:

(INT): 0 = job successful, otherwise error code

Note:

Properties of a session

§ Address: address of the application server

§ SessGUID: Session GUID

§ Statname: computer name of the workstation

§ instname: instance name

§ StatGUID: GUID of the workstation

§ UserGUID: GUID of the user

§ hasserveraccount: does the session have a server account?

§ loggedin: is a user logged in?

§ haschannel: does the session have a channel?

§ autologin: login through dialog or autologin

§ supervisor: name of the supervisor

§ LangID: ID of the language

See also:

krn.SessionPropertiesGet, krn.SessionPropertiesEnum

krn.UserSessionCreate

Description:

This job creates a user session for a B2B connection. A B2B connection is established between two servers (e.g. DRT server and Java server).

Parameter:

Flags (INT): not currently supported

Return values:

SessionGUID (STRING): ID of the session

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.UserSessionDelete

krn.UserSessionDelete

Description:

This job deletes a user session for a B2B connection. A B2B connection is established between two servers (e.g. DRT server and Java server).

Parameter:

Flags (INT): not currently supported

SessionGUID (STRING): ID of the session

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.UserSessionCreate

Engine administration

These jobs serve the engine administration.

§ krn.EnumJobs

§ krn.EnumNameSpaces

§ krn.GetNameSpaceParams

§ krn.LoadExecutor

§ krn.NameSpaceEnum

§ krn.NameSpaceGetInfo

§ krn.NameSpaceGetJobsInfo

§ krn.ReloadExecutor

§ krn.UnloadExecutor

krn.EnumJobs

Description:

The job returns a list of the implemented jobs for a specified namespace.

Parameter:

Flags (INT): not currently supported

NameSpace (STRING): short description of the namespace for which the list of jobs will be created

Return values:

[Job name] (STRING): name of the job implemented

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.EnumNameSpaces

krn.EnumNameSpaces

Description:

This job returns a list of the implemented namespaces.

Parameter:

Flags (INT): not currently supported

Return values:

NameSpace[1..n] (STRING): name of the namespace, sorted according to the alphabet

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.EnumJobs

krn.GetNameSpaceParams

Description:

The job returns the namespace parameters of a specified namespace.

Parameter:

Flags (INT): not currently supported

NameSpace (STRING): short description of the namespace

Return values:

Child (INT): 1 = the executor was started as a designated process, otherwise 0

ExecutorPresent (BOOL): 1 = executor is available, otherwise 0

Internal (BOOL): internal namespace

§ 1 = internal namespace (implemented in the kernel)

§ 0 = implemented in the executor

Queue (STRING): name of the queue for the namespace

State (INT): status of the namespace as a number

§ 0 = CREATED

§ 1 = LOADING

§ 2 = LOADED

§ 3 = UNLOADING

§ 4 = UNLOADED

StateText (STRING): status as text

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.EnumNameSpaces

krn.LoadExecutor

Description:

The job loads an executor.

Parameter:

Flags (INT): not currently supported

Name (STRING): short description of the namespace which will be loaded

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.ReloadExecutor

krn.NameSpaceEnum

Description:

This job returns a list of the implemented namespaces.

Parameter:

Flags (INT): not currently supported

Return values:

Namespaces (STRING): semicolon-separated short descriptions of all namespaces

Return:

(INT): 0 = job successful, otherwise error code

krn.NameSpaceGetInfo

Description:

This job returns all information on the specified namespace.

Parameter:

Flags (INT): not currently supported

NameSpaces (STRING): short description of the namespace

Return values:

NameSpaceInfo (STRING): MIME encoded buffer which contains information

NameSpaceInfoType (STRING): contains the description of the data returned in the parameter NameSpaceInfo

Return:

(INT): 0 = job successful, otherwise error code

krn.NameSpaceGetJobsInfo

Description:

This job provides information (name, number of job calls...) for the jobs of the specified namespace.

Parameter:

Flags (INT): not currently supported

NameSpaces (STRING): short description of the namespace

Return values:

NameSpaceInfo (STRING): MIME encoded buffer which contains information

NameSpaceInfoType (STRING): contains the description of the data returned in the parameter NameSpaceInfo

Return:

(INT): 0 = job successful, otherwise error code

krn.ReloadExecutor

Description:

The job reloads an executor.

Parameter:

Flags (INT): not currently supported

Name (STRING): namespace which will be reloaded

Return:

(INT): 0 = job successful, otherwise error code

krn.UnloadExecutor

Description:

This job deletes a specified namespace (executor) from the server.

Parameter:

Flags (INT): not currently supported

Name (STRING): namespace which will be deleted

Return:

(INT): 0 = job successful, otherwise error code

General Administration

§ krn.EnumModules

§ krn.JobThreadBreak

§ krn.JobThreadGetInfo

§ krn.QueueEnum

§ krn.QueueGetParams

§ krn.QueueGetStatistic

krn.EnumModules

Description:

The job returns a list of the loaded modules (libraries). System32 dlls can be hidden for this process.

Parameter:

Flags (INT): not currently supported

NoSystem32 (BOOL): hide System32 dlls (0 = no, 1 = yes)

Return values:

Module[1..n] (STRING): semicolon-separated information for the respective modules

§ Basename: module name without path information

§ FileName: module name with path information

§ Version: Module version

§ Create: time of file creation

§ Write: time of the last modification of the file

§ Access: time of the last access to the file

Return:

(INT): 0 = job successful, otherwise error code

krn.JobThreadBreak

Description:

This job interrupts the execution of the currently specified job. The job can only be used for correctly running jobs which support this job.

Parameter:

Flags (INT): not currently supported

ThreadID (INT): ID of the thread

JobNumber (INT): job number

Queue (STRING): queue name

Return:

(INT): 0 = job successful, otherwise error code

krn.JobThreadGetInfo

Description:

This job provides information on the thread.

Parameter:

Flags (INT): not currently supported

Return values:

Info (STRING): semicolon-separated MIME-encoded information on threads

Infotype (STRING): contains the description of the data returned in the parameter info

Return:

(INT): 0 = job successful, otherwise error code

krn.QueueEnum

Description:

This job creates a list with the names of existing queues. Typical queues are: common, dbpipe, ocr, workflow and redir.

Parameter:

Flags (INT): not currently supported

Return values:

Queue[1..n] (STRING): name of the respective queue

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.QueueGetParams, krn.QueueGetStatistic

krn.QueueGetParams

Description:

This job returns the parameters of a specified queue.

Parameter:

Flags (INT): not currently supported

Queue (STRING): queue name

Return values:

NumThreads (INT): number of threads

Priority (INT): Priority

§ 0 = low

§ 1 = normal

§ 2 = high

MaxQueueSize (INT): maximum number of jobs which can be handled

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.QueueEnum, krn.QueueGetStatistic

krn.QueueGetStatistic

Description:

This job provides static information for a specified queue.

Parameter:

Flags (INT): not currently supported

Queue (STRING): queue name

Return values:

JobsPosted (INT): number of sent jobs

JobsWaiting (INT): number of waiting jobs

LastPop (STRING): time of the last pop

LastPush (STRING): time of the last push

NameSpaces (STRING): semicolon-separated namespaces which use this queue

Threads (STRING): semicolon-separated threads of this queue

Return:

(INT): 0 = job successful, otherwise error code

See also:

krn.QueueEnum

Other jobs

- krn.CheckDiskSpace
- krn.GetFileVersionList
- krn.GetNextIndex
- krn.RunScript
- krn.SendAdminMail
- krn.SendMail
- krn.SendMessageToClients
- krn.ProcessGetInformation
- krn.GetCounter

krn.CheckDiskSpace

Description:

This job provides information on the capacity of the specified hard disk.

Parameter:

Flags (INT): not currently supported

Disk (STRING): name of the hard disk

- ROOT = determines the capacity of a drive on which the server directory is located
- DATA = determines the capacity of a drive on which the WORK server directory is located

§ LOG = determines the capacity of a drive on which the LOG server directory is located

MinSpace (INT): minimum free storage in MB (if this value is exceeded, the administrator is notified by e-mail; empty = MinSpace is read from the registry)

InformAdmin (BOOL): 1 = the administrator will be informed by e-mail, otherwise 0

Return values:

Total (INT): size of the hard disk in MB

Free (INT): free space in MB

Min (INT): minimum free space in MB

Return:

(INT): 0 = job successful, otherwise error code

krn.GetFileVersionList

Description:

This job returns a list of files of the queried directories including information on the time of creation as strings. For dll, ocx and exe files the version number is returned instead of the creation time.

Parameter:

Flags (INT): not currently supported

Directory (STRING): directory, the entry '.' corresponds to server\etc, '..' corresponds to server directory

Return values:

FileVersionList (STRING): string containing file names and creation time,
formatting: [file name]+[date time/version number]#
[file name 2]+[date time 2 version number2]#..

Return:

(INT): 0 = job successful, otherwise error code

krn.GetNextIndex

Description:

This job returns the next index from the database table 'osnextindex' for all DB entries which require a unique ID.

Parameter:

Flags (INT): not currently supported

Return values:

Index (INT): queried index

Return:

(INT): 0 = job successful, otherwise error code

krn.RunScript

Description:

This job executes a specified VB script Script text can be passed as parameter 'Script,' as parameter 'ScriptFile' or as file list. Script text is determined in this order.

Parameter:

Flags (INT): not currently supported

Script (STRING): VB script which will be executed

CtxName (STRING): name of the context, can be empty. The default name is used in such a case.

GUI (BOOL): specifies whether MsgBox can be called from the script. Whether MsgBox can really be called, depends on the server environment.

Eval (BOOL): defines whether Eval (if true) or Exec (if false) will be called.

Main (STRING, optional): name of the main function; default value "Main"

ScriptFile (STRING, optional): name of the file with script text

Return:

(INT): 0 = job successful, otherwise error code

Krn.EmptyJob

Description:

This job does nothing. It can be used to run Before and After Event scripts. The job has no specific parameters.

Return:

(INT): 0 = job successful, otherwise error code

krn.SendAdminMail

Description:

The job sends an e-mail to the preset administrator. The registry must therefore contain the registry entries Mailserver(SMTP IP address) and AdminMail(e-mail address)
under the key HKLM\\Software\\Optimal Systems\\[application_server_name]\\Schemata. The passed file list will then become a mail attachment.

Parameter:

Flags (INT): not currently supported

Sender (STRING): sender name

Subject (STRING): e-mail subject line

Text (STRING): text content of the e-mail

Return:

(INT): 0 = job successful, otherwise error code

See also:

Registry administration

krn.SendMail

Description:

This job sends an e-mail to a recipient which has been identified by an e-mail address. The registry must therefore contain the registry entry 'Mailserver' (SMTP IP address) under the key HKLM\\Software\\Optimal Systems\\[application_server_name]\\Schemata.

Parameter:

Flags (INT): not currently supported

Receiver: (STRING): E-mail address of the recipient

Sender (STRING): sender name

Subject (STRING): e-mail subject line

Text (STRING): text content of the e-mail

FileNamePrefix (STRING): (optional) prefix for names of files (attachments) that are sent with an e-mail. If a prefix was specified, a file `XYZ.abc` is renamed to `<prefix>.XYZ.abc`.

Return:

(INT): 0 = job successful, otherwise error code

See also:

Registry administration

krn.SendMessageToClients

Description:

This job sends a message to one or all connected clients.

Parameter:

Flags (INT): not currently supported

Computer (STRING): empty = notify all computers, otherwise a computer name

Instance (STRING): program name

User (STRING): name of the user receiving a notification

Message (STRING): type of message

Info (STRING): notification text (program notification)

Text (STRING): notification text for users (e.g. message box)

Return:

(INT): 0 = job successful, otherwise error code

krn.ProcessGetInformation

Description:

This job returns Windows performance counter with information on kernel processes. Information will be determined only from the server at which the own session is logged in.

Parameter:

Flags (INT): If 0 or 1, the information is returned in binary form (for Monitor and enaio® enterprise-manager). If 2, the information is returned in text form and logged in Channel 11.

Return:

(INT): 0 = job successful, otherwise error code

krn.GetCounter

Description:

This job administers counters in the oscounters table. It is possible to query the counter value and to reset or create the counter. When calling the job, the counter value will be increased unless the counter is to be reset. Resetting is carried out depending on the counter type. The time of the job call is saved on the server in order to determine if the counter is to be reset at the next call.

Counter has type:

0 = counter is reset manually by job parameter

1 = counter is reset daily

2 = counter is reset monthly

3 = counter is reset yearly

Counters are identified with GUID and type.

Parameter:

Flags (INT): not currently supported

CounterGUID (STRING): GUID of the counter

CounterType (INT): type of the counter

Reset (BOOLEAN): specifies if the counter will be reset in case the CounterType is 0.

Initial (INT): value with which the counter will be initialized if it does not exist yet or is reset

Return values:

Counter (INT): value of the counter

Return:

(INT): 0 = job successful, otherwise error code

License Core Services (Namespace lic)

This namespace is located besides the ADM executor and the KRN executor in the server kernel and includes all jobs which are responsible for the license management of the entire enaio® system. To use different functions a login for the respective licenses must be performed before execution.

In particular for the use of the standard, DMS and workflow engine the kernel checks if the called instance has been authorized and if a license has been used.

The license login is performed with the jobs LicLogin or LicLoginEx. The required license strings must be passed (in particular ASC, MWC) before the DMS and workflow engine can be used.

§ lic.CheckLicense

§ lic.LicCopyDefault

§ lic.LicFreeResource

§ lic.LicGetGlobalInfo

§ lic.LicGetGlobalInfoEx

§ lic.LicGetModuleInfo

§ lic.LicGetQueueStatus

§ lic.LicLogin

§ lic.LicLoginEx

§ lic.LicLogout

§ lic.LicLogoutEx

§ lic.LicResetData

lic.CheckLicense

Description:

This job checks if the queried modules are licensed for the workstation of the enaio® client. The license is not locked in the database.

Parameter:

Flags (INT): not currently supported

Modules (STRING): short descriptions of the modules separated by space characters

Return values:

Result (STRING): return codes for the queried modules separated by space characters

§ 0 = a license exists for the module

§ 602 = Error

Return:

(INT): 0 = job successful, otherwise error code

lic.LicCopyDefault

Description:

This job distributes all Named licenses, which are defined in the DB table 'oslicresources' for the Standard station, for all other stations. This happens during network setup, for example.

Parameter:

Flags (INT): not currently supported

Return:

(INT): 0 = job successful, otherwise error code

lic.LicFreeResource

Description:

This job releases a resource which can be found in the DB table 'osresources'. The resources are: modules (ADM, M_X, etc.) and important system files (aslisten.dat, .cfg files, background images, etc.).

Parameter:

Flags (INT): not currently supported

SessionGUID (STRING): the current SessionGUID

ResourceID (STRING): the ResourceID of the resource which will be released

Return:

(INT): 0 = job successful, otherwise error code

lic.LicGetGlobalInfo

Description:

This job returns the value of the specified parameter which is contained in license data (aslic.dat).

Parameter:

Flags (INT): not currently supported

Info (STRING): name of the queried parameter

- § IDENT = identification method
- § ADDRESS = GUID or IP address
- § CREATED = license creation date
- § CREATEDFROM = created by
- § CUSTOMER00 = Licensee
- § LASTMODIFIED = last modified
- § MODIFIEDFROM = modified by
- § SERVICENAME = service name
- § EXPIRES = valid until

Return values:

Result (STRING): value of the queried parameter

Return:

(INT): 0 = job successful, otherwise error code

See also:

lic.LicGetGlobalInfoEx

lic.LicGetGlobalInfoEx

Description:

This job returns the value of the specified parameter which is contained in license data (aslic.dat).

Parameter:

Flags (INT): not currently supported

Info (STRING): names of the queried parameters in the format ;Parameter;Parameter;...

- § empty = all parameters are returned
- § ? = all parameter names are returned

Return values:

[parameter name] (STRING): value of the queried parameter

[PARAM[000...nnn]]: Parameter name (only for Info = ?)

Return:

(INT): 0 = job successful, otherwise error code

lic.LicGetModuleInfo

Description:

This job returns information (type, max. number of users) for the license of the specified module.

Parameter:

Flags (INT): not currently supported

Module (STRING): short name of the module which will be queried

Return values:

Result (STRING): string which describes the license characteristics of the specified module

- MaxUseCount: max. number of clients which can use the module
- License type
  - N = the module can only by used by clients which work at the specific workstations
  - C = the module can be used by the clients of the respective workstations, but only by a limited number (MaxUseCount) of clients
- Number of configurations which can be created

Return:

(INT): 0 = job successful, otherwise error code

lic.LicGetQueueStatus

Description:

This job provides information on licenses for specified modules and system files which are currently issued by the server for the specified stations.

Parameter:

Flags (INT): not currently supported

Modules (STRING): short module names which are queried separated by space characters (empty = all modules and system files)

Stations (STRING): station names which are queried separated by space characters (empty = all stations)

Return values:

File list: name and path of the file; contains queried information on licenses (file format .rpt)

- Time of license issue (time stamp)
- Short name for module/system file
- License type (0 = module, 1 = system file)
- Workstation name
- User name

- Flags -> currently not used
- Parameters -> currently not used
- ID of the server group
- Session GUID
- ID of the module/system file

FileCount (INT): always 1

Return:

(INT): 0 = job successful, otherwise error code

lic.LicLogin

Description:

This job issues a license for the specified module. Before the client application can use certain enaio® modules, the server must grant permission for their use. The server checks (e.g. the number of licenses, access to the module only from specific workstations) if the client is authorized to use this module. If the check is successful, the server will lock the license in the database.

Parameter:

Flags (INT): not currently supported

Module (STRING): short description of the module

Return values:

Result (STRING): 0 = license granted, >0 = error

Return:

(INT): 0 = job successful, otherwise error code

See also:

lic.LicLoginEx, lic.LicLogout

lic.LicLoginEx

Description:

This job grants licenses for the specified modules. Before the client application can use certain enaio® modules, the server must grant permission for their use. The server checks (e.g. the number of licenses, access to the module only from specific workstations) if the client is authorized to use this module. If the check is successful, the server will lock the license in the database.

Parameter:

Flags (INT): not currently supported

Module (STRING): short descriptions of the modules separated by space characters

Return values:

Result (STRING): result notifications separated by space characters (0 = license granted, >0 = error)

Return:

(INT): 0 = job successful, otherwise error code

See also:

lic.LicLogoutEx

lic.LicLogout

Description:

This job releases a license for a specified module which was previously locked with LicLogin or LicLoginEx.

Parameter:

Flags (INT): not currently supported

Module (STRING): short description of the module

Return values:

Result (STRING): 0 = license released, >0 = error

Return:

(INT): 0 = job successful, otherwise error code

lic.LicLogoutEx

Description:

This job releases a license for multiple modules which were previously locked with LicLogin or LicLoginEx.

Parameter:

Flags (INT): not currently supported

Modules (STRING): short descriptions of the modules separated by space characters

Return values:

Result (STRING): result notifications separated by space characters (0 = license granted, >0 = error)

Return:

(INT): 0 = job successful, otherwise error code

lic.LicResetData

Description:

This job forces the server to reread the license information (aslic.dat) from the database (oslicense). The job is sent to the active servers of the server group by the server which has changed the license information. The servers which receive this information must modify their internal data structures which are related to the license system.

Parameter:

Flags (INT): not currently supported

Return:

(INT): 0 = job successful, otherwise error code

Data Transfer Services (Namespace dtr)

This namespace contains jobs for the execution of the data transfer server.

Important:

Since the data transfer server uses MS Office, the application server cannot run under the local system account, but must use a user account for login.

For the call on the server side of the data transfer server, 'German' is used as the default language. The language can be specified via a registry entry for the enaio® server:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\OPTIMAL SYSTEMS\MS Office
integration\OS:4.x-Office-Utilities\
```

Create the DWORD value 'Language' with the language-specific value (cf. 'Language codes').

§ dtr.SynchronizeData

Return:

(INT): 0 = job successful, otherwise error code

dtr.SynchronizeData

Description

Parameter:

Flags (INT): not currently supported

ObjectType (INT): Object type

ObjectID (INT): Object ID

[TemplateAlias] (String): name of the template which will be filled with the existing index data. Instead of a template name a template file can be passed using a file list.

[File list]: Name and path of the template file. Alternative to the parameter template

Return:

[File list]: Name and path of the filled document

OxSvrSpt

General Description

The `OxSvrSpt.dll` library allows access to server functionalities through a Microsoft COM interface. The main focus is on convenient manageability by the user and by VB as well as script compliant COM interfaces. An easy-to-use approach is followed that allows the execution of simple calls through fewer lines of code.

The following properties are supported:

§ Error handling using COM errors (IErrorInfo)

§ All errors that occurred will be returned as COM errors. In case a call of a method returns multiple errors (e.g. server job call), the first error will be returned as a COM error and all other errors are available in a collection.

§ Transparent use of the Base64 parameter

§ Processing of the Base64 server parameter is done by OxSvrSpt completely. Stream and XML access methods are made available to the user for this purpose.

§ Automatic encoding and decoding of the binary parameter (BASE64)

§ Data transferred to the OxSvrSpt library by the user will be encoded automatically for the transfer to the server in MIME-BASE64 code and when returned from the server the data will be decoded. Users no longer come into contact with encoding.

§ Encoding and decoding of XML data through the library

§ With the XML properties of the parameter and file parameter interfaces, it is possible to read and set the basic string representation (BSTR) of the XML data. In doing so, the OxSvrSpt library applies the correct encoding and decoding to the respective binary representation (UTF-8, UTF-16, ...). Furthermore, the stream methods of these interfaces allows further processing of data directly using an IStream interface. That way it is possible, for example, to load the parameters directly into a MSXML-DOM document or transfer them from such a document.

§ Automatic password encoding through the module

§ During login it is possible to transfer the password in encoded or decoded form. The required conversion for the transfer to the server is done by the OxSvrSpt library.

§ Collections that can be used with ForEach

§ All COM collections can be used with the ForEach statement. As a result, it is possible to iterate over these statements in Visual Basic, scripting environments and .NET.

§ Correct processing of collections and parameters in the debugger

§ It is ensured that used stream access will not be disturbed in debug environments that can access the properties of COM objects directly and display them (e.g. Visual Basic). Collections are suitable for VB, resulting in the possibility to view elements in the VB debugger.

§ Return parameters are generated by the called property or method.

- All return parameters are made available in such a way that they are generated using a method of the OxSvrSpt library. It is not necessary to transfer a variable for filling from a COM script environment to the library. The server object is, with the exception of the support object, the only object that can be generated. All other interfaces are made available by this object or subordinate interfaces.
- Convenient processing of notifications
- For the processing of notifications an event interface is available that processes in and out parameters, comparable to the job interface.
- Methods for reading and writing of ASCII data from streams
- If an attempt is being made to serialize basic strings from within Visual Basic or COM based script environments, data will be processed as wide chars. The provided method allows to read and write data that is not XML from Base64 parameters as ASCII.

Modules

Integration of Library

In this section, it is described how to use the `OxSvrSpt` library in different programming languages. Depending on the used programming language and the development environment, there are different possibilities to access the library.

Visual Basic

In order to use `OxSvrSpt` with the type library, the enaio® server library must be integrated using the "Links" menu of the development environment.

This can be done with the following code:

```
Private m_oServer As new OxSvrSpt.Server
```

The component can also be integrated through late binding. The source code corresponds to the VBScript code.

Visual Basic Script

In VBScript, the server object can be generated as follows:

```
Dim oServer
Set oServer = CreateObject( "OxSvrSpt.Server" )
```

Visual C++

It is recommended to use the `import` directive for the integration of the `OxSvrSpt` library in Visual C++. The following part of source code shows the import. It should be done at a central point, e.g. the `StdAfx.h`.

```
// warning C4192: automatic exclusion of 'IStream'
// when importing the type library 'OxSvrSpt.tlb'
#pragma warning ( disable: 4192 )
#import "OxSvrSpt.dll" raw_method_prefix("raw_") rename_namespace( "OxSvrSpt" )
#pragma warning ( default: 4192 )
```

Instead of the `raw` methods, the `high` methods should be used. In order to avoid confusion, `raw` functions have the prefix `raw_` in the previously mentioned `import` directive. COM errors are automatically mapped for `_com_error` exception handling by `high` methods. Additionally, it is ensured that transferred basic strings have the correct type. The following part of source code shows how to use a wide char string instead of a basic string as transfer parameter. Since the access to a basic string in C++ corresponds to the access to a `WIDECHAR` string, it can be compiled and executed as long as the COM client and the COM server are in the same apartment.

But it must not be assumed that this is the case!

Such a call can result in errors that are difficult to locate.

```
HRESULT Test( BSTR Message );
...
HRESULT hr = Test( L"something" );
```

But if the `high` method is used instead of the `raw` method, the `_bstr_t` class ensures correct processing.

```
HRESULT Test(_bstr_t Message );
...
HRESULT hr = Test( L"something" );
```

Note:

When implementing `high` methods without `retval` return value, the `import` directive declares these methods as `HRESULT` instead of `void`. However, `HRESULT` never sends an error value, because in case of an error the `_com_error` exception will be thrown. The structure of these methods is very similar to the structure of `raw` features. Also for this reason, the `raw_method_prefix` should be used in order to avoid confusion.

The creation of the object should be done with the `IServerPtr` constructor and not with the `CreateInstance` method, because with the latter no `_com_error` exception is thrown. When using the second option, it is necessary to evaluate the `HRESULT` value individually to receive an appropriate error message.

```
try
{
OxSvrSpt::IServerPtr spServer( __uuidof( OxSvrSpt::Server ));
}
catch( _com_error& ex )
{
// error handling takes place here
}
```

Alternatively, the creation of the `IServer` object can be done with the name of the coclass. This, in turn, corresponds to the late binding when creating an object in Visual Basic.

```
try
{
OxSvrSpt::IServerPtr spServer( "OxSvrSpt::Server" ));
}
catch( _com_error& ex )
{
// error handling takes place here
}
```

Microsoft C#

In order to use the component, the enaio® server access library link must be added in Microsoft Visual Studio. To do so, click on **Project**>**Add reference**>**COM**. Then, the `OxSvrSpt` namespace can be integrated through the `using` directive.

```
Server server = new Server();
```

Registration

This section describes how to use the `ISession` object in different programming languages. Login procedures will be presented with which you can log in to an enaio® server using an unencrypted or encrypted password, a session GUID and the NT user name.

Login Options

In order to log in to the system, an object of the `IServer` class must be created. This object provides methods, such as `Login()`, that you can use to log in. In case of a successful authentication, the methods return an `ISession` object. If the authentication failed, a COM exception is thrown. If an empty string is passed for the parameters `User name` and `Password`, the library tries to log in with the NT user data. Logging in automatically is possible, if this option was activated in enaio® administrator. The NTLM authentication option is not yet implemented.

- The `Login()` method is used for logging in the specified user to the specified server.
- The `LoginGUID()` method is used for logging in to an existing `SessionGUID`.
- The `LoginBalanced()` method is used for logging in to a server group. Every possible server entry in the list consists of a server name, the port and a weighting. The weighting indicates the possibility to establish a connection to the respective server. The sum of all servers' weightings must not be greater than 100.
- The `OpenSession()` method is used for logging in to an existing `DefaultSession`. `DefaultSessions` can be created with the methods mentioned before by setting the `DefaultSession` parameter to `true`.

Visual Basic and VBScript

The parameters `User name`, `Password`, `Server`, and `Port` are passed to the `Login()` method. In this case, the parameters `PasswordType` and `DefaultSession` are set to `false` automatically.

```
Dim session As session
' User login on a specified server
Set session = server.Login("root", "optimal", "localhost", "4000")
' Login using an existing SessionGUID
Set session = server.LoginGUID("D57D21256EFB4C91B79EDD5A4928400B", "localhost",
"4000")
' User login to a group of servers
Set session = server.LoginBalanced("root", "optimal",
"localhost#4000#90;10.1.3.100#4600#10")
```

Visual C++

In C++, the parameters `PasswordType` and `DefaultSession` must not be omitted, because C++ does not support default values. The following examples show the login process with an encrypted password.

```
// user login on a specified server
OxSvrSpt::ISessionPtr spSession = spServer->Login("root",
"HB01601611651521561421550000000000000000000000000", "localhost",
"4000", OxSvrSpt::PasswortTypeEnum::pwEncrypted, false );
// login using an existing SessionGUID
OxSvrSpt::ISessionPtr spSession = spServer-
>LoginGUID("D57D21256EFB4C91B79EDD5A4928400B",
"localhost", "4000", false );
// user login to a group of servers
OxSvrSpt::ISessionPtr spSession = spServer->LoginBalanced("root",
"HB01601611651521561421550000000000000000000000000",
"localhost#4000#90;10.1.3.100#4600#10",
OxSvrSpt::PasswortTypeEnum::pwEncrypted, false );
// create a DefaultSession (parameter DefaultSession = true)
OxSvrSpt::ISessionPtr spDefaultSession = spServer->Login("root",
"HB01601611651521561421550000000000000000000000000", "localhost",
"4000", OxSvrSpt::PasswortTypeEnum::pwEncrypted, true );
// log in to a DefaultSession
OxSvrSpt::ISessionPtr spSession = spServer->OpenSession(
(bstr_t)spDefaultSession->Properties->Item["SessionGUID"]->Value,
OxSvrSpt.OpenSession");
```

Visual C#

In C#, the parameters `PasswordType` and `DefaultSession` must not be omitted. In this example, the NT user name is used for the login which is why empty strings are passed for the parameters `User name` and `Password`.

```
// user login on a specified server
Session session = server.Login("","","localhost," "4000",
PasswortTypeEnum.pwNotEncrypted, false);
// login using an existing SessionGUID
Session session = server.LoginGUID("D57D21256EFB4C91B79EDD5A4928400B",
"localhost", "4000", false);
// user login to a group of servers
Session session = server.LoginBalanced("", "",
"localhost#4000#90;10.1.3.100#4600#10",
PasswortTypeEnum.pwNotEncrypted, false);
// create a DefaultSession (parameter DefaultSession = true)
Session defaultSession = server.Login("root","optimal","localhost", "4000",
PasswordTypeEnum.pwNotEncrypted, true);
// log in to a DefaultSession
Session session =
server.OpenSession(defaultSession.Properties["SessionGUID"].Value.ToString(),
"OxSvrSpt.OpenSession");
```

License Management

For validating and using licenses, the `ISession` collection is available in the `ILicenses` object. This collection can be used to register, deregister and validate the licenses to be used at the server. All registered licenses are kept in the collection.

```
/*
JavaScript version
Unlike the VBS version, this permits the use of
COM exceptions. If an error occurs, the application flow is
canceled and the system jumps to error handling.
*/
try
{
var oServer, oSession, oJob;
// create access object and log in
oServer = new ActiveXObject( "OxSvrSpt.Server" );
oSession = oServer.Login( "root", "optimal", "localhost", "4000" );
// add and delete licenses
oSession.Licenses.Add( "ASC" );
oSession.Licenses.Delete( "ASC" );
oSession.Licenses.Add( "ASC" );
// worked:o)
WScript.Echo( "ok" );
}
catch( ex )
{
// output errors occurred
WScript.Echo( ex.description );
}
```

Server Events

Description

You have the option to receive information about certain events through `notifications` sent by the server.

VB

In Visual Basic, `notifications` are available using the event mechanism, as described in the following example.

```
Private WithEvents m_oSession As OxSvrSpt.Session
Private m_oServer As OxSvrSpt.Server
public sub Start()
m_oServer.Properties("NotifyNeeded") = True
Set m_oSession = m_oServer.Login( , , "localhost", "4000")
end sub
Private Sub m_oSession_Notify(Job As OxSvrSpt.INotifyJob)
On Error GoTo ErrTrap
Dim strFileXML As String
strFileXML = Job.InputFileParameters(1).XML
' Return parameter
Job.OutputParameters.AddNewIntegerParameter "test", 100
Job.OutputParameters.AddNewStringParameter "meier", "huhu"
Exit Sub
ErrTrap:
MsgBox Err.Description
End Sub
```

VBScript

If you want to use `notifications` in a scripting-host environment (VBScript as vbs), you have to generate the instance using the `CreateObject()` method of the `WScript` environment.

```
Dim oServer
Set oServer  = WScript.CreateObject( "OxSvrSpt.Server", "oServer_" )
```

If the `IServer` object is created using the `WScript` host, it is possible to specify a prefix for event functions. With this mechanism, `notifications` can be intercepted.

Furthermore, the `callback` functionalities of the `OxSvrCom` library are available through the `CreateJobSink()` method and the `CreateJobSink()` method of the `ISession` interface.

```
Dim oServer, oSession
Set oServer  = WScript.CreateObject( "OxSvrSpt.Server", "oServer_" )
' Notifications will be used
oServer.Properties("NotifyNeeded") = True
' Use auto login
set oSession = oServer.Login( , , "localhost", "4000" )
...
'
' This function is called by the server object if
' a notification appears within the specified rest period
'
' @param oJob
'           contains the data for the notification call.
'           This corresponds to the job object for the session. The only
'           difference is that instead of the input parameters
'           the output parameters contain the methods for creating
'           parameters.
'
Sub oServer_Notify( oJob )
Dim strText
' Output the notification name
strText = "Name: " + oJob.Name + vbCrLf
' Output all input parameters
strText = strText + "Parameter:" + vbCrLf
Dim oParameter
For Each oParameter In oJob.InputParameters
strText = strText + "   " + oParameter.Name + " - " + CStr( oParameter.Value ) +
vbCrLf
next
' Write result in a text box
MsgBox strText
End sub
```

XML Processing

Depending on the used XML encoding, XML data can be different in their binary appearance.

Example for determining the object definition

JavaScript version

```
/*
JavaScript version
Unlike the VBS version, this permits the use of
COM exceptions. If an error occurs, the application flow is
canceled and the system jumps to error handling.
*/
try
{
var oServer, oSession, oJob;
// create access object and log in
oServer = new ActiveXObject( "OxSvrSpt.Server" );
oSession = oServer.Login( "root", "optimal", "localhost", "4000" );
// create job for the specification of the object definition
oJob     = oSession.NewJob( "dms.GetObjDef" );
// set query parameter
oJob.InputParameters.AddNewIntegerParameter( "Flags", 0 );
// execute job
oJob.Execute();
// determine XML from the output file
// During reading the XML character encoding
// is taken into account.
// After the end of this call the transferred
// file is automatically deleted.
WScript.Echo( oJob.OutputFileParameters(1).XML );
}
catch( ex )
{
// output errors occurred
WScript.Echo( ex.description );
}
```

VBScript version

```
Option Explicit
Dim oServer, oSession, oJob,o
Set oServer  = CreateObject( "OxSvrSpt.Server" )
set oSession = oServer.Login( "root", "optimal", "localhost", "4000" )
set oJob     = oSession.NewJob( "dms.GetObjDef" )
oJob.InputParameters.AddNewIntegerParameter "Flags", 0
oJob.Execute
msgbox oJob.OutputFileParameters(1).XML,,"Object definition"
```

Processing of Binary Data

Different options are available for the processing of binary data in the input and output parameters. Both the `IParameter` and `IFileParameter` objects provide two different methods.

§ 1. Access via chunks (byte arrays)

§ 2. Access via stream (`IStream`)

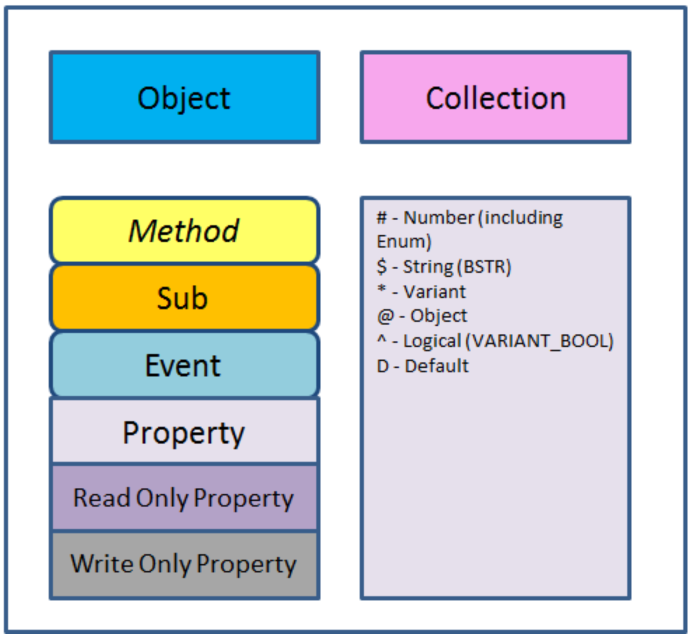
```
'
' Example script for byte-by-byte reading of binary data using the
' GetChunk method of a parameter.
'
Option Explicit
Dim oServer, oSession, oJob, oFileParameter
Set oServer  = CreateObject("OxSvrSpt.Server")
set oSession = oServer.Login("root", "optimal", "localhost", 4000)
Set oJob     = oSession.NewJob("dummy")
Set oFileParameter = oJob.InputFileParameters.AddTempFile()
oFileParameter.xml = "<?xml version='1.0' encoding='utf-16' ?><abc>äöü</abc>"
' Set stream to read at the starting position
oFileParameter.ResetStream
Dim abReadData
abReadData = oFileParameter.GetChunk(oFileParameter.ActualSize)
Dim cPos
For cPos = LBound(abReadData) + 1 To UBound(abReadData)
Dim bData
bData = Ascb(Midb(abReadData, cPos, 1))
WScript.Echo(cPos & ":" & bData)
next
WScript.Echo("complete")
```

Error Handling

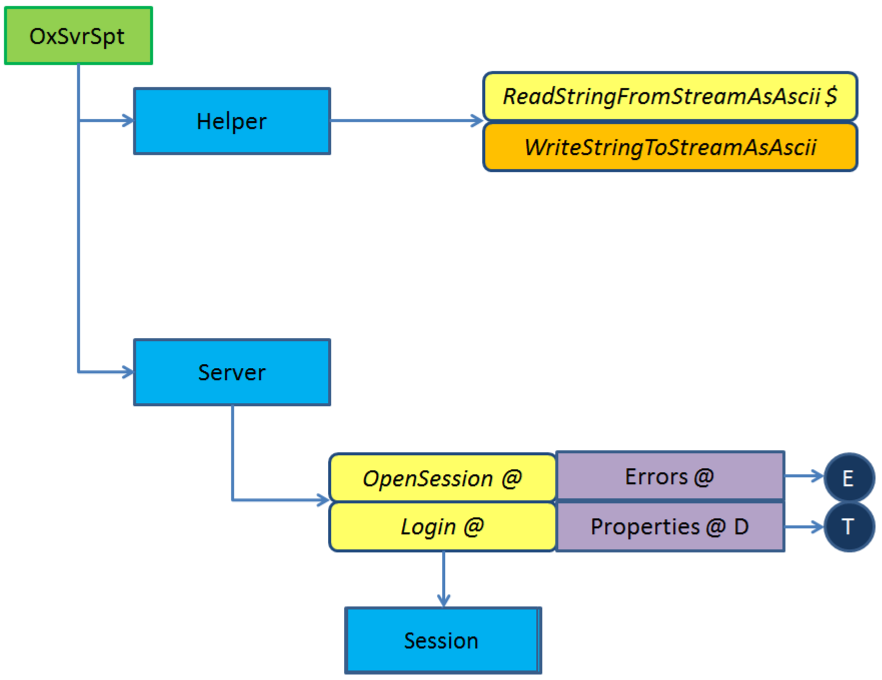
In any case of error, the `OxSvrSpt` library sends a COM error. In addition, errors of the `IServer` and `IJob` objects are also recorded in the `IError` collection. In both cases, it is possible that more than one error is returned. The COM error always corresponds to the first error in the `IError` collection.

Schema of the Structure

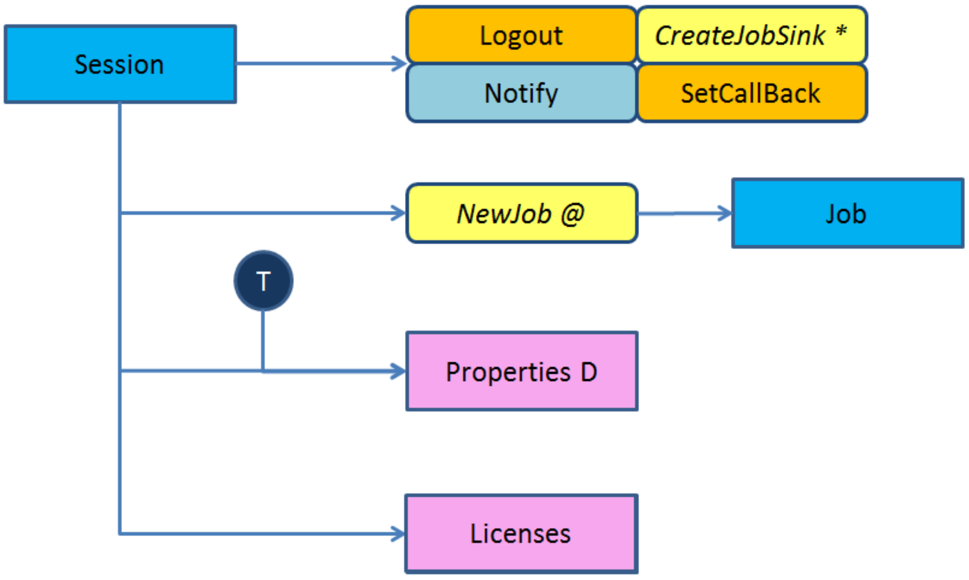
The following illustrations show the structure of OxSvrSpt:



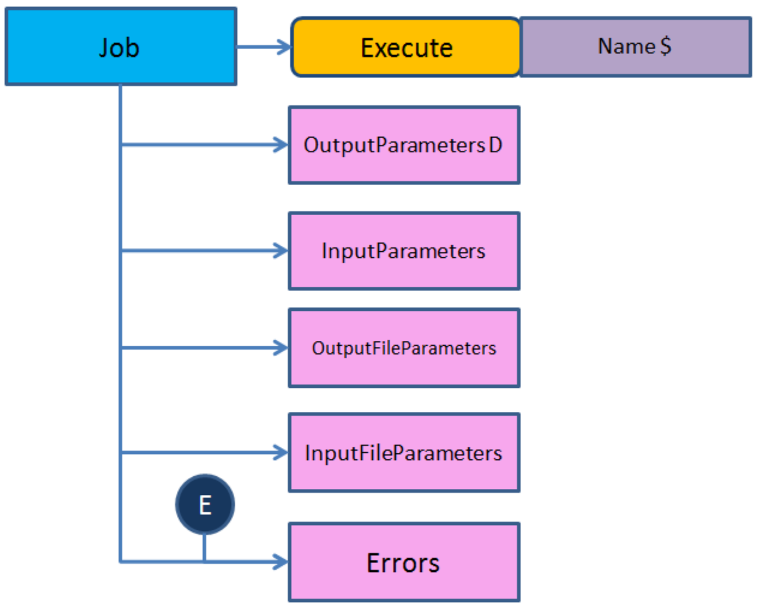
Legend of the schema



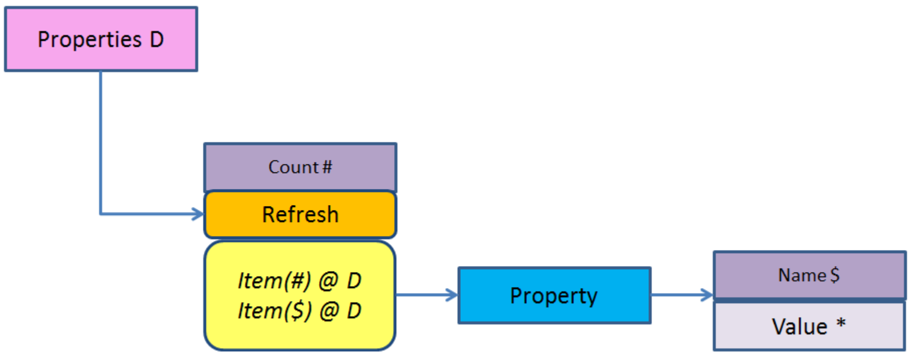
`OxSvrSpt` library



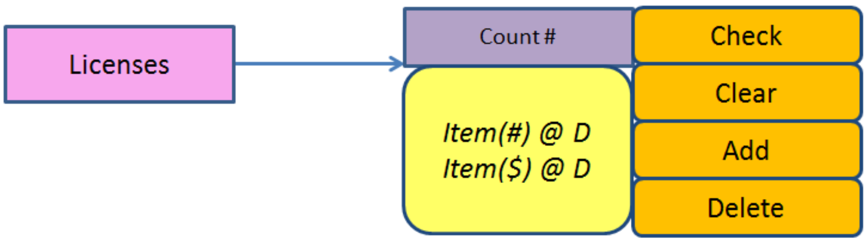
`Session` object



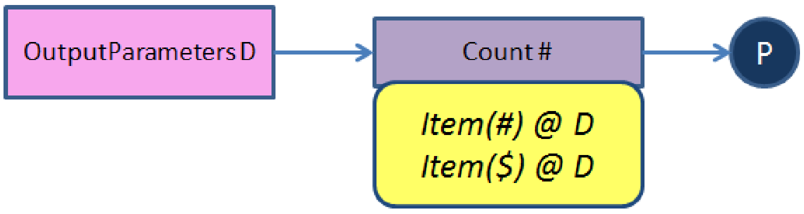
Job object



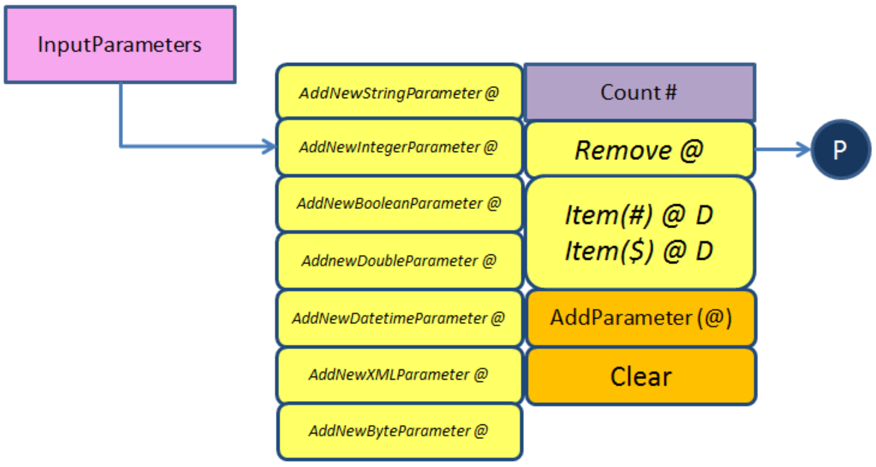
`Properties D` collection



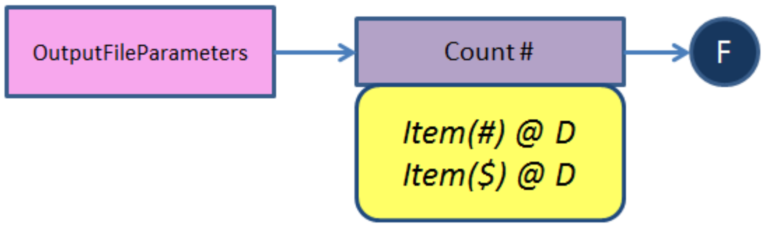
`Licenses` collection



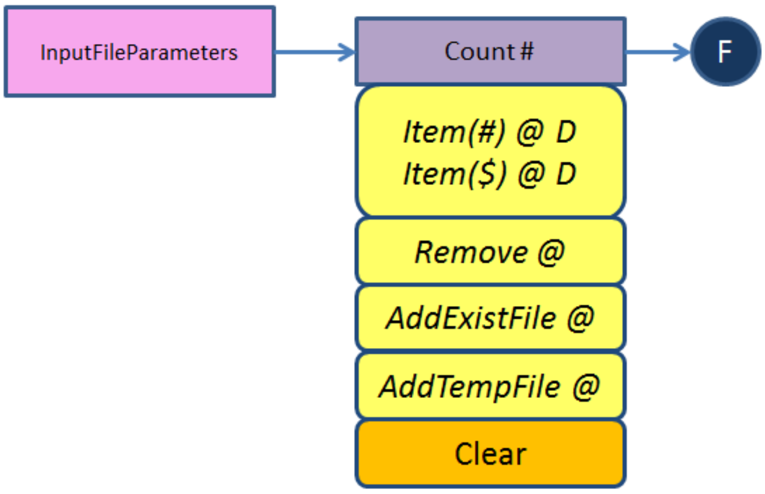
`OutputParameters D` collection



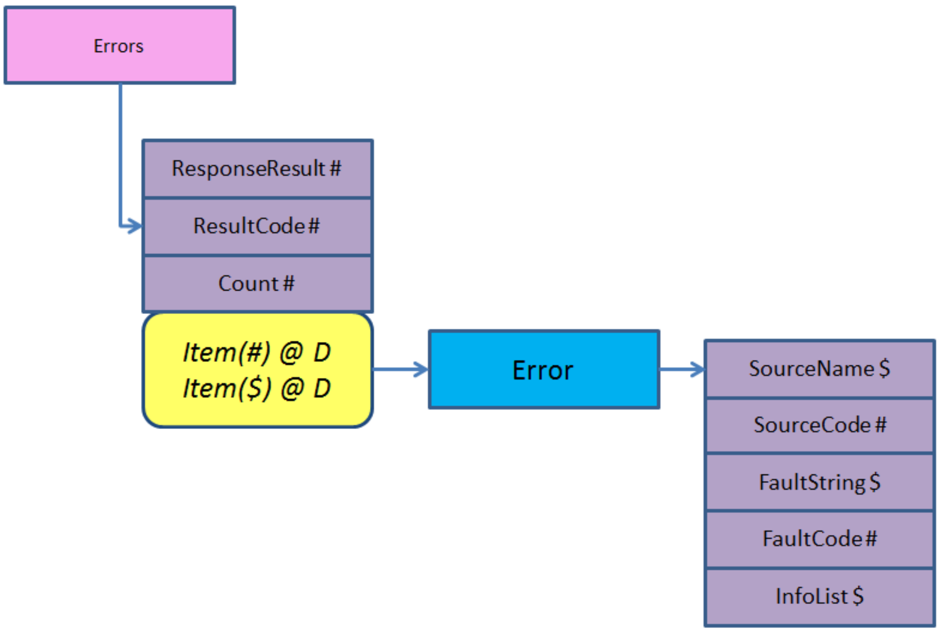
`InputParameters` collection



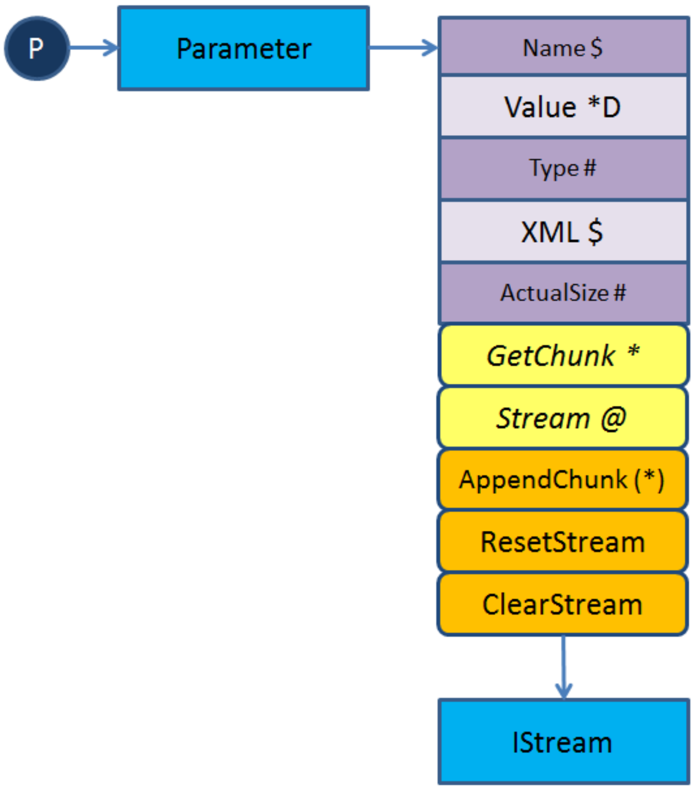
`OutputFileParameters` collection



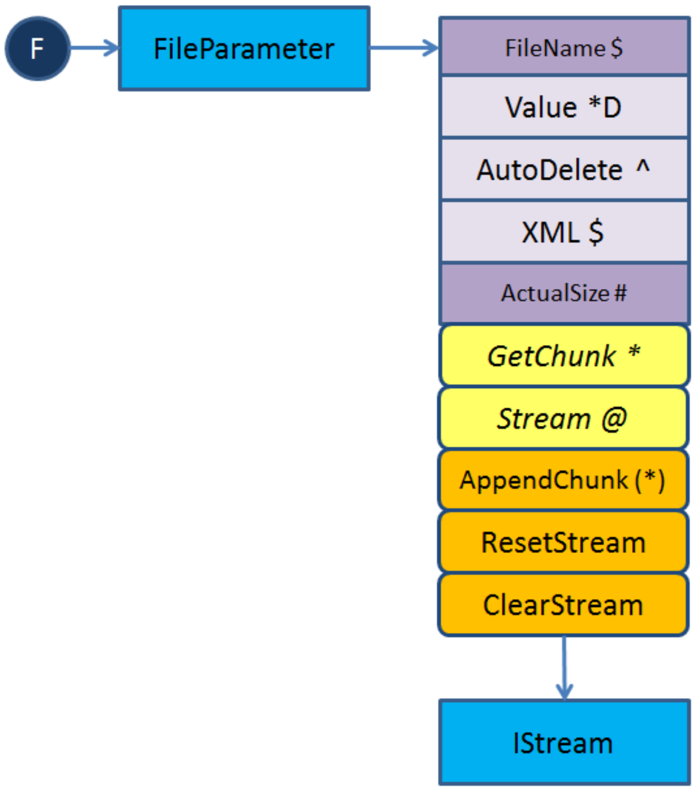
`InputFileParameters` collection



`Errors` collection



`Parameter` object



`FileParameter` object

Adding Watermarks to PDF Documents

As an alternative to adding watermark labels to PDF documents through settings in enaio® administrator, there is the possibility to add watermarks using extended functions. To do so, select PDF as the target format and set the value of `job` parameter 'Watermark' to 1. The following parameters consist of a prefix and a postfix, so the parameter `'HeaderText'` consists of the prefix `'Header'` and the postfix `'Text'`. The parameters are described in detail below:

Four areas for text entries can be defined with the following prefixes:
'Header', 'Footer', 'Side', and 'Center'.

The following parameter postfixes are possible where one of the above-mentioned prefixes must be used. All parameter postfixes are optional; however, at least one `'*Text'` parameter must be specified so that the extended watermark is added, otherwise the enaio® administrator settings will be applied. The postfixes are listed below:

Text (STRING):

The text which will be applied. If no text is specified, all other specifications for the respective watermark type will be ignored and no text will be applied.

The following replacement variables are possible for the specified text:

Variable	**Description**
`#TIME#`	current time
`#DATE#`	current date

`#USER#`	User name
`#FULL_USER#`	full user name
`#COMPUTER-IP#`	IP address
`#COMPUTER-GUID#`	GUID of the computer
`#COMPUTER-NAME#`	Computer name

TextColor (INT):

0 – 7 (0 is the default value) the following applies:

Value	**Description**
0	Black
1	White
2	Yellow
3	Red
4	Green
5	Magenta
6	Cyan
7	Blue

Alternatively:

TextColorRGB (STRING): 0-255,0-255,0-255 (RGB values for a color, separated by commas)

Font (STRING): Helvetica…, Times…, or Courier… (Helvetica is default)

FontBold (INT): 0 or 1 (1 -> bold, 0 -> default)

FontItalic (INT): 0 or 1 (1 -> italic, 0 -> default)

FontSize (INT): font size in points (10 is default)

With the following parameters, the text position can be specified:

Position (INT): 0-8 (0 is default) 0 for 'Header', 1 for 'Footer', 4 for 'Side' and 8 for 'Center'

Value	**Description**
0	Top left

1	Bottom left
2	Top right
3	Bottom right
4	Left centered
5	Right centered
6	Top centered
7	Bottom centered
8	Centered centered

OffsetX (INT): (0 is default) offset in mm from the left page margin.

OffsetY (INT): (0 is default) offset in mm from the top margin.

PlaceType (INT): 0-2 (0 is default) 0 -> all pages, 1 -> odd pages, 2-> even pages

FillStyle (INT): 0-2 (0 is default) 0 -> filled, 1 -> shadow outlines, 2 -> shadow fill

Angle (INT): 0-360 (0 is default) angle in degrees

An example:

```
Set oServerJob = o.CreateServerJob("cnv.ConvertDocument")
oServerJob.AddFile "myfile" (File that will be converted)
oServerJob.AddInputParameter "DestinationFormat", "pdf", 1
oServerJob.AddInputParameter "HeaderText", "Header", 1
oServerJob.AddInputParameter "HeaderFont", "Courier", 1
oServerJob.AddInputParameter "HeaderFontSize", "10", 2
oServerJob.AddInputParameter "HeaderTextColor", "3", 2
oServerJob.AddInputParameter "FooterText", " Created by #USER#, #DATE#, #TIME#",
1
oServerJob.AddInputParameter "FooterFontSize", "14", 2
oServerJob.AddInputParameter "FooterFontBold", "1", 2
oServerJob.AddInputParameter "FooterFontItalic", "1", 2
oServerJob.AddInputParameter "FooterTextColorRGB", "255,0,0", 1
```

- Note: Instead of the prefixes 'Header' and 'Footer', you can also use 'Side' or 'Center'.

Example for positioning with the 'Center' type:

```
oServerJob.AddInputParameter "CenterAngel", "45", 2
oServerJob.AddInputParameter "CenterOffsetX", "-10", 2
oServerJob.AddInputParameter "CenterOffsetY", "10", 2
oServerJob.AddInputParameter "CenterPlaceType", "1",  1
oServerJob.AddInputParameter "CenterFillStyle", "2", 1
```

- Note: Please note case sensitivity. (`Job` parameters are generally case sensitive).

Data Structures

_INotificationEvents

Description:

`_INotificationEvent` is an event interface for handling `notifications`.

`import "OxSvrSpt.idl"`

Public methods:

`void Notify([in, out] INotifyJob **Job)`

Parameter:

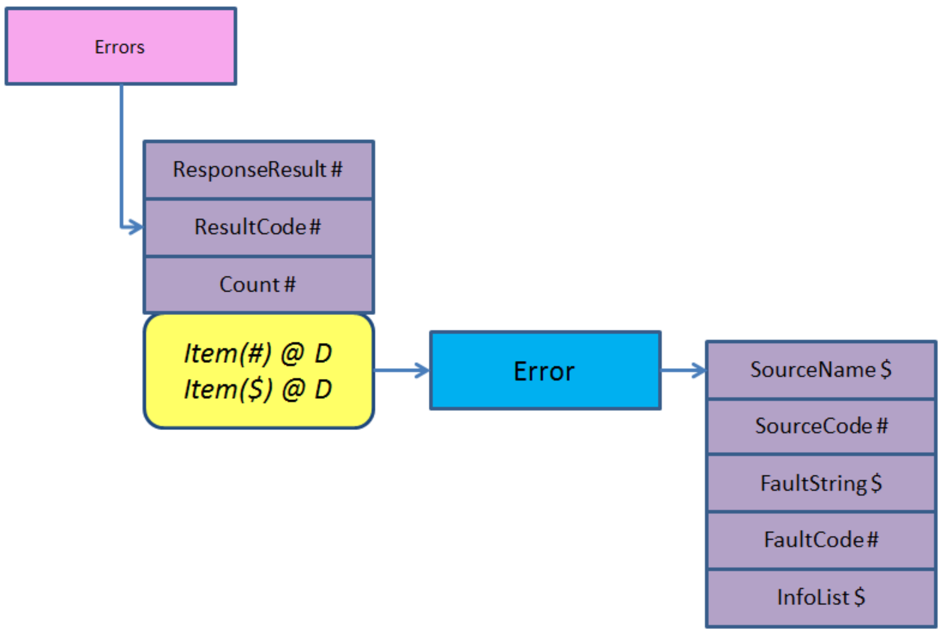
[in, out]: `Job` interface for accessing the input and output parameters of the notification. This interface is similar to the `IJob` interface.

IError

Description:

`IError` represents an error message sent by the server.

`import "OxSvrSpt.idl"`



Properties:

`long FaultCode [get]`

`BSTR SourceName [get]`

`long SourceCode [get]`

`BSTR FaultString [get]`

`BSTR InfoList [get]`

Documentation of properties:

§ `long FaultCode [get]`

`FaultCode` returns the error code.

Parameter:

[out]: `pVal` (VB return value) error description

§ `BSTR SourceName [get]`

`SourceName` returns the name of the source in which the error occurred.

Parameter:

[out]: `pVal`	(VB return value) identifier of the source

§ `long SourceCode [get]`

`SourceCode` returns the source code line in which the error occurred.

Parameter:

[out]: `pVal`	(VB return value) identifier of the source

§ `BSTR FaultString [get]`

`FaultString` returns the error description.

Parameter:

[out]: `pVal`	(VB return value) error description

§ `BSTR InfoList [get]`

`InfoList` see documentation of `OxSvrCom`.

Parameter:

[out]: `pVal`	(VB return value) InfoList

IErrors

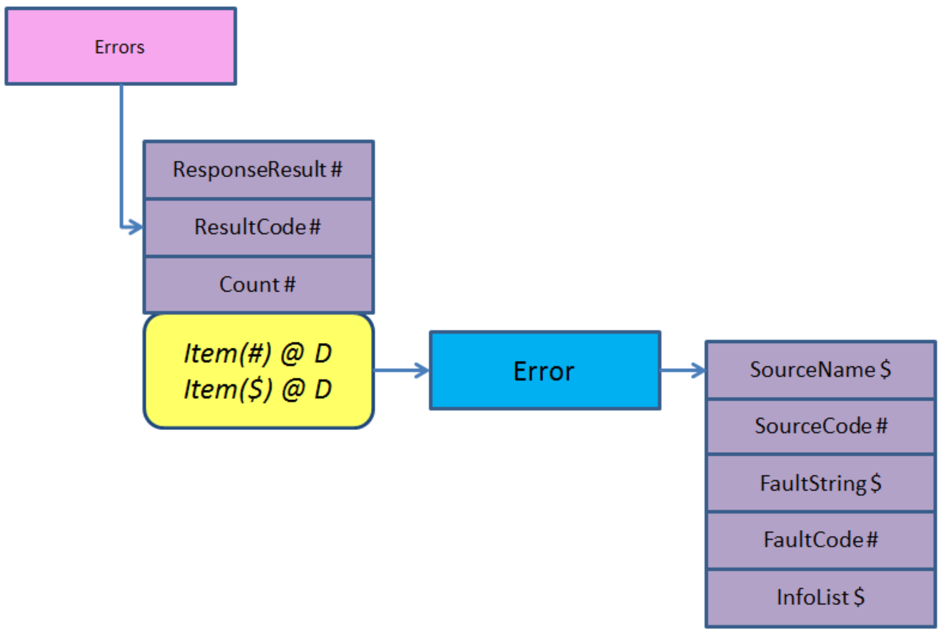
Description:

`IErrors` is a collection in which errors are collected that were sent by the server during an operation.

```
import "OxSvrSpt.idl"
```

Note:

For the individual entries, the name "`errorX`" is used as a keyword, X being the position of the error in the collection. This keyword is used in very rare cases. Instead, the collection should be queried either using the `ForEach` statement or through the position.



Properties:

`long Count [get]`

`long ResponseResult [get]`

`IError Item([in] VARIANT Index) [get]`

Documentation of properties:

§ `long Count [get]`

`Count` returns the number of items of the collection.

Parameter:

[out]: `plNumber` (VB return value) Number of elements in the collection.

§ `IError Item([in] VARIANT Index) [get]`

`Item` returns the specified item of the collection using the key or its position.

If a position is specified outside of the valid index, an error with the error value `errCollectionIndexOutOfRange` is returned. If the item cannot be found, an error with the error value `errCollectionItemNotFound` is returned.

Parameter:

[in]: `Index` Position and name of the requested element

[out]: `pptItem`associated `JobError` object

§ `long ResponseResult [get]`

`ResponseResult` returns the return value of the server call which caused the error.

Parameter:

[out]: `pVal` (VB return value) return value of the server call.

§ `long ResultCode [get]`

`ResultCode` returns the last error value of the server call.

Parameter:

[out]: `pVal` (VB return value) error value for the server.

IFileParameter

Description:

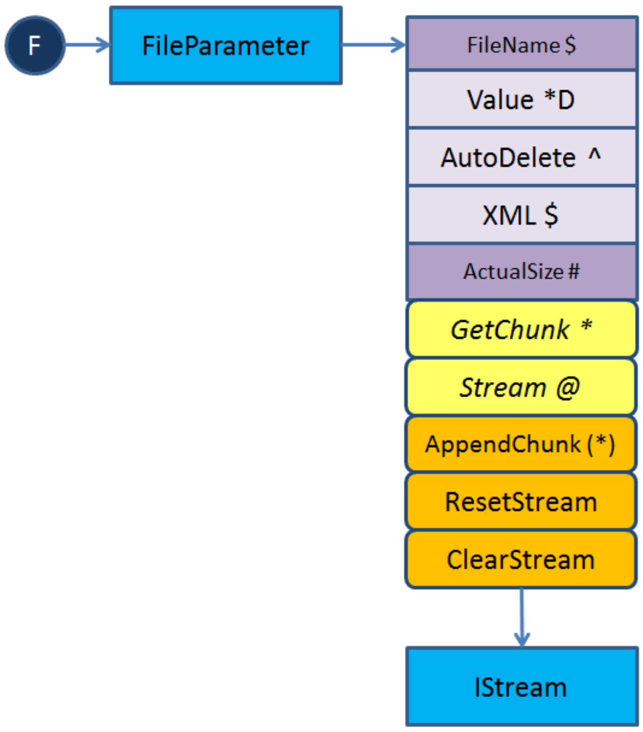
`IFileParameter` represents both an input parameter and an output parameter of a `job` that contains a file.

`import "OxSvrSpt.idl"`

The `IFileParameter` interface provides methods for accessing the `FileParameter` and the related file.

For accessing the file, similar to the `IParameter` interface there are stream, chunk, and XML functionalities.

As long as no file access functionalities are invoked, the file will not be opened. When accessing a method for the first time that operates with a file, it will be opened in the `modeReadWrite|shareDenyNone` mode. With `UnloadStream` it is possible to close this method explicitly in order to enable exclusive access for other processes.



Public methods:

```
HRESULT Stream ([out, retval] IStream ** ppStream)
HRESULT AppendChunk ([in]VARIANT Data)
HRESULT GetChunk ([in] long Length, [out, retval] VARIANT * pResult)
HRESULT ResetStream ()
HRESULT ClearStream ()
```

`HRESULT UnloadStream ([in, defaultvalue(-1)] VAIRANT_BOOL AutoReload)`

Properties:

`BSTR FileName [get, set]`

`long ActualSize [get]`

`BSTR XML [get, set]`

`VARIANT_BOOL AutoDelete [get, set]`

Documentation of the element functions:

§ `HRESULT AppendChunk ([in]VARIANT Data)`

`AppendChunk` appends additional bytes to the stream.

When accessing the stream functionalities of the interface for the first time, the related file is opened and kept open. If another caller subsequently requires exclusive access to this file, it can be released by invoking the `UnloadStream` method.

After invoking this method, the position pointer of the internal stream is at the end of the stream.

Parameter:

[in]: `Data`	contains the data that will be appended to the stream.

The data will be converted to `VT_ARRAY|VT_UI1` and further processed using the `OLE32 ChangeType` method. If, for example, a `BSTR` is passed, the data will be processed as `WIDECHAR`. Writing 8-bit characters into the stream can be done with the methods of the `Helper-COM` object.

Example:

```
Dim abWriteData() As Byte
ReDim abWriteData(0 To 5)
abWriteData(0) = 0
abWriteData(1) = 1
abWriteData(2) = 2
abWriteData(3) = 3
abWriteData(4) = 4
abWriteData(5) = 5
oFileParameter.AppendChunk abWriteData
```

§ `HRESULT ClearStream ()`

`ClearStream` deletes the data of the stream and of the related file.

§ `HRESULT GetChunk ([in] long Length, [out, retval] VARIANT * pResult)`

`GetChunk` returns the specified number of bytes from the stream.

When accessing the stream functionalities of the interface for the first time, the related file is opened and kept open. If another caller subsequently requires exclusive access to this file, it can be released by invoking the UnloadStream method.

This method starts reading the data in the stream beginning at the current position. If the end of the stream is reached before the required number of characters was read, only those characters read so far will be returned. The number of read characters can be determined with the size of the returned buffer (see example).

Parameter:

[in]: `Length`	Number of maximum returned bytes.

[out]: `pResult` (VB return parameter) contains the bytes read. These will be returned in a variant of type VT_ARRAY|VT_UI1.

Example:

```
' The following program section corresponds to:
' Dim var
' var = oFileParameter.Value
' However, the current position of the stream is also preserved there
' Set stream to read at the starting position
oFileParameter.ResetStream
Dim abReadData() As Byte
...
' Adapt array to the required size
ReDim abReadData(0 To oParameter.ActualSize - 1)
' Read data
abReadData = oFileParameter.GetChunk(oParameter.ActualSize)
' Determine the size of the data read using the data returned
Dim nSize As Long
nSize = UBound(abReadData) - LBound(abReadData)
```

§ `HRESULT ResetStream ()`

`ResetStream` resets the data stream to the beginning.

When accessing the stream functionalities of the interface for the first time, the related file is opened and kept open. If another caller subsequently requires exclusive access to this file, it can be released by invoking the UnloadStream method.

§ `HRESULT Stream ([out, retval] IStream ** ppStream)`

`Stream` provides the stream of binary data

The stream contains the binary data of the file.

When accessing the stream functionalities of the interface for the first time, the related file is opened and kept open. If another caller subsequently requires exclusive access to this file, it can be released by invoking the UnloadStream method.

Parameter:

[out]: `ppStream` (VB return parameter) contains the stream as `IStream` interface.

§ `HRESULT UnloadStream ([in, defaultvalue(-1)] VAIRANT_BOOL AutoReload)`

`UnloadStream` unloads the internal `Stream` object

If external references no longer refer to the `Stream` object, the object is released and the related file is closed. Access to a stream object can be established again using a method of the object.

Note:

If `AutoDelete` is set to true and the `Stream` is unloaded, the related file will be deleted in case no other reference to the stream exists.

Parameter:

[in]: `AutoReload` indicates whether the next time an `IFileParameter` method that requires the `Stream` is accessed it is again reinitialized automatically. Especially when using debug environments that reload the respective properties automatically, problems may occur when the file is to be processed exclusively by another program. In this case, the `VARIANT_FALSE` value should be used for this parameter. For languages that support default parameters, this value is set to `VARIANT_TRUE` if no specification was made.

Documentation of properties:

§ `long ActualSize [get]`

`ActualSize` returns the size of the data stream in bytes.

When accessing the stream functionalities of the interface for the first time, the related file is opened and kept open. If another caller subsequently requires exclusive access to this file, it can be released by invoking the UnloadStream method.

Parameter:

[out]: `pVal`	(VB return parameter) contains the size of the data stream.

§ `VARIANT_BOOL AutoDelete [get, set]`

`AutoDelete` gets and sets whether the file of this object should be deleted automatically after being used.

The automatic file deletion is controlled in the related file stream. Therefore it is possible to continue working with the `Stream` even after the deletion of the `FileParameter` object. If none of the stream functionalities was used, the FileParameter checks at release whether the file should be deleted.

Parameter:

[out]: `pVal`	(VB return parameter) will the file be automatically deleted?

The automatic file deletion is controlled in the related file stream. Therefore it is possible to continue working with the `Stream` even after the deletion of the `FileParameter` object. If none of the stream functionalities was used, the `FileParameter` checks at release whether the file should be deleted.

Parameter:

[in]: `newVal`	will the file be automatically deleted?

§ `BSTR FileName [get, set]`

`FileName` returns the full name of the file and sets a new file name.

This property is a default property of the interface

Parameter:

[out]: `pVal`	(VB return parameter) Name of the file

If another file is already being processed, its resources will be released. If the assignment of a new file name fails (the file cannot be found), an error will be sent and the old object properties remain unchanged.

Parameter:

[in]: `newVal`	New file name

§ `BSTR XML [get, set]`

`XML` gets the file parameter content as an XML string and writes the XML string to be passed into the related file.

When accessing the stream functionalities of the interface for the first time, the related file is opened and kept open. If another caller subsequently requires exclusive access to this file, it can be released by invoking the `UnloadStream` method.

The output XML string is read out with the XML parser.

If no valid XML string can be generated from the data, an error is sent. The error message is related to the XML parser used for the validation (MS-XML4).

The position pointer of the `Stream` remains unchanged when invoking this property.

Parameter:

[out]: `pVal` (VB return parameter) contains the data decoded as an XML string.

Old data of the file will be overwritten when invoking this property. After invoking this property, the position pointer of the `Stream` is at the beginning of it.

Data will be written into the file according to the coding specified in the XML string.

The passed data will be validated during conversion. If data non-compliant with XML are passed, an error will be sent. In this case, the `Stream` has the length 0. The error message is related to the XML parser used for the validation (MS-XML4).

Parameter:

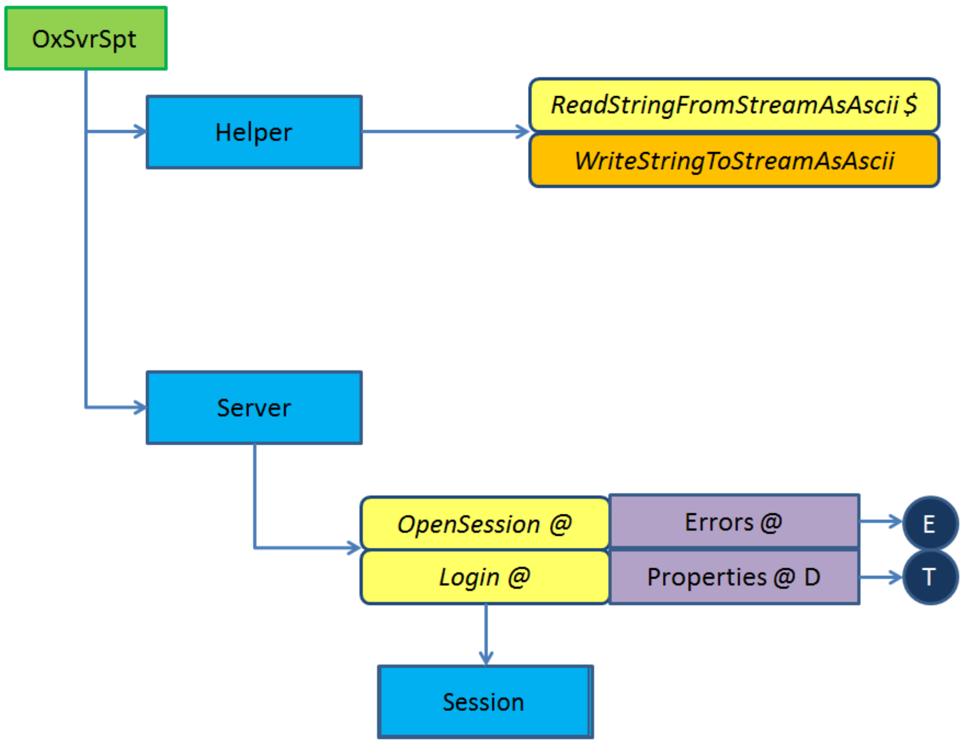
[in]: `newVal` Basic string with XML data.

IHelper

Description:

`IHelper` provides the general help functions for the access from script environments.

`import "OxSvrSpt.idl"`



Public methods:

```
HRESULT WriteStringToStreamAsAscii ([in] IStream * Stream, [in] BSTR
Text)
```

`HRESULT ReadStringFromStreamAsAscii ([in] IStream * Stream, [in] long Length, [out, retval] BSTR * pOut)`

`HRESULT CreateStream ([out, retval] IStream ** ppVal)`

Documentation of the element functions:

§ `HRESULT CreateStream ([out, retval] IStream ** ppVal)`

`CreateStream` generates an new `IStream` object and returns it.

§ `HRESULT ReadStringFromStreamAsAscii ([in] IStream * Stream, [in] long Length, [out, retval] BSTR * pOut)`

`ReadStringFromStreamAsAscii` reads text as ASCII characters from the stream.

A specified number of ASCII characters is read from the `stream` and returned as a basic string. If the `Stream` contains less than the specified number of characters, all existing characters up to the end of the stream will be read and returned. The method starts reading at the current position of the position pointer of the `Stream`. After reading, the position pointer remains at the position where it was located after the operation.

Parameter:

[in]: `Stream`	`IStream` instance which should be read from.
[in]: `Length`	Number of maximum read characters.

[out]: `pOut` (VB return parameter) character that is read from the stream and converted into a BSTR.

§ `HRESULT WriteStringToStreamAsAscii ([in] IStream * Stream, [in] BSTR Text)`

`WriteStringToStreamAsAscii` writes the transferred text to the specified `Stream`.

The text expected as `BSTR` is converted to ASCII and written to the `Stream`. Before writing, the position pointer of the `Stream` is set to its end.

Parameter:

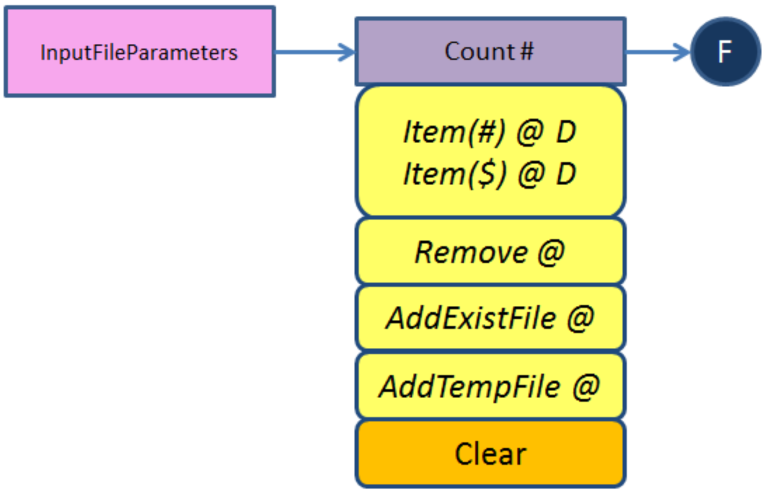
[in]: `Stream`	Stream in which data will be written.
[in]: `Text`	The text as basic string that will be transferred to the `stream`.

IInputFileParameters

Description:

`IInputFileParameters` is a collection for managing the input files of a `Job`.

`import "OxSvrSpt.idl"`



Public methods:

`HRESULT Delete ([in] VARIANT Index)`

`HRESULT Clear ()`

`HRESULT AddExistFile ([in] BSTR FileName, [in, defaultvalue(0)] VARIANT_BOOL AutoDelete, [out, retval] IFileParameter ** ppNewParameter)`

`HRESULT AddTempFile ([in, defaultvalue(0xffff)] VARIANT_BOOL AutoDelete, [in, defaultvalue("tmp")] BSTR FileExtension, [out, retval] IFileParameter ** ppNewParameter)`

Properties:

`long Count [get]`

`IFileParameter Item([in] VARIANT Index) [get]`

Documentation of the element functions:

§ `HRESULT AddExistFile ([in] BSTR FileName, [in, defaultvalue(0)] VARIANT_BOOL AutoDelete, [out, retval] IFileParameter ** ppNewParameter)`

`AddExistFile` adds an existing file as a new `FileParameter` and returns it.

The transferred file is not deleted by default.

It is checked whether the file exists. But it will be opened only when its data is accessed using the respective methods of the IFileParameter interface.

Parameter:

[in]: `FileName`name of the file.

[in]: `AutoDelete` (default value: VARIANT_FALSE) indicates whether the file will be automatically deleted by this class after use.

[out]: `ppNewParameter` (VB return value) associated `FileParameter` object.

§ `HRESULT AddTempFile ([in, defaultvalue(0xffff)] VARIANT_BOOL AutoDelete, [in, defaultvalue("tmp")] BSTR FileExtension, [out, retval] IFileParameter ** ppNewParameter)`

`AddTempFile` generates a `File` parameter and automatically creates a temporary file for the data transfer to the server

This file is deleted by default after being used.

If required, the name of the file can be determined using the FileName property of the returned IFileParameter object.

Parameter:

[in]: `AutoDelete` (default value: VARIANT_FALSE) indicates whether the file will be automatically deleted by this class after use.

[in]: `FileExtension` (default value: tmp) indicates a file extension for the temporary file.

[out]: `ppNewParameter` (VB return value) associated `FileParameter` object.

§ `HRESULT Clear ()`

`Clear` removes all items of the collection.

§ `HRESULT Delete ([in] VARIANT Index)`

`Delete` deletes the parameter based on the position in the list or on the name.

Parameter:

[in]: `Index` Name or position of the license entry to be deleted.

Documentation of properties:

§ `long Count [get]`

`Count` returns the number of items of the collection.

Parameter:

[out]: `plNumber` (VB return value) Number of elements in the collection.

§ `IFileParameter Item([in] VARIANT Index) [get]`

`Item` returns the specified item of the collection using the key or its position.

If a position is specified outside of the valid index, an error with the error value `errCollectionIndexOutOfRange` is returned. If the item cannot be found, an error with the error value `errCollectionItemNotFound` is returned.

Parameter:

[in]: `Index` Position and name of the requested element.

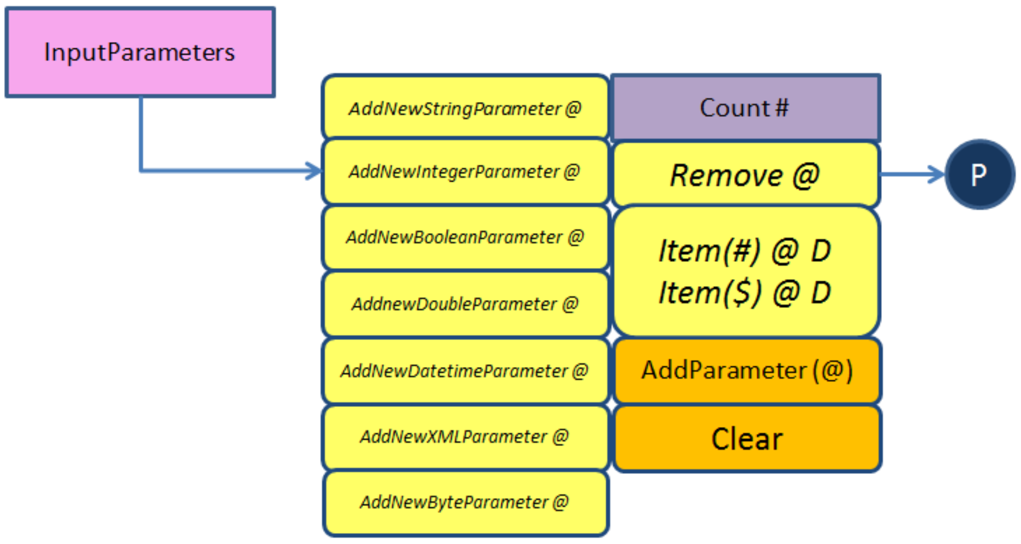
[out]: `ppItem` (VB return value) associated `FileParameter` object.

IInputParameters

Description:

`IInputParameters` is a collection for creating and managing the `InputParameters` of a `Job`.

`import "OxSvrSpt.idl"`



Public methods:

```
HRESULT AddNewStringParameter ([in] BSTR Name, [in] BSTR Value, [out,
retval] IParameter ** ppVal)
```

```
HRESULT AddNewIntegerParameter ([in] BSTR Name, [in] long Value, [out,
retval] IParameter ** ppVal)
```

```
HRESULT AddNewBooleanParameter ([in] BSTR Name, [in] VARIANT_BOOL Value,
[out, retval] IParameter ** ppVal)
```

```
HRESULT AddNewDoubleParameter ([in] BSTR Name, [in] double Value, [out,
retval] IParameter ** ppVal)
```

```
HRESULT AddNewDatetimeParameter ([in] BSTR Name, [in] DATE Value, [out,
retval] IParameter ** ppVal)
```

```
HRESULT AddNewXMLParameter ([in] BSTR Name, [in, defaultvalue("")] BSTR
XML, [out, retval] IParameter ** ppVal)
```

```
HRESULT AddNewByteParameter ([in] BSTR Name, [in, optional] VARIANT
Value, [out, retval] IParameter ** ppVal)
```

```
HRESULT AddParameter ([in] IParameter * Parameters)
```

```
HRESULT Remove ([in] VARIANT Index, [out, retval] IParameter ** ppVal)
```

```
HRESULT Clear ()
```

Properties:

```
long Count [get]
```

```
IParameter Item([in] VARIANT Index) [get]
```

Documentation of the element functions:

§ `HRESULT AddNewBooleanParameter ([in] BSTR Name, [in] VARIANT_BOOL Value, [out, retval] IParameter ** ppVal)`

`AddNewBooleanParameter` generates a new Boolean parameter, adds it to the collection, and returns it. The parameter name must not be an empty string, otherwise an error will be returned. If any attempts are made to add a parameter with a name that already exists, an error will be caused.

Parameter:

[in]: `Name` Parameter name.

[in]: `Value` Initialization value of the parameter.

[out]: `ppVal` (VB return parameter) generated and initialized parameter object.

Exception handling:

`errInputParametersCantCreate` (1101) Unable to create the parameter. In this case, no further information is available.

`errParameterNameEmpty` (1201) No name was specified for the parameter.

`errParameterDoubleName` (1202) A parameter with this name already exists.

§ `HRESULT AddNewByteParameter ([in] BSTR Name, [in, optional] VARIANT Value, [out, retval] IParameter ** ppVal)`

`AddNewByteParameter` generates a new Byte parameter, adds it to the collection, and returns it. The parameter name must not be an empty string, otherwise an error will be returned. If any attempts are made to add a parameter with a name that already exists, an error will be caused.

Upon initialization, the value can be passed as a string or added subsequently using the `AppendChunk` or `Stream` functions. If `Value` is to be initialized later, a variant of type `VT_NULL` or `VT_ERROR` must be passed.

In VB or VBScript, the `Value` parameter of the method must not be specified, if binary data is to be added to the parameter object at a later point.

Parameter:

[in]: `Name` Parameter name.

[in]: `Value` Initialization value of the parameter.

[out]: `ppVal` (VB return parameter) generated and initialized parameter object.

Exception handling:

`errInputParametersCantCreate` (1101) Unable to create the parameter. In this case, no further information is available.

`errParameterNameEmpty` (1201) No name was specified for the parameter.

`errParameterDoubleName` (1202) A parameter with this name already exists.

Example:

```
Dim oParameter As OxSvrSpt.Parameter
Set oParameter = m_oInputParameters.AddNewByteParameter(strName)
```

§ `HRESULT AddNewDatetimeParameter ([in] BSTR Name, [in] DATE Value, [out, retval] IParameter ** ppVal)`

`AddNewDatetimeParameter` generates a new Double parameter, adds it to the collection, and returns it. The parameter name must not be an empty string, otherwise an error will be returned. If any attempts are made to add a parameter with a name that already exists, an error will be caused. VT_8

Parameter:

[in]: `Name` Parameter name.

[in]: `Value` Initialization value of the parameter.

[out]: `ppVal` (VB return parameter) generated and initialized parameter object.

Exception handling:

`errInputParametersCantCreate` (1101) Unable to create the parameter. In this case, no further information is available.

`errParameterNameEmpty` (1201) No name was specified for the parameter.

`errParameterDoubleName` (1202) A parameter with this name already exists.

§ `HRESULT AddNewDoubleParameter ([in] BSTR Name, [in] double Value, [out, retval] IParameter ** ppVal)`

`AddNewDoubleParameter` generates a new Double parameter, adds it to the collection, and returns it. The parameter name must not be an empty string, otherwise an error will be returned. If any attempts are made to add a parameter with a name that already exists, an error will be caused. VT_8

Parameter:

[in]: `Name` Parameter name.

[in]: `Value` Initialization value of the parameter.

[out]: `ppVal` (VB return parameter) generated and initialized parameter object.

Exception handling:

`errInputParametersCantCreate` (1101) Unable to create the parameter. In this case, no further information is available.

`errParameterNameEmpty` (1201) No name was specified for the parameter.

`errParameterDoubleName` (1202) A parameter with this name already exists.

§ `HRESULT AddNewIntegerParameter ([in] BSTR Name, [in] long Value, [out, retval] IParameter ** ppVal)`

`AddNewIntegerParameter` generates a new Integer parameter, adds it to the collection, and returns it.

The parameter name must not be an empty string, otherwise an error will be returned. If any attempts are made to add a parameter with a name that already exists, an error will be caused. VT_I4

Parameter:

[in]: `Name` Parameter name.

[in]: `Value` Initialization value of the parameter.

[out]: `ppVal` (VB return parameter) generated and initialized parameter object.

Exception handling:

`errInputParametersCantCreate` (1101) Unable to create the parameter. In this case, no further information is available.

`errParameterNameEmpty` (1201) No name was specified for the parameter.

`errParameterDoubleName` (1202) A parameter with this name already exists.

§ `HRESULT AddNewStringParameter ([in] BSTR Name, [in] BSTR Value, [out, retval] IParameter ** ppVal)`

`AddNewStringParameter` generates a new String parameter, adds it to the collection, and returns it.

The parameter name must not be an empty string, otherwise an error will be returned. If any attempts are made to add a parameter with a name that already exists, an error will be caused.

Parameter:

[in]: `Name`	Parameter name.
[in]: `Value`	Initialization value of the parameter.
[out]: `ppVal`	(VB return parameter) generated and initialized parameter object.

Exception handling:

`errInputParametersCantCreate` (1101) Unable to create the parameter. In this case, no further information is available.

`errParameterNameEmpty` (1201) No name was specified for the parameter.

`errParameterDoubleName` (1202) A parameter with this name already exists.

Example:

C++

```
try
{
IInputParametersPtr spParameters( spJob->InputParameters );
_bstr_t bstrParameterName( L"Name" );
_bstr_t bstrParameterValue( L"Value" );
spParameters->AddNewStringParameter( bstrParameterName, bstrParameterValue );
}
catch( _com_error& e )
{
// Determine error description
_bstr_t bstrError = e.Description( );
// if no error description was delivered by the COM error object...
if( bstrError.length() == 0 )
{
// ... determine the system error message
bstrError = e.ErrorMessage();
}
// TODO bstrError contains the error description for
//       further processing.
}
```

§ `HRESULT AddNewXMLParameter ([in] BSTR Name, [in, defaultvalue("")] BSTR XML, [out, retval] IParameter ** ppVal)`

`AddNewXMLParameter` generates a new XML parameter, adds it to the collection, and returns it. The parameter name must not be an empty string, otherwise an error will be returned. If any attempts are made to add a parameter with a name that already exists, an error will be caused.

Upon initialization, the value can be passed as a string or added subsequently using the `AppendChunk` or `Stream` functions.

In VB or VBScript, the XML parameter of the method must not be specified, if XML data is to be added to the parameter object at a later point.

Parameter:

[in]: `Name` Parameter name.

[in]: `XML` Initialization value of the parameter.

[out]: `ppVal` (VB return parameter) generated and initialized parameter object.

Exception handling:

`errInputParametersCantCreate` (1101) Unable to create the parameter. In this case, no further information is available.

`errParameterNameEmpty` (1201) No name was specified for the parameter.

`errParameterDoubleName` (1202) A parameter with this name already exists.

Example:

```
Dim oParameter As OxSvrSpt.Parameter
Set oParameter = m_oInputParameters.AddNewXMLParameter(strName)
```

§ `HRESULT AddParameter ([in] IParameter * Parameters)`

`AddParameter` adds the passed parameter to the collection.

With this method, it is possible to pass parameters of other calls without a copy.

Parameter:

[in]: `Parameter` Parameter to be added.

Exception handling:

`errInputParametersCantCreate` (1101) Unable to create the parameter. In this case, no further information is available.

`errParameterNameEmpty` (1201) No name was specified for the parameter.

`errParameterDoubleName` (1202) A parameter with this name already exists.

Example:

```
Dim oParameter As OxSvrSpt.Parameter
...
' Fill parameter in another job (call)
...
m_oInputParameters.AddParameter(oParameter)
```

§ `HRESULT Clear ()`

`Clear` removes all items of the collection.

§ `HRESULT Remove ([in] VARIANT Index, [out, retval] IParameter ** ppVal)`

`Remove` removes the entry with the passed identifier from the parameter list and returns it.

If the specified entry cannot be found, an error is returned.

Parameter:

[in]: `Index` Name or position of the entry that will be removed from the parameter collection. If the `Index` parameter is passed as `Integer` or `Long`, the item at the respective position is searched for. In C++, a variant of type VT_12 or TV_14 must be passed in this case. If the `Index` parameter is passed as a string (VT_BSTR), the name of the item is searched for.

[out]: `ppVal` (VB return parameter) returns the removed parameter object.

Exception handling:

`errCollectionItemNotFound` (1304) Unable to find the requested entry.

`errCollectionIndexOutOfRange` (1303) The requested index is out of range.

Documentation of properties:

§ `long Count [get]`

`Count` returns the number of items of the collection.

Parameter:

[out]: `plNumber`	(VB return value) Number of elements in the collection

§ `IParameter Item([in] VARIANT Index) [get]`

`Item` returns the specified item of the collection using the key or its position.

If a position is specified outside of the valid index, an error with the error value `errCollectionIndexOutOfRange` is returned. If the item cannot be found, an error with the error value `errCollectionItemNotFound` is returned.

Parameter:

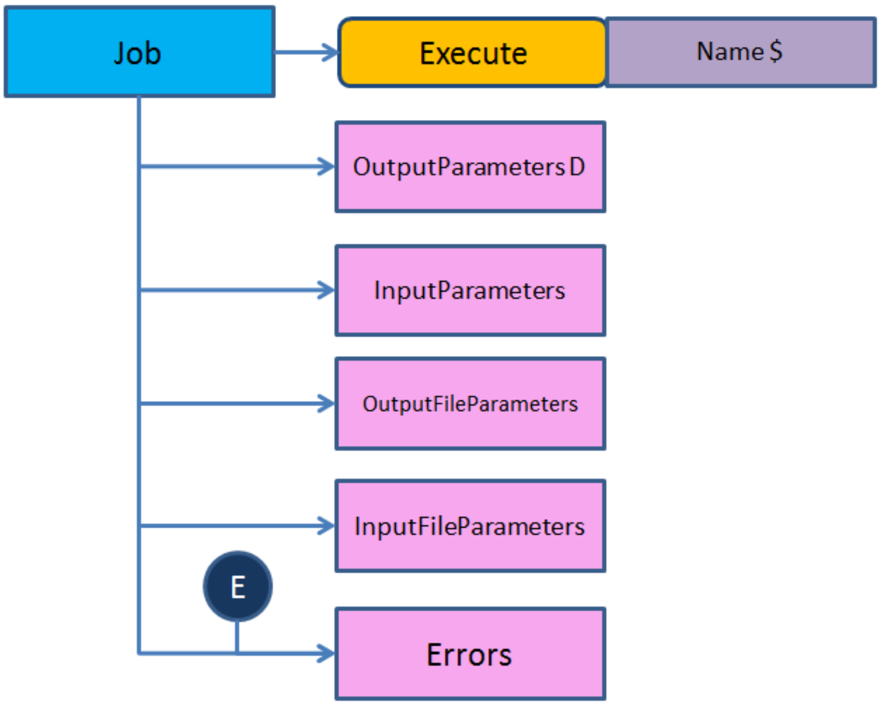
[in]: `Index`	Position and name of the requested element
[out]: `ppItem`	Associated parameter object

IJob

Description:

`IJob` encapsulates access to the parameters and a `Job` call.

`import "OxSvrSpt.idl"`



Public methods:

`HRESULT Execute ()`

Properties:

`IOutputParameters OutputParameters [get]`

`BSTR Name [get]`

`IInputParameters InputParameters [get]`

`IErrors Errors [get]`

`IInputFileParameters InputFileParameters [get]`

`IOutputFileParameters OutputFileParameters [get]`

Documentation of the element functions:

§ `HRESULT Execute ()`

After calling this method, the output parameters are available in the collections `OutputParameters` and `OutputFileParameters`. If an error occurs when executing the `Job`, a COM error is thrown. This is the case both for logical and system errors. The COM error contains the first message of the `Error` collection. All other messages can be found in the `Error` collection with the `Errors` property of this object.

Examples:

VB

```
' Log in
Dim oServer As New OxSvrSpt.Server
Dim oSession As OxSvrSpt.Session
Set oSession = oServer.Login("root", "optimal", "localhost", "4000",
pwNotEncrypted)

' Create job
Dim oJob As OxSvrSpt.Job
Set oJob = oSession.NewJob("dms.GetResultList")

' Add parameter
Dim oParameter As OxSvrSpt.Parameter
oJob.InputParameters.AddNewIntegerParameter "Flags", 16
Set oParameter = oJob.InputParameters.AddNewXMLParameter("XML")
Dim oDomDocument As New MSXML2.DOMDocument40
Dim bSuccess As Boolean
bSuccess = oDomDocument.Load("c:\dmstest.xml")
oDomDocument.save oParameter.Stream

' Execute job
oJob.Execute

' Read XML from the XML file parameter
Dim strXML As String
strXML = oJob.OutputFileParameters(1).XML
```

C++ using the import mechanism

```
#import "../Debug/OxSvrSpt.dll" raw_method_prefix("raw_")
using namespace OxSvrSpt;
.
.
.
try
{
_bstr_t bstrUser                = L"root";
_bstr_t bstrPassword            = L"optimal";
_bstr_t bstrServer              = L"adunkel";
_bstr_t bstrPort                = L"4000";
_bstr_t bstrJobName             = L"krn.GetServerInfo";
OxSvrSpt::IServerPtr spServer( __uuidof( OxSvrSpt::Server ));
OxSvrSpt::ISessionPtr spSession = spServer->Login( bstrUser, bstrPassword,
bstrServer, bstrPort, pwNotEncrypted );
OxSvrSpt::IJobPtr spJob = spSession->NewJob( bstrJobName );
spJob->InputParameters->AddNewIntegerParameter( L"Flags", 0 );
spJob->InputParameters->AddNewIntegerParameter( L"Info", 1 );
spJob->Execute();
_variant_t varName = spJob->OutputParameters->GetItem( L"Name" )->Value;
}
catch( _com_error& e )
{
_bstr_t bstrError = e.Description( );
if( bstrError.length() == 0 )
{
bstrError = e.ErrorMessage();
}
AfxMessageBox( bstrError );
}
```

Properties documentation:

§ `IErrors Errors [get]`

`Errors` returns the collection with the errors that occurred at the server.

This collection contains objects of the `IError` type.

Parameter:

[out]: `pVal` (VB return value) Collection with the errors that occurred

§ `IInputFileParameters InputFileParameters [get]`

`InputFileParameters` returns the collection with the `FileParameters` for the transfer to the server

This collection contains objects of the `IFileParameters` type

Parameter:

[out]: `pVal` (VB return value) Collection with the files for the server call

§ `IInputParameters InputParameters [get]`

`InputFileParameters` returns the collection of `InputParameters`.

Parameter:

[out]: `pVal` (VB return value) Collection of the input parameters

§ `BSTR Name [get]`

`Name` returns the name of the `Job`.

The name of the `Job` is entered when the `Job` object is created and cannot be modified afterwards.

Parameter:

[out]: `pVal` (VB return value) `Job` name

§ `IOutputFileParameters OutputFileParameters [get]`

`OutputFileParameters` returns the collection with the `FileParameters` that are provided by the server after a `Job` call.

This collection contains objects of the `IFileParameters` type

Parameter:

[out]: `pVal` (VB return value) Collection with the result files after a server call.

§ `IOutputParameters OutputParameters [get]`

`OutputParameters` returns the collection of `OutputParameters`.

This property is a default property of the `IJob` interface.

Parameter:

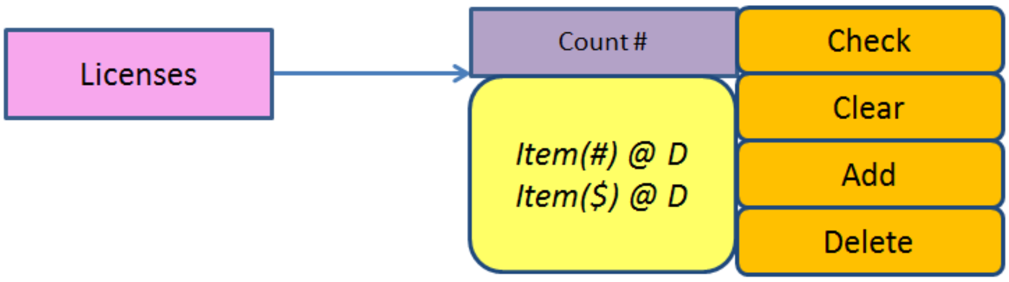
[out]: `pVal` (VB return value) Collection of the output parameters for the `Job`

ILicenses

Description:

`ILicenses` is a collection for managing licenses.

`import "OxSvrSpt.idl"`



Public methods:

`HRESULT Add ([in] BSTR Name)`

`HRESULT Delete ([in] VARIANT Index)`

`HRESULT Clear ()`

`HRESULT Check ([in] BSTR Name)`

Properties:

`long Count [get]`

`BSTR Item([in] VARIANT Index) [get]`

Documentation of the element functions:

§ `HRESULT Add ([in] BSTR Name)`

`Add` adds a new license string.

If any attempts are made to add a license string twice, the second string will be ignored. No error will be thrown. The transferred license string is converted into capitals and managed internally.

Parameter:

[in]: `Name`	License string to be added

§ `HRESULT Check ([in] BSTR Name)`

`Check` validates the entered license at the server without checking it in using `LicLogin`.

This method operates through the server job "`lic.CheckLicense`"

If the validation fails, the first error in the error collection is returned as a COM error.

Parameter:

[in]: `Name`	the license to be checked

§ `HRESULT Clear ()`

`Clear` removes all items of the collection.

§ `HRESULT Delete ([in] VARIANT Index)`

`Delete` deletes the license entry based on the position in the list or on the name.

Parameter:

[in]: `Index`	Name or position of the license entry to be deleted.

Documentation of properties:

§ `long Count [get]`

`Count` returns the number of items of the collection.

Parameter:

[out]: `plNumber`	(VB return value) Number of elements in the collection

§ `BSTR Item([in] VARIANT Index) [get]`

`Item` returns the specified item of the collection using the key or its position.

If a position is specified outside of the valid index, an error with the error value `errCollectionIndexOutOfRange` is returned. If the item cannot be found, an error with the error value `errCollectionItemNotFound` is returned.

Parameter:

[in]: `Index`	Position and name of the requested element
[out]: `pItem`	(VB return value) associated license that corresponds to the key

ILogger

Description:

`ILogger` is an interface for accessing OS-Logger.

`import "OxSvrSpt.idl"`

Public methods:

```
HRESULT Log ([in] BSTR Message, [in] LogLevelEnum Level, [in,
defaultvalue("")] BSTR FileName, [in, defaultvalue("")] BSTR Function,
[in, defaultvalue(0)] long Line)
```

```
HRESULT Error ([in] BSTR Message, [in, defaultvalue("")] BSTR Function,
[in, defaultvalue(0)] long Line)
```

```
HRESULT Info ([in] BSTR Message, [in, defaultvalue("")] BSTR Function,
[in, defaultvalue(0)] long Line)
```

```
HRESULT MethodEntry ([in] BSTR Message, [in, defaultvalue("")] BSTR
Function, [in, defaultvalue(0)] long Line)
```

```
HRESULT Debug ([in] BSTR Message, [in, defaultvalue("")] BSTR Function,
[in, defaultvalue(0)] long Line)
```

```
HRESULT Trace ([in] BSTR Message, [in, defaultvalue("")] BSTR Function,
[in, defaultvalue(0)] long Line)
```

```
HRESULT Init ([in] BSTR FileName, [in, defaultvalue("")] BSTR Alias)
```

Documentation of the element functions:

§ `HRESULT Debug ([in] BSTR Message, [in, defaultvalue("")] BSTR Function, [in, defaultvalue(0)] long Line)`

`Debug` logs the transferred message in the debug level.

Parameter:

[in]: `Message` Message to be transferred

[in]: `Function` Name of the function in which the log entry is made. If it is not specified, an empty string is displayed in the log.

[in]: `Line` Source text lines to be logged. If it is not specified, the value 0 will be displayed in the log.

§ `HRESULT Error ([in] BSTR Message, [in, defaultvalue("")] BSTR Function, [in, defaultvalue(0)] long Line)`

`Error` logs the transferred message in the error level.

Parameter:

[in]: `Message` Message to be transferred

[in]: `Function` Name of the function in which the log entry is made. If it is not specified, an empty string is displayed in the log.

[in]: `Line` Source text lines to be logged. If it is not specified, the value 0 will be displayed in the log.

§ `HRESULT Info ([in] BSTR Message, [in, defaultvalue("")] BSTR Function, [in, defaultvalue(0)] long Line)`

`Info` logs the transferred message in the info level.

Parameter:

[in]: `Message` Message to be transferred

[in]: `Function` Name of the function in which the log entry is made. If it is not specified, an empty string is displayed in the log.

[in]: `Line` Source text lines to be logged. If it is not specified, the value 0 will be displayed in the log.

§ `HRESULT Init ([in] BSTR FileName, [in, defaultvalue("")] BSTR Alias)`

`Init` initializes the logger instance.

Parameter:

[in]: `FileName` File name for which all log entries will be made

[in]: `Alias` Alias of the current library for which logs will be made

§ `HRESULT Log ([in] BSTR Message, [in] LogLevelEnum Level, [in, defaultvalue("")] BSTR FileName, [in, defaultvalue("")] BSTR Function, [in, defaultvalue(0)] long Line)`

`Log` logs the transferred message in the specified level.

Parameter:

[in]: `Message` Message to be transferred

[in]: `Level` Log level to be applied

[in]: `FileName` Name of the source file from which the log entry came. If it is not specified, an empty string is displayed in the log.

[in]: `Function` Name of the function in which the log entry is made. If it is not specified, an empty string is displayed in the log.

[in]: `Line` Source text lines to be logged. If it is not specified, the value 0 will be displayed in the log.

§ `HRESULT MethodEntry ([in] BSTR Message, [in, defaultvalue("")] BSTR Function, [in, defaultvalue(0)] long Line)`

`MethodEntry` logs the method entry.

Parameter:

[in]: `Message` Message to be transferred

[in]: `Function` Name of the function in which the log entry is made. If it is not specified, an empty string is displayed in the log.

[in]: `Line` Source text lines to be logged. If it is not specified, the value 0 will be displayed in the log.

§ `HRESULT Trace ([in] BSTR Message, [in, defaultvalue("")] BSTR Function, [in, defaultvalue(0)] long Line)`

`Trace` logs the transferred message in the trace level.

Parameter:

[in]: `Message` Message to be transferred

[in]: `Function` Name of the function in which the log entry is made. If it is not specified, an empty string is displayed in the log.

[in]: `Line` Source text lines to be logged. If it is not specified, the value 0 will be displayed in the log.

INotifyErrors

Description:

`INotifyErrors` is an interface for managing errors of a notification. This interface extends the IErrors interface by adding the possibility to add error messages to the collection or remove them.

```
import "OxSvrSpt.idl"
```

Public methods:

```
HRESULT Add ([in] BSTR SourceName, [in] long ISourceCode, [in] BSTR
FaultString, [in] long FaultCode, [in] BSTR InfoList)
```

```
HRESULT Clear ()
```

```
HRESULT NewResultCode ([in] long newValue)
```

```
HRESULT NewResponseResult ([in] long newValue)
```

Documentation of the element functions:

§ `HRESULT Add ([in] BSTR SourceName, [in] long ISourceCode, [in] BSTR FaultString, [in] long FaultCode, [in] BSTR InfoList)`

Add adds a new error message to the collection.

Parameter:

[in]: `SourceName`	Name of the source in which the error occurred
[in]: `ISourceCode`	Source line in which the error occurred.
[in]: `FaultString`	Error description
[in]: `FaultCode`	Error code
[in]: `InfoList`	Info list

§ `HRESULT Clear ()`

`Clear` removes all error objects from the collection.

§ `HRESULT NewResponseResult ([in] long newValue)`

`NewResponseResult` sets the return value of the notification.

Parameter:

[out]: `newVal` Return value for the notification

§ `HRESULT NewResultCode ([in] long newValue)`

`NewResultCode` sets the error value of the notification.

Parameter:

[out]: `newVal` Error value of the server

INotifyInputFileParameters

Description:

`INotifyInputFileParameters` contains the input file parameters of a notification.

`import "OxSvrSpt.idl"`

INotifyInputParameters

Description:

`INotifyInputParameters` is a collection for managing the input parameters of a notification. The methods correspond to the `IOutputParameters` interface.

`import "OxSvrSpt.idl"`

INotifyJob

Description:

`INotifyJob` contains the data when invoking a notification.

`import "OxSvrSpt.idl"`

Properties:

`INotifyOutputParameters OutputParameters [get]`

`INotifyErrors Errors [get]`

`INotifyInputParameters InputParameters [get]`

`BSTR Name [get]`

`INotifyOutputFileParameters OutputFileParameters [get]`

`INotifyInputFileParameters InputFileParameters [get]`

`long UserData [get, set]`

Documentation of properties:

§ `INotifyErrors Errors [get]`

`Errors` returns the collection in which errors are recorded.

Parameter:

[out]: `pVal` (VB return parameter) `INotifyErrors` collection

§ `INotifyInputFileParameters InputFileParameters [get]`

`InputFileParameters` returns the collection of file parameters transferred to the notification.

Parameter:

[out]: `pVal` (VB return parameter) `INotifyInputFileParameters` collection

§ `INotifyInputParameters InputParameters [get]`

`InputParameters` returns the collection of parameters transferred to the notification.

Parameter:

[out]: `pVal` (VB return parameter) `INotifyInputParameters` collection

§ `BSTR Name [get]`

`Name` returns the name of the notification.

Parameter:

[out]: `pVal`	(VB return parameter) Name of the notification

§ `INotifyOutputFileParameters OutputFileParameters [get]`

`OutputFileParameters` returns the collection of output file parameters.

Parameter:

[out]: `pVal`	(VB return parameter) `INotifyOutputFileParameters` collection

§ `INotifyOutputParameters OutputParameters [get]`

`OutputParameters` returns the collection of output parameters.

Parameter:

[out]: `pVal`	(VB return parameter) `INotifyOutputParameters` collection

§ `long UserData [get, set]`

`UserData` returns and sets the user data of the notification.

Parameter:

[out]: `pVal`	(VB return parameter) User data of the notification
[in]: `newVal`	User data of the notification

INotifyOutputFileParameters

Description:

`INotifyOutputFileParameters` is a collection for managing the output file parameters of a notification.

```
import "OxSvrSpt.idl"
```

INotifyOutputParameters

Description:

`INotifyOutputParameters` is a collection for creating and managing the `OutputParameters` of a notification.

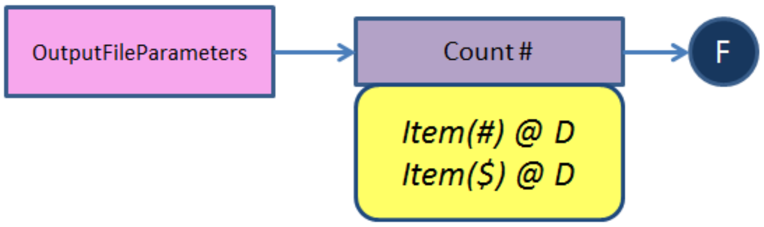
```
import "OxSvrSpt.idl"
```

IOutputFileParameters

Description:

`IOutputFileParameters` contains the file parameters after a server job call.

```
import "OxSvrSpt.idl"
```



Properties:

`long Count [get]`

`IFileParameter Item([in] VARIANT Index) [get]`

Documentation of properties:

§ `long Count [get]`

`Count` returns the number of items of the collection.

Parameter:

[out]: `plNumber` (VB return value) Number of elements in the collection.

§ `IFileParameter Item([in] VARIANT Index) [get]`

`Item` returns the specified item of the collection using the key or its position.

If a position is specified outside of the valid index, an error with the error value `errCollectionIndexOutOfRange` is returned. If the item cannot be found, an error with the error value `errCollectionItemNotFound` is returned.

Parameter:

[in]: `Index` Position and name of the requested element.

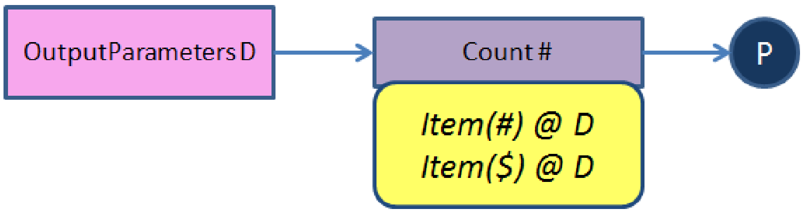
[out]: `ppItem` (VB return value) associated `FileParameter` object.

IOutputParameters

Description:

`IOutputParameters` is a collection for managing the `OutputParameters` of a job.

`import "OxSvrSpt.idl"`



Properties:

`long Count [get]`

`IFileParameter Item([in] VARIANT Index) [get]`

Documentation of properties:

§ `long Count [get]`

`Count` returns the number of items of the collection.

Parameter:

[out]: `plNumber`	(VB return value) Number of elements in the collection.

§ `IFileParameter Item([in] VARIANT Index) [get]`

`Item` returns the specified item of the collection using the key or its position.

If a position is specified outside of the valid index, an error with the error value `errCollectionIndexOutOfRange` is returned. If the item cannot be found, an error with the error value `errCollectionItemNotFound` is returned.

Parameter:

[in]: `Index`	Position and name of the requested element.
[out]: `ppItem`	(VB return value) associated `FileParameter` object.

IParameter

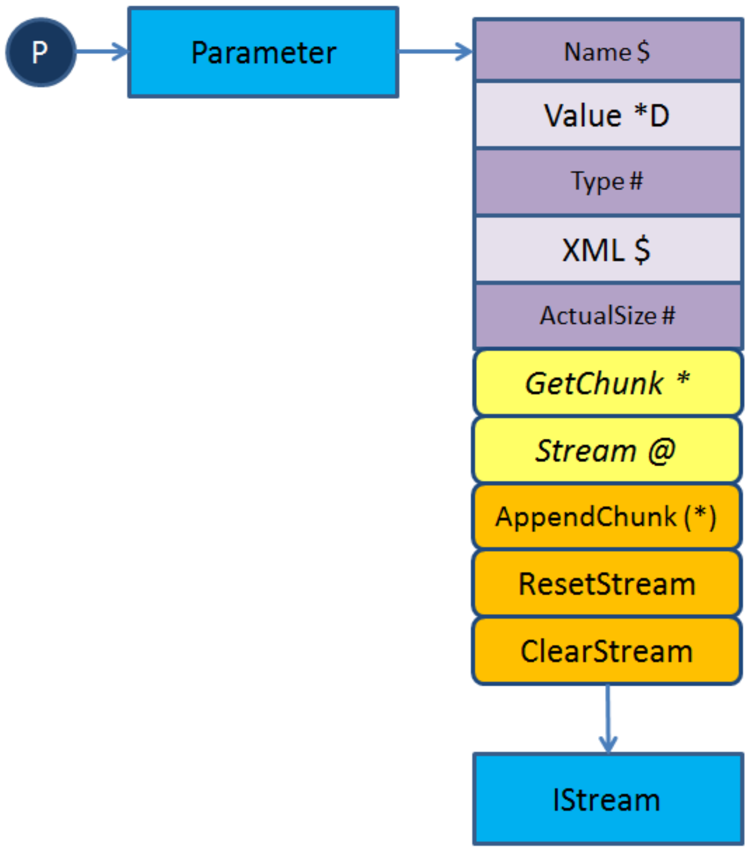
Description:

`IParameter` represents both an input parameter and an output parameter of a job.

The `IParameter` interface provides methods for accessing the properties of a parameter. The methods are divided into two groups. The first group is available for all parameter types. The second group is only available for parameters of the binary type.

Name, type and value belong to the first group. They represent the easiest access to the parameter object. Name and type are only available as read-only properties. The value property can also be modified subsequently. Depending on the parameter type, values passed to this property are treated differently. In the process, no validation takes place whether the passed value corresponds to the requested variant type. Instead, it is attempted to convert the passed value into this type. This is done using the COM conversion functions of the variant. If the conversion fails, the respective COM error is thrown. The used target types are listed in the value property. When setting the value property

All other methods belong to the second group. They are used for processing the binary data of the parameter. If one of these methods is called for a non-binary parameter, the `errParameterMethodUnsupported` error is thrown.



Public methods:

```
HRESULT Stream ([out, retval] IStream ** ppStream)
HRESULT AppendChunk ([in] VARIANT Data)
HRESULT GetChunk ([in] long Length, [out, retval] VARIANT * pResult)
HRESULT ResetStream ()
HRESULT ClearStream ()
```

Properties:

```
VARIANT Value [get, set]
BSTR Name [get]
long ActualSize [get]
BSTR XML [get, set]
ParameterTypeEnum Type [get]
```

Documentation of the element functions:

§ `HRESULT AppendChunk ([in] VARIANT Data)`

`AppendChunk` appends additional bytes to the `stream` of the parameter.

This method is only available for parameters of the types `ptBinary` and `ptXML`.

After invoking this method, the position pointer of the internal stream is at the end of the stream.

Parameter:

[in]: `Data` contains the data that will be appended to the stream

The data will be converted to `VT_ARRA|VT_UI1` and further processed using the `OLE32 ChangeType` method. If, for example, a BSTR is passed, the data will be processed as `WIDECHAR`. Writing 8-bit characters into the stream can be done with the methods of the `Helper-COM` object.

Examples:

VB

```
Dim abWriteData() As Byte
ReDim abWriteData(0 To 5)
abWriteData(0) = 0
abWriteData(1) = 1
abWriteData(2) = 2
abWriteData(3) = 3
abWriteData(4) = 4
abWriteData(5) = 5
oParameter.AppendChunk abWriteData
```

§ `HRESULT ClearStream ()`

`ClearStream` deletes the data of the stream.

§ `HRESULT GetChunk ([in] long Length, [out, retval] VARIANT * pResult)`

`GetChunk` returns the specified number of bytes from the stream.

This method is only available for parameters of the types `ptBinary` and `ptXML`.

This method starts reading the data in the stream beginning at the current position. If the end of the stream is reached before the required number of characters was read, only those characters read so far will be returned. The number of read characters can be determined with the size of the returned buffer (see example).

Parameter:

[in]: `Length` Number of maximum returned bytes.

[out]: `pResult` (VB return parameter) contains the bytes read. These will be returned in a variant of type `VT_ARRAY|VT_UI1`.

Example:

VB

```
' The following program section corresponds to:
' Dim var
' var = oParameter.Value
' However, the current position of the stream is also preserved there
' Set stream to read at the starting position
oParameter.ResetStream
Dim abReadData() As Byte
...
' Adapt array to the required size
ReDim abReadData(0 To oParameter.ActualSize - 1)
' Read data
abReadData = oParameter.GetChunk(oParameter.ActualSize)
' Determine the size of the data read using the data returned
Dim nSize As Long
nSize = UBound(abReadData) - LBound(abReadData)
```

§ `HRESULT ResetStream ()`

`ResetStream` resets the data stream to the beginning.

This method is only available for parameters of the types `ptBinary` and `ptXML`.

§ `HRESULT Stream ([out, retval] IStream ** ppStream)`

`Stream` provides the stream of binary data

This property is only available for parameters of the types `ptBinary` and `ptXML`.

The stream contains the binary data of the respective parameter. The data must not be encoded or decoded in MIME-BASE64 format by the caller. This is done automatically by the `OxSvrSpt` library.

Parameter:

[out]: `ppStream` (VB return parameter) contains the stream as `IStream` interface.

Documentation of properties:

§ `long ActualSize [get]`

`ActualSize` returns the size of the data stream in bytes.

This method is only available for parameters of the types `ptBinary` and `ptXML`.

Parameter:

[out]: `pVal` (VB return parameter) contains the size of the data stream.

§ `BSTR Name [get]`

`Name` returns the name of the parameter.

Parameter:

[out]: `pVal` (VB return parameter) Name of the file

§ `ParameterTypeEnum Type [get]`

`Type` returns the type of the parameter.

The following types are available:

`ptString` = 1

`ptInteger` = 2

`ptBoolean` = 3

`ptDouble` = 4

`ptDateTime` = 5

`ptBinary` =6

`ptXML` = 6

The `ptBinary` and `ptXML` parameters relate to the Base64 parameter of the server. They only differ in how they are created.

Parameter:

[out]: `pVal` (VB return parameter) Type of parameter

§ `VARIANT Value [get, set]`

`Value` returns the parameter value for non-Base64 parameters and sets the value of this parameter.

This property is not available for parameters of the Base64 type. Instead, decoded binary data can be accessed directly with the `Stream` and `Chunk` functions. Furthermore, the XML property is an option.

For parameters of the Base64 type, it is possible to access data with the `Stream` and `Chunk` functionalities.

When setting the property, an attempt is being made to convert the transferred value of the variant into the required type. The following types can be used as target types:

`ptString - VT_BSTR ptInteger - VT_I4 ptBoolean - VT_BOOL ptDouble - VT_R8 ptDateTime - VT_DATE ptBase64 - VT_ARRAY | VT_UI1`

Parameter:

[in]: `newVal` New value of the parameter

For parameters of the Base64 type, binary data is returned in the return variant as an `Array`. The position pointer of the `Stream` remains unchanged when invoking this property.

The type of the returned variant depends on the parameter type.

The following types can be used:

`ptString - VT_BSTR ptInteger - VT_I4 ptBoolean - VT_BOOL ptDouble - VT_R8 ptDateTime - VT_DATE ptBinary - VT_ARRAY | VT_UI1 ptXML - VT_ARRAY | VT_UI1`

Parameter:

[out]: `pVal` (VB return parameter) contains the value of the parameter

§ `BSTR XML [get, set]`

`XML` returns the value of the Base64 parameter as an XML string and sets the value of the Base64 parameter based on the XML string to be transferred.

This method is only available for parameters of the types `ptBinary` and `ptXML`.

The output XML string is read out with the XML parser.

If no valid XML string can be generated from the data, an error is sent. The error message is related to the XML parser used for the validation (MS-XML4).

The position pointer of the `Stream` remains unchanged when invoking this property.

Parameter:

[out]: `pVal` (VB return parameter) contains the data decoded as an XML string.

Example:

VB

```
Option explicit
Dim oServer, oSession, oJob, oInputParameters, oByteParameter
Set oServer = CreateObject("OxsvrSpt.Server")
Set oSession = oServer.Login()
set oJob = oSession.NewJob("med.ObservationValues")
Set oInputParameters = oJob.InputParameters
set oByteParameter = oInputParameters.AddNewByteParameter("Parametername")
oByteParameter.XML = "<?xml version='1.0' encoding='utf-8'?>" + _
"<med>any test with umlauts äöüß</med>"

oByteParameter.ResetStream
' Read data out again using MSXML
Dim oDocument, bSuccess
Set oDocument = CreateObject("MSXML.DOMDocument")
bSuccess = oDocument.Load(oByteParameter.Stream)
' Read out and output XML text from the DOM
MsgBox oDocument.XML
```

This property is only available for parameters of the Base64 type.

In the process, the transferred XML string is passed to the `Stream` made available in the parameter object. The specified coding in the XML file is automatically respected.

Old data of the `Stream` will be overwritten when invoking this property. After invoking this property, the position pointer of the `Stream` is at the beginning of it.

The passed data will be validated during conversion. If data non-compliant with XML are passed, an error will be sent. In this case, the `Stream` has the length 0. The error message is related to the XML parser used for the validation (MS-XML4).

Parameter:

[in]: `newVal`	Basic string with XML data.

Example:

VB

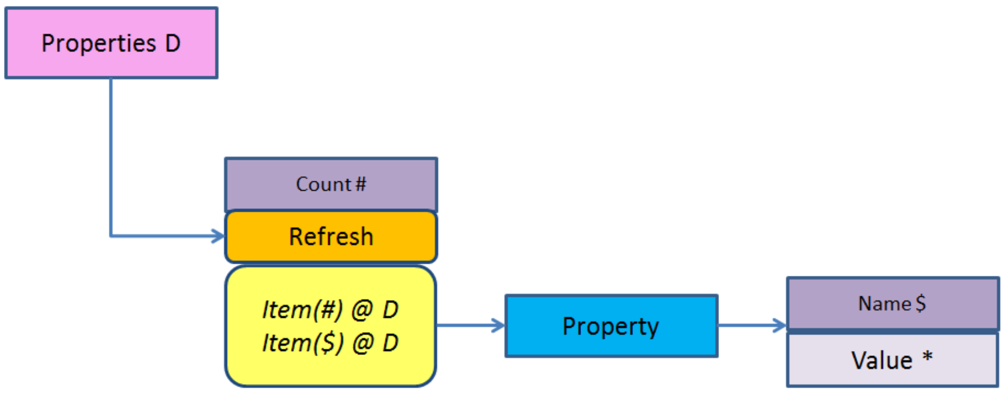
```
Dim oParameter As OxSvrSpt.Parameter
...
oParameter.XML = "<xml version='1.0' encoding='utf-8'?>" + _
"<med>any test with umlauts äöüß</med>"
```

IProperties

Description:

`IProperties` is a collection for properties.

```
import "OxSvrSpt.idl"
```



Public methods:

`HRESULT Refresh ()`

Properties:

`long Count [get]`

`IProperty Item([in] VARIANT Index) [get]`

Documentation of the element functions:

§ `HRESULT Refresh ()`

`Refresh` refreshes the `Property` collection. All entries of this collection will be generated again in the process.

Properties documentation:

§ `long Count [get]`

`Count` returns the number of items of the collection.

Parameter:

[out]: `plNumber`	(VB return value) Number of elements in the collection.

§ `IProperty Item([in] VARIANT Index) [get]`

`Item` returns the specified item of the collection using the key or its position.

If a position is specified outside of the valid index, an error with the error value `errCollectionIndexOutOfRange` is returned. If the item cannot be found, an error with the error value `errCollectionItemNotFound` is returned.

Parameter:

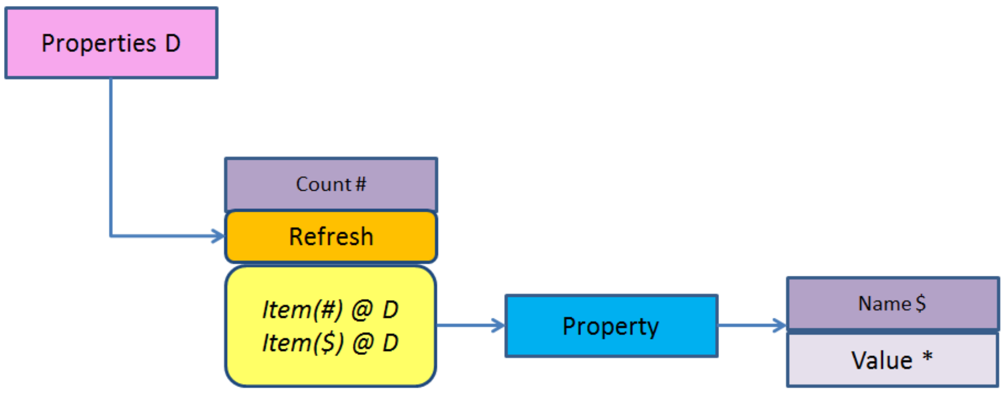
[in]: `Index`	Position and name of the requested element.
[out]: `ppItem`	(VB return value) associated `FileParameter` object.

IProperty

Description:

`IProperty` represents a property.

`import "OxSvrSpt.idl"`



Properties:

```
VARIANT Value [get, set]
```

```
BSTR Name [get]
```

Documentation of properties:

§ `BSTR Name [get]`

`Name` returns a name of the property.

Parameter:

[out]: `pVal`	(VB return value) Name of the property.

§ `VARIANT Value [get, set]`

`Value` returns and sets the value of a property. This property is a default property of the IProperty interface.

Parameter:

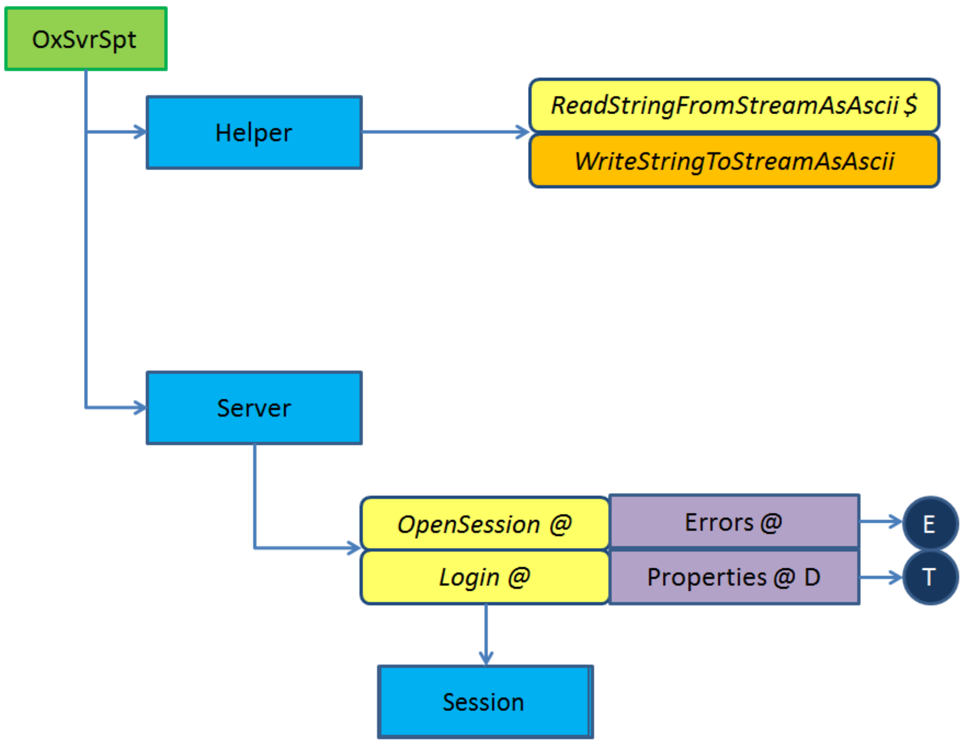
[out]: `pVal`	(VB return value) Value of the property.
[in]: `Val`	Value of the property.

IServer

Description:

`IServer` is the central entry point of the library. It is the only creatable object in the hierarchy for accessing the server.

```
import "OxSvrSpt.idl"
```



Public methods:

```
HRESULT Login ([in, defaultvalue("")] BSTR User, [in, defaultvalue("")]
BSTR Password, [in, defaultvalue("")] BSTR Server, [in, defaultvalue("")]
BSTR Port, [in, defaultvalue(pwNotEncrypted)] PasswortTypeEnum
PasswortType, [in, defaultvalue()] VARIANT_BOOL DefaultSession, [out,
retval] ISession ** ppSession)
```

```
HRESULT Connect ([in, defaultvalue("localhost")] BSTR Server, [in,
defaultvalue("4000")] BSTR Port)
```

```
HRESULT OpenSession ([in, defaultvalue("")] BSTR SessionGUID, [in,
defaultvalue("")] BSTR Alias, [out, retval] ISession ** ppSession)
```

```
HRESULT LoginBalanced ([in, defaultvalue("")] BSTR User, [in,
defaultvalue("")] BSTR Password, [in, defaultvalue("")] BSTR ServerList,
[in, defaultvalue(pwNotEncrypted)] PasswortTypeEnum PasswortType, [in,
defaultvalue()] VARIANT_BOOL DefaultSession, [out, retval] ISession **
ppSession)
```

```
HRESULT LoginGUID ([in, defaultvalue("")] BSTR GUID, [in,
defaultvalue("")] BSTR Server, [in, defaultvalue("")] BSTR Port, [in,
defaultvalue()] VARIANT_BOOL DefaultSession, [out, retval] ISession **
ppSession)
```

Properties:

```
IProperties Properties [get]
```

```
IErrors Errors [get]
```

Documentation of the element functions:

§ `HRESULT Connect ([in, defaultvalue("localhost")] BSTR Server, [in, defaultvalue("4000")] BSTR Port)`

`Connect` establishes a connection to the specified servers.

This method currently returns the error `E_NOTIMPL`.

Establishing a connection to a server without a login is necessary when the server properties are required. After `Connect`, they will be available in `Properties`. If none of these data is required, it is not necessary to establish a connection to the server with `Connect` before logging in. In this case, server data can be specified directly when logging in.

Parameter:

[in]: `Server` (Default value is `localhost`) IP address or name of the server to which the connection will be established

[in]: `Port` (Default value is `4000`) Server port

§ `HRESULT Login ([in, defaultvalue("")] BSTR User, [in, defaultvalue("")] BSTR Password, [in, defaultvalue("")] BSTR Server, [in, defaultvalue("")] BSTR Port, [in, defaultvalue(pwNotEncrypted)] PasswortTypeEnum PasswortType, [in, defaultvalue()] VARIANT_BOOL DefaultSession, [out, retval] ISession ** ppSession)`

`Login` performs a login at the server and returns the related session.

If a connection to the server was already established using the `Connect` method before the `Login` call, the connection data of Connect is used for `Login` in case none were specified.

If both the user parameter and the password parameter are empty when passed or passed with an empty string, an attempt is being made to perform the login automatically using `Login`. The automatic login must be activated in the enaio® administrator and must not be compared to the NTLM authentication.

Parameter:

[in]: `User`	User account
[in]: `Password`	Password of the user account to be used
[in]: `Server`	IP address or name of the server to which the connection will be established
[in]: `Port`	Server port

[in]: `PasswordType` indicates whether the password transferred is already encrypted. If this parameter is not specified, it is assumed that the password was transferred in an unencrypted form.

[in]: `DefaultSession`(default value is VARIANT_FALSE) indicates whether this is the session to which the user can add themselves later using `OpenSession`.

[out]: `ppSession` (VB return parameter) created session for the account.

Exception handling:

`errLoginUnknownUser` (602) The specified user does not exist.

`errLogin3TimesWrong` (603) The third login attempt failed.

`errLoginInvalidPassword` (604) The entered password is incorrect.

`errLoginLocked` (605) The user account is blocked.

§ `HRESULT LoginBalanced ([in, defaultvalue("")] BSTR User, [in, defaultvalue("")] BSTR Password, [in, defaultvalue("")] BSTR ServerList, [in, defaultvalue(pwNotEncrypted)] PasswortTypeEnum PasswortType, [in, defaultvalue()] VARIANT_BOOL DefaultSession, [out, retval] ISession ** ppSession)`

`LoginBalanced` performs a login at the server and returns the related session.

If both the user parameter and the password parameter are empty when passed or passed with an empty string, an attempt is being made to perform the login automatically using `Login`.

Parameter:

[in]: `User` User account

[in]: `Password` Password of the user account to be used

[in]: `ServerList` List with the servers to which the library should connect. This list has the following structure: Server1#Port1#Weighting1;Server2#Port2#Weighting2;

Server3#Port3#Weighting3

[in]: `PasswordType` indicates whether the password transferred is already encrypted. If this parameter is not specified, it is assumed that the password was transferred in an unencrypted form.

[in]: `DefaultSession`(default value is VARIANT_FALSE) indicates whether this is the session to which the user can add themselves later using `OpenSession`.

[out]: `ppSession` (VB return parameter) created session for the account.

Exception handling:

`errLoginUnknownUser` (602) The specified user does not exist.

`errLogin3TimesWrong` (603) The third login attempt failed.

`errLoginInvalidPassword` (604) The entered password is incorrect.

`errLoginLocked` (605) The user account is blocked.

§ `HRESULT LoginGUID ([in, defaultvalue("")] BSTR GUID, [in, defaultvalue("")] BSTR Server, [in, defaultvalue("")] BSTR Port, [in, defaultvalue()] VARIANT_BOOL DefaultSession, [out, retval] ISession ** ppSession)`

`LoginGUID` executes a login using the passed `SessionGUID` at the server and returns the related session

Using the `SessionGUID`, a connection to an existing session at the server is established.

Parameter:

[in]: `GUID` GUID of the server session with which the `Session` object should work

[in]: `Server` IP address or name of the server to which the connection will be established

[in]: `Port` Server port

[in]: `DefaultSession`(default value is VARIANT_FALSE) indicates whether this is the session to which the user can attach themselves later using `OpenSession`

[out]: `ppSession` (VB return parameter) created session for the account

Exception handling:

`errLoginLocked` (605) The user account is blocked.

§ `HRESULT OpenSession ([in, defaultvalue("")] BSTR SessionGUID, [in, defaultvalue("")] BSTR Alias, [out, retval] ISession ** ppSession)`

`OpenSession` creates a new `Session` object and connects it to the `DefaultSession`.

Parameter:

[in]: `SessionGUID`	GUID of the existing session with which the connection will be established
[in]: `Alias`	Any name for the call in order to assign errors or states
[out]: `ppSession`	(VB return parameter) created session for the account

Documentation of properties:

§ `IErrors Errors [get]`

`Errors` returns the error collection with errors occurred during server access.

Parameter:

[out]: `pVal` (VB return value) Error collection

§ `IProperties Properties [get]`

`Properties` returns the collection with the server properties

Currently, the following properties are set by the OxSvrSpt library when creating the server object:

TempDir:

Contains the temporary directory in which file parameters are stored. This directory is initialized together with the temporary directory of the user.

NotifyNeeded:

is initialized with 0.

If the value is set to a value that is not 0 (or VARIANT_FALSE), the notification support is activated. In that case, it is possible to process notifications with the event interface of the server and of the session object.

The specified identifiers correspond to the parameter names.

Parameter:

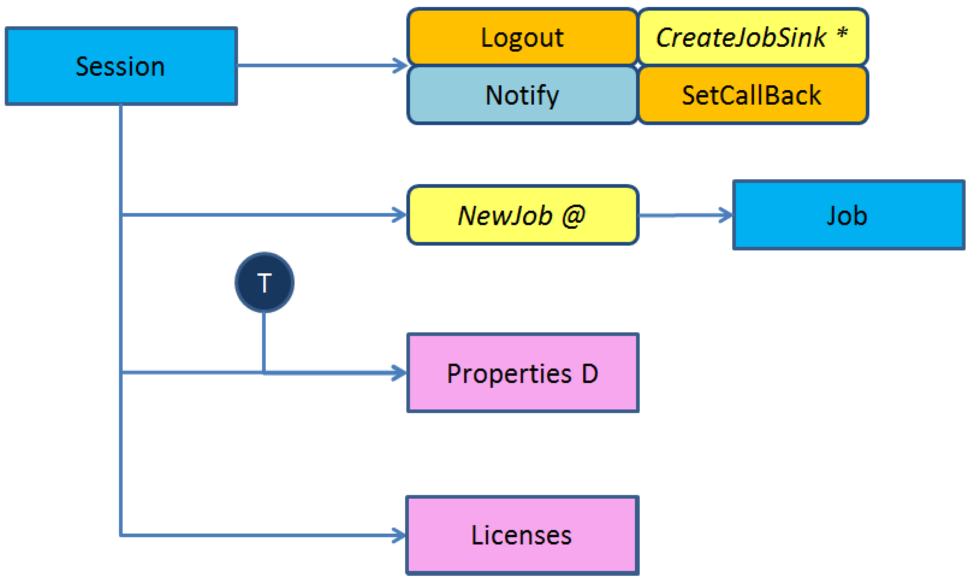
[out]: `pVal` (VB return parameter) `IProperties` interface

ISession

Description:

`ISession` is provided by the `Server` interface after a successful login. Session represents an open connection to a server. It allows server job calls.

```
import "OxSvrSpt.idl"
```



Public methods:

`HRESULT Logout ()`

`HRESULT NewJob ([in] BSTR Name, [out, retval] IJob ** ppJob)`

`HRESULT CreateJobSink ([out, retval] VARIANT * pJobSink)`

`HRESULT SetCallBack ([in] long ICallbackType, [in] IDispatch * pUnkSink, [in] long IUserData)`

Properties:

`IProperties Properties [get]`

`ILicenses Licenses [get]`

Documentation of the element functions:

§ `HRESULT CreateJobSink ([out, retval] VARIANT * pJobSink)`

`CreateJobSink` returns the `IJobSink` interface which is based on the OxSvrCom library

A detailed description can be found in the documentation of the OxScrCom.dll library.

Parameter:

[out]: `pJobSink`	(VB return parameter) Created `IJobSink` interface

§ `HRESULT Logout ()`

`Logout` disconnects the current `Session`.

After a `Session` logout, every access to other methods and properties of the `Session` will produce the `errNoSession` error.

Exception handling:

`errNoSession` (1501) The respective `Session` is no longer available. This `Session` was either closed or the session object was released.

§ `HRESULT NewJob ([in] BSTR Name, [out, retval] IJob ** ppJob)`

`NewJob` generates a new job object with the passed name for the job

Parameter:

[in]: `Name` Job name. The name consists of the namespace of the job.

[out]: `ppJob` (VB return parameter) contains the created Job object

- `HRESULT SetCallBack ([in] long ICallbackType, [in] IDispatch * pUnkSink, [in] long IUserData)`

  `SetCallBack` sets the `IJobSink` interface for callbacks to the related OxSvrCom library

  A detailed description can be found in the documentation of the OxScrCom.dll library.

  Parameter:

[in]: `ICallbackType` Documentation `ICallbackType`

[in]: `pUnkSink` Documentation `pUnkSunk`

[in]: `IUserData` Documentation `IUserData`

Documentation of properties:

- `ILicenses Licenses [get]`

  `Licenses` returns the collection of currently registered licenses.

  Parameter:

[out]: `pVal` (VB return value) delivers a COM collection with the licenses

- `IProperties Properties [get]`

  `Properties` returns the server and session properties of this object.

  Parameter:

[out]: `pVal` (VB return value) returns a COM collection with the associated properties. This collection contains objects of the `IProperty` type

Class Hierarchy

- `_INotificationEvent`
- `IError`
- `IErrors`
- `INotifyErrors`
- `IFileParameter`
- `IHelper`
- `IInputFileParameters`
- `INotifyOutputFileParameters`
- `IInputParameters`
- `INotifyOutputParameters`
- `IJob`
- `ILicenses`
- `ILogger`

- `INotifyJob`
- `IOutputFileParameters`
- `INotifyInputFileParameters`
- `IOutputParameters`
- `IInputParameters`
- `IParameter`
- `IProperties`
- `IProperty`
- `IServer`
- `ISession`

Index