Kofax Communication Server

TCSI Technical Manual - Client Server Interface

Version: 10.2.0

Contents

1 Introduction

1.1 How to Read This Manual

Advice to first time readers:

To make best use of your time – do not read this manual from beginning to end!

Follow this sequence:

- Read the overview.
- In the section Class Reference read 'General' and the class descriptions for classes Object, Container, Set and List. This will give you a good basic understanding of the objects.
- Read the Structure section starting with the object SET_OBJ_HANDLER and follow the hierarchy from left to right. Consult the class reference for each new type of object you encounter.

When you follow this sequence you have read the full manual.

Even when the concepts may seem unfamiliar at the beginning, you will soon discover the huge potential the interface has to simplify your application programming.

Your comments on product improvement are always welcome!

Legend

Unclear, to be defined paragraphs

Internal – doc only paragraphs

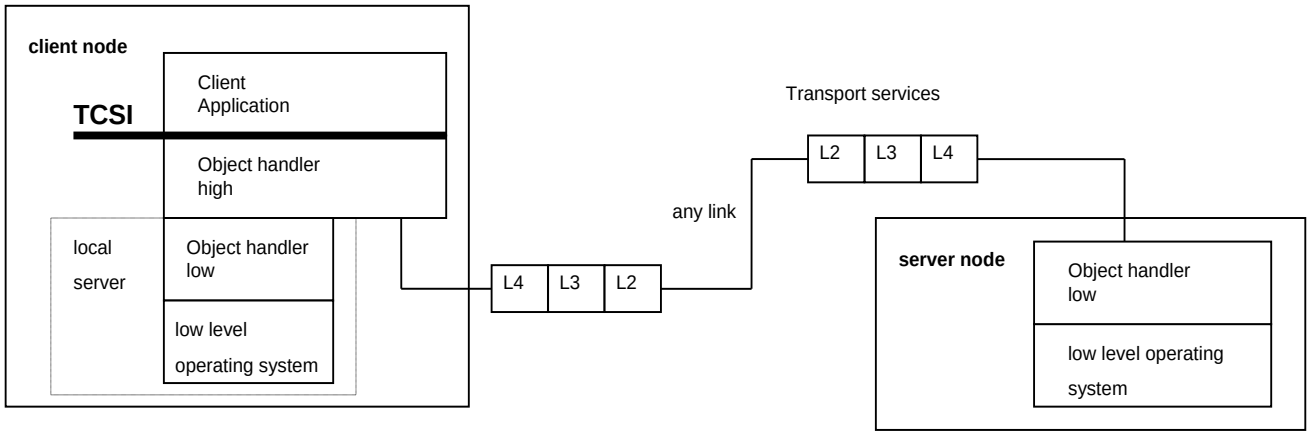
1.2 Overview

In a client / server environment software modules (applications) running on different systems have to access server objects. To gain easy and well defined access to these objects the TCOSS Client Server Interface (**TCSI**) has been defined.

TCSI is the interface between an application and the TCOSS object handler. The TCOSS object handler is running under different environments (written in ANSI C). This object handler shields all performance, compatibility and transport issues from the application.

The application only has to deal with this interface for any kind of server communication (a server can be a TCOSS server, a LAN server or the local workstation).

Location of the interface:



Idea of operation:

The interface encapsulates all the server functionality in a hierarchical structure of objects.

You can go down in the hierarchy by opening the objects in sequence, from high to low level. You can change objects at any level. When you have changed an object, you can either forget it (all the changes will be lost) or close it (this will update its parent).
Normally when you close an object you can still forget the changes by forgetting the object at the next higher level. This provides an easy way out for (user-) cancel operations.

Care has been taken to limit the total number of function calls. There are only a few base classes of objects that define standard functions. So, for example to open a server session, you simply use the standard c_open function to open the server session object.

2 Structure of the Interface

For an overview of the object hierarchy please refer to the separate HTML documentation “tcsi_nnnnn_obj.htm”, where nnnnn stand for the version number, e.g. “tcsi_25100_obj.htm”.

2.1 Prerequisites

2.1.1 Multiprocess and Multithread Safety

2.1.1.1 32-Bit Version

The 32-bit TCSI version reads its configuration parameters from an application-specific key in the registry. Therefore, all TCSI settings are application-specific as long as every instance of an application has its own key within the registry. In this case session data is private and the applications do not interfere with each other.

To get access to the application specific registry key, TCSI uses the functions exported from TCLIB32.DLL. In addition, tracing capabilities are changed to the TCLIB32 standard. For compatibility all trace level parameters TraceLevel, DebugLevel, and DebugSwitch are supported.

All 32-bit applications using TCSI must load and initialize the TCLIB32.DLL!

The new 32-bit TCSI version does not use global variables and critical data gets locked before being changed to prevent any corruption.

Restriction: TCSI does not allow multiple threads to access the SAME object at the SAME time. This would require locking at the object level which is not implemented. Locking is performed at the system level, i.e. internal global data structures, handle tables, memory management is thread safe.

2.1.1.2 16-Bit Version

16-bit TCSI versions do not allow multiple processes to concurrently use TCSI on one PC. This restriction exists because TCSI tries to reuse the session number and session id of the last user session to the server. For this purpose it writes those values into the INI file after being received from the server. There is only one INI file for all applications using TCSI. Thus, if multiple applications use TCSI simultaneously, they all use the same session values from the INI file. This may lead to strange effects when logging into the server.

2.2 Accessing the Server

As the starting point, to work with the interface a predefined handle to the object handler (H_OBJ_HANDLER) always exists. With this handle you can first read out the program version of the interface to check for compatibility, and then open an application session. An application session is needed to create objects (it is the only type of object that provides an object create function).

You now can create server entries and fill in the server name, path, userid and password. Then you create the list of servers and save your server entries to it.

Next, you save the list of servers to the application session.

Within the server entry you now can open the server session. A server session gives you direct access to the server's objects.

To end your application session, first close() or forget() all objects you have open. Then (as a check for your application) test if the count of open objects in the application session object is zero and forget () the application session.

Example of main program (without error handling):

```
// First, obtain an application session (you need it to create objects), and
// set your application id.
ohh_c_open    ( H_OBJ_HANDLER,    SET_APPL_SESSION,    &h_app_session, &type );
ohh_c_ts_put   ( h_app_session,    TS_APPL_ID,       "Gateway ABC 1.0" );


/ Create a server entry and fill it with data.
ohh_c_create ( h_app_session, SET_SERV_ENTRY, &h_serv_entry );
ohh_c_ts_put   ( h_serv_entry,    TS_SERV_ID,    "TOPCALL1" );
ohh_c_ts_put   ( h_serv_entry,    TS_PATH,       "TOPCALL1" );
ohh_c_ts_put   ( h_serv_entry,    TS_USER_ID,    "BILL" );
ohh_c_ts_put   ( h_serv_entry,    TS_PASSWORD,   "1234xyz" );


/ Open the server session. When opened, you can access objects at the server.
ohh_c_open    ( h_serv_entry,    SET_SERVER_SESSION,    &h_serv_session, &type );


/ ... work with your server session
// ...


/ Forget your server session
ohh_o_forget   ( h_serv_session );


/ Check if there are no objects left, then forget your application session.
ohh_c_int_get ( h_app_session,   INT_OPEN_OBJECTS, &obj_count );
if (obj_count != 1 ) { print error message };
ohh_o_forget   ( h_app_session );
```

3 Class Reference

3.1 General

The interface is defined on the basis of objects.

Objects that have the same appearance and behavior are said to be of the same **class**.

The classes are defined in a hierarchy, with the most common class defined on top. Classes that are defined on the basis of another class 'inherit' the features of their base class. As you follow the hierarchy down, the classes become more and more specialized. See diagram below.

For each class the concepts, the functions, the class-general errors and the child attributes (if any) are defined.

The function prototypes can be found in TCSI.H.

Base classes / Real classes:

Base classes (gray background) define all the functions of a class. Objects of base classes do not exist.

Real classes (white background) are derived from base classes. Objects of real classes do exist.
Properties such as the types of child objects an object may hold, default values for child objects, maximum size, etc. are defined for real classes.

Inheritance:

A derived class inherits all the properties of its base class. Additionally, it may add new functions, redefine existing functions, or add and redefine other properties, such as types of child objects, defaults, etc.

Note: From a base class you inherit just the functions and the general errors; from a real class you also inherit its child objects and defaults.

Function redefinition:

When a function exists in a base class but changes its meaning in the new class this is called a 'redefinition'.
Redefined functions are printed in italic.

Types, Names & Defaults:

Type:

In the following the **'type'** of an object defines its real class.

Names:

To select child objects of certain containers **'names'** are used. (See class SET.) The names are local to the type of the container. When you take an object out of its container it keeps its type but looses its name.

Normally, by convention, the name begins with the type of the child.
For child objects that may be of different types the names start with 'UN_' for union.

Internally types and names are represented by integer values. They are both defined in TCSI.H.

Default values:

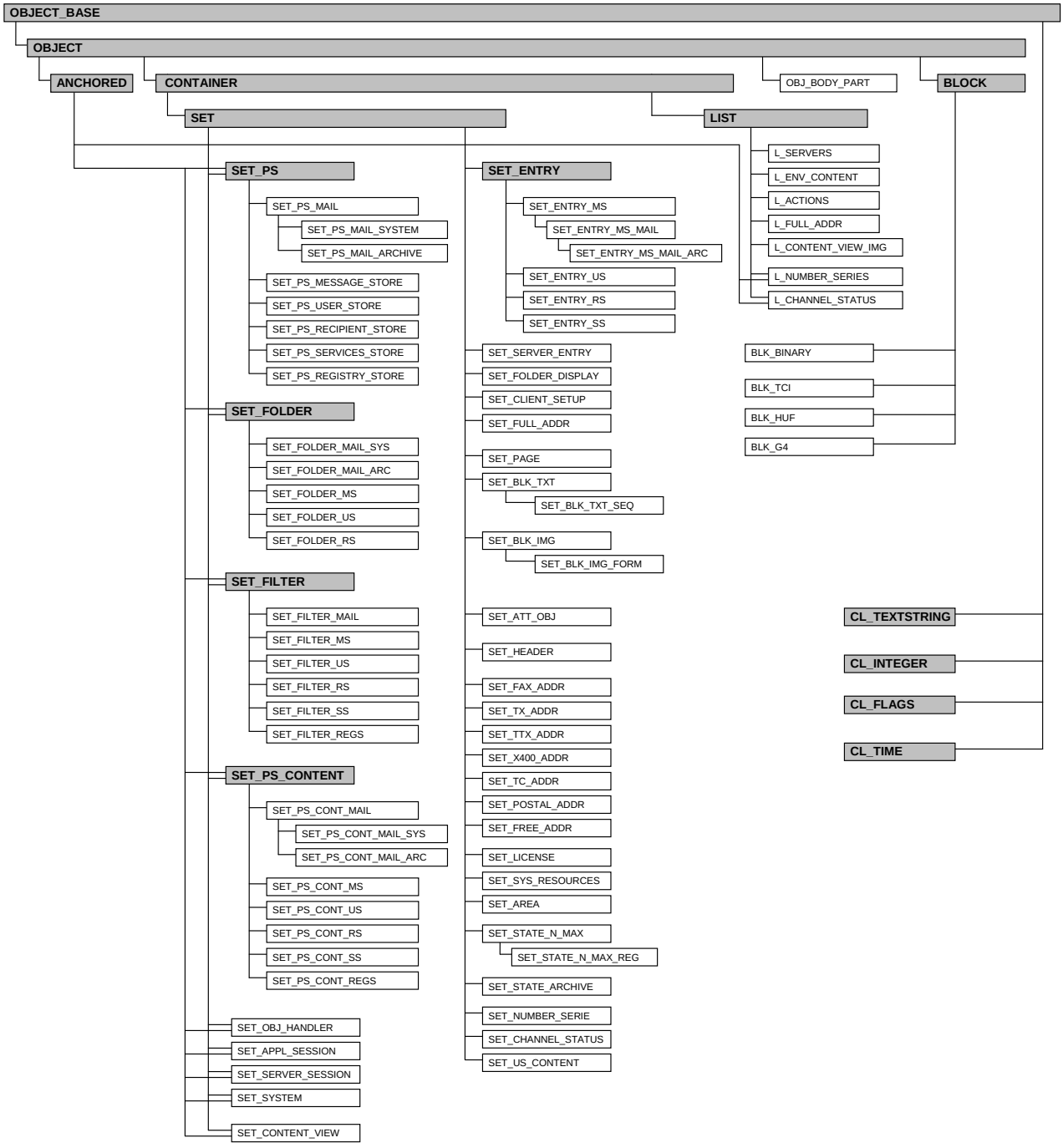
The container class SET defines default values for child objects of types integer, time and string. This makes handling of the interface simpler – you only have to set those values that are non-default.

Default values are defined in TCSI.H.

Permanent objects:

Some of the objects in the hierarchy have child objects that are 'permanent'. When you update a permanent object (for example you open, change and close it) the object will be changed permanently (even if power is lost). With permanent objects there is no way to make the changes undone by forgetting the parent.

Class hierarchy:



3.2 OBJECT_BASE

It is the base class for all other classes. It defines that objects have a **type**.

3.3 OBJECT

Defines the general behavior of objects on which a handle can be obtained.

Objects can be:

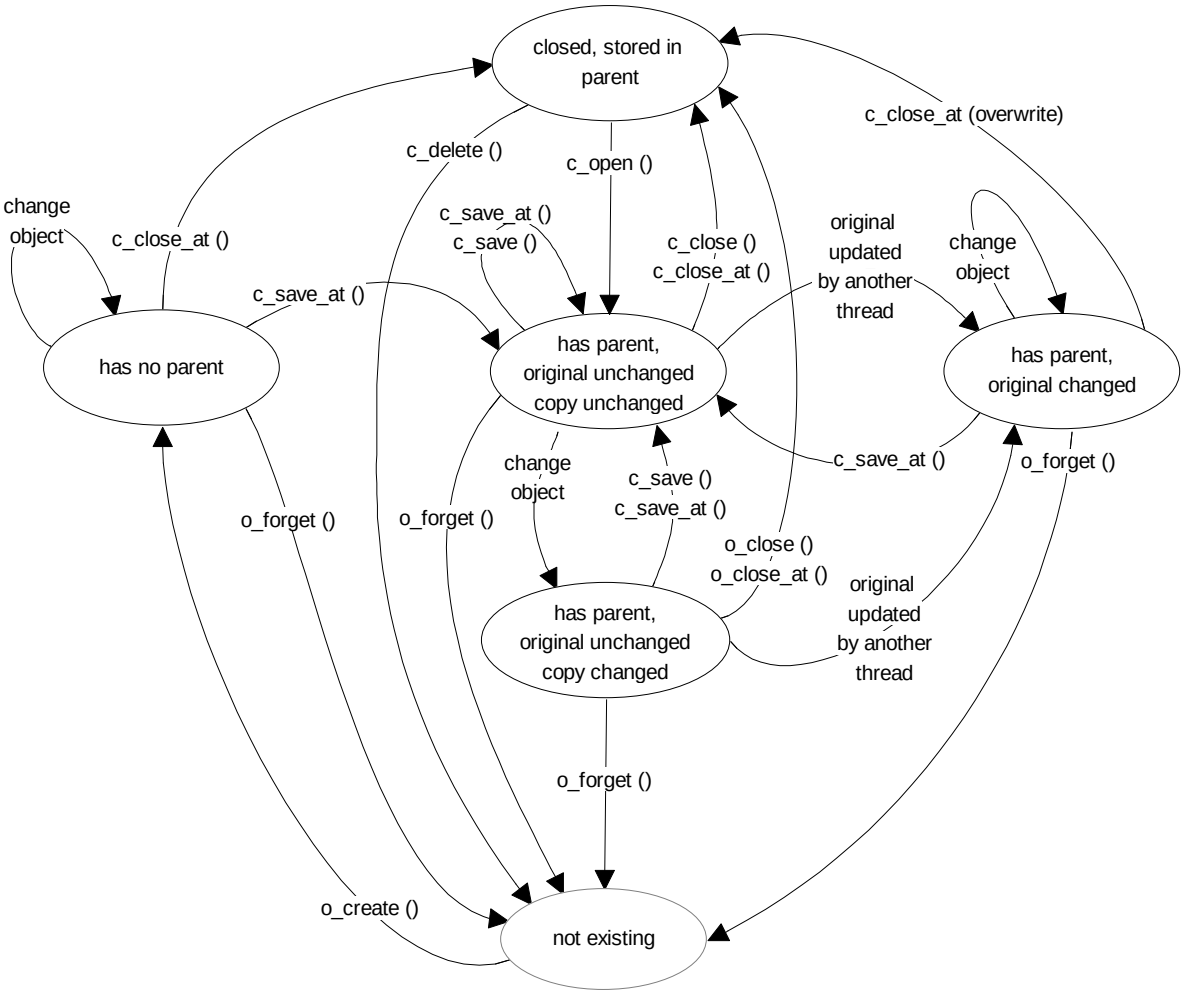
Permanent:	These objects are permanently stored at the server. There are no handles to permanent objects.
Temporary:	All objects of which you have a handle are temporary ones. You can create a temporary object by **creating** or by **opening** it.

Object states:

When you **create** an object you get a handle to a new object of the type you requested. The object is open after creation.

When you **open** an object you get the handle to a copy of the original one. You can now work with your copy (read it, change its contents, replace its child objects, etc.).

When you call **forget ()** your object is forgotten and any changes to it are not stored (if the object was opened, the original still exists).
With **close ()** or **close_at ()** you close your object and update the old (or a newly specified) parent object.
When you close an object its handle becomes invalid.



Note on event 'Original updated by another thread':

This happens if, e.g., an envelope was opened by one user and then changed and updated by another user. The updating changes the state of the object for the first user. The first user will not notice the state change until he tries to save the object – saving is not permitted anymore. The user has to decide whether to overwrite the original object or save at another position.

General for all functions:

Return codes:

These return codes may happen with any object operation:

OK	the operation was OK
ERR_NOT_PERMITTED	the rights of the caller do not permit this operation
ERR_INVALID_HANDLE	the specified handle is not valid anymore
ERR_OUT_OF_MEMORY	the operation could not be completed, close other objects

See TCSI.H for additional error codes.

Note: The constant decimal 1000 is added to the error code for errors that originate at the server.
Example: 301 (err_notfound at the client), 1301 (err_notfound at the server)

Function names:

The names of all functions exported by TCSI.DLL are prefixed with 'ohh_' (e.g. ohh_ o_forget(..)).

ohh_o_forget (h_obj)

Forgets the object.
Sets the handle to invalid.

Note:
When child objects are open they are **not** automatically forgotten.
When an object is forgotten with child objects still open, those child objects loose their parent (see ERR_NO_PARENT) but they stay open.

errors: none [1]

ohh_o_gettype (h_container, ptype)

Returns the type of the object.

errors: none

ohh_c_int_get(h_obj, INT_H_APPL_SESSION, ph_apps)

Returns the handle of the application session associated with the object.

errors: none

ohh_o_close (h_obj, bool_overwrite)

Only possible if the object has a parent. Closes the object, saves all changes (updates the original in its parent).
Sets the handle to invalid.

Note:
If child objects are open they are **not** automatically closed. If you want to save the changes in child objects you have to **close()** or **save()** them before closing they parent.
When an object is closed with child objects still open, the child objects loose their parent (see ERR_NO_PARENT).

errors:	ERR_NO_PARENT	The object currently has no parent - it can not be closed. You can either **forget()** the object or do a **close_at()**.
	ERR_CHANGED	The parent has been changed since the **open()** that gave the specified object handle. You can either **forget()** the object or do a **close (... ,overwrite = TRUE)**.
	ERR_OTHER_UPDATING	Only returned with overwrite. Another task is saving or closing to the specified child. This is a temporary condition. You may try again.

ohh_o_save (h_obj, bool_overwrite)

Only possible if the object has a parent.
Closes the object, saves all changes (updates the original in its parent).
The handle stays valid.

---

[1] In each case all the general errors for this class and its base classes are possible.

Note:
If child objects are open they are **not** automatically saved. If you want to save the changes in child objects you have to **close()** or **save()** them before saving they parent.

errors:	ERR_NO_PARENT	The object currently has no parent - it can not be saved. You can either **forget()** the object or do a **save_at()**.
	ERR_CHANGED	The parent has been changed since the **open()** that gave the specified object handle. You can either **forget()** the object or do a **save (... ,overwrite = TRUE)**.
	ERR_OTHER_UPDATING	Only returned with overwrite. Another task is saving or closing to the specified child. This is a temporary condition. You may try again.

ohh_o_getid (h_obj, pid)

Only possible if the object has a parent.
Returns the id of the object within its container. The id may be the index of a list or the name within a set.

errors:	ERR_NO_PARENT	The object currently has no parent.

3.3.1.2 OBJ_BODY_PART

Defines an object that is a mark in the list of envelope contents. See L_ENV_CONT.

3.4 ANCHORED

It is a base class that restricts the general functions on objects.

'Anchored' objects exist only at a fixed position in the hierarchy. They always exist at their position.

You can not **create(), close(), save(), close_at()** or **save_at()** anchored objects.

You can not stream anchored objects (neither ASCII stream nor binary block access is permitted).

You can only **open()** them from their parent, use them and **forget()** them.

When an object that contains anchored sub objects is saved the anchored sub objects are not saved.

general errors: ERR_NOT_PERMITTED for any of the above functions

3.5 CONTAINER

Is the base class for all objects that may hold other objects. [2]

ASCII representation:

```
value : ( [ object ] { , [ object ] } )
```
[a]

a) See class BLOCK for general description of ASCII representation.

General attributes for child objects:

R/W	**Read / Write**	Is the standard attribute. Is treated in the following as no attribute.
R/O	**Read only**	When opened the object has no parent. The opened object may be changed. It is possible to store the object under another name or to another parent. It is not possible to save_at() or to close_at() any objects to R/O child objects.
W/O	**Write only**	This child in the container can not be opened, it only accepts c_save_at () or c_close_at () operations.
D/O	**Delete only**	This child in the container can not be opened, and does not accept save_at() or c_close_at(). It only accepts c_delete () operations. (see registry store)

Note: The current implementation of TCSI does not directly check the attributes. Example: you may write to read only childs - but if you update such an object at the TCOSS server (in a permanent store) the R/O childs are ignored.

eneral errors: one

ohh_c_exist (h_container, id, ptype)

Allows testing if the specified child object exists. Returns the type of the child if it exists.
A child exists if, since creation, an object was put to the child and it has not been deleted since.
When a write only child is checked for existence it does not exist.

[2] The class CONTAINER inherits all the functions of class OBJECT.

errors: ERR_OBJ_NOT_EXIST the sub object does not exist

ohh_c_delete (h_container, id)

Deletes the specified child object from the container. Returns also OK if the child did not exist.
If opened copies of the target exist, their original is set to changed.

errors: **ERR_OTHER_UPDATING** **Only with permanent objects. The object can not be deleted – it is currently updated by another task.**

ohh_c_open (h_container, id, phandle, ptype)

Creates a copy of the child object and returns a handle to it. The type of the object is also returned. You can work with the copy now. For any open you have to free the handle with either **ohh_o_forget()**, **ohh_o_close()** or **ohh_c_close_at()**.

Note1:
An object can be opened more than once at the same time. For each open a separate handle is returned. When one of those copies is closed or saved, the original is set to changed (see ohh_o_close()). This prevents loss of data from competing update.

Note2:
Integer, time and textstring child objects can not be opened, they are accessed directly by container functions.

errors: ERR_OBJ_NOT_EXIST the sub object does not exist

ohh_c_save_at (h_container, id, h_obj, bool_overwrite [3])

Saves the object as the specified child in the given container. The h_obj stays valid.
This function sets the position for all following save() operations.
If open copies of the target exist their original is set to changed.

errors:	ERR_OBJ_EXISTS	Is returned if not overwrite and the specified child already exists.
	ERR_OTHER_UPDATING	Only returned with overwrite. Another task is saving or closing to the specified child. This is a temporary condition. You may try again.

ohh_c_close_at (h_container, id, h_obj, bool_overwrite)

Saves the object as the specified child in the given container. Closes the object.
The h_obj is set to invalid.
If open copies of the target exist their original is set to changed.

errors: see ohh_c_save_at ()

ohh_c_getid(h_obj, index, pID)

This function allows sequential processing of SETs (and LISTs). It returns the ID of the child object at index *index.* *Index* = 0,1,2,.. If *index* is < 0 or >= number of child objects ERR_OBJ_NOT_EXIST is returned.

[3] bool_ parameters may be TRUE or FALSE (as defined in TCSI.H)

CL_INTEGER, CL_FLAGS, CL_TIME and CL_TEXTSTRING:

These types of child objects cannot be opened. They can only be accessed by the following container functions.

The values are set by **put** functions. Put functions create a child object if it did not exist before.

The values are read by **get** functions. Get functions return the default value if the child is not existing.

CL_INTEGER, CL_FLAGS:

For integers and flags the container defines (see TCSI.H for the definitions):

- the default value	(Not implemented in this release of TCSI (default always = 0), default values consistent with TCSI-definition are set/assumed by TOPCALL server when required.)
- value restrictions	(Not implemented in this release of TCSI, TCOSS server may reject objects that do not conform to restrictions.)

Note: The only difference between integers and flags is the filtering of entries where integers can be filtered by upper and lower bound, flags can be filtered by active bits and mask. (see SET_FILTER).

ASCII representation:

value : [-] decimal-integer (2^31-1 ... -2^31)

ohh_c_int_put (h_container, id, value)

Puts the integer to the specified child.
The type of the child object must be INT.

errors:	ERR_WRONG_TYPE	child is not of type INT
	ERR_WRONG_VALUE	value not permitted for this child of container

ohh_c_int_get (h_container, id, pvalue)

Gets the integer from the specified child.
The type of the child object must be INT.

errors:	ERR_WRONG_TYPE	child is not of type INT

CL_TIME:

For times the container defines (see TCSI.H for the definitions) the default value

ASCII representation:

see textstring, the format is YYMMDD:hhmmss

ohh_c_time_put (h_container, id, pbuffer, format)

Puts the time to the specified child.
The type of the child object must be TIME.
For formats, see below.

errors:	ERR_WRONG_TYPE	child is not of type TIME
	ERR_INVALID_TIME	not a valid time value

ohh_c_time_get (h_container, id, pbuffer, format)

Gets the time from the specified child.
The type of the child object must be TIME.

errors: ERR_WRONG_TYPE child is not of type CL_TIME

Format constant	Value	Meaning
TIME_INTEGER	0	Time as integer (seconds since 1993-01-01 00:00)
TIME_ASCII	1	Time as string “yymmdd:hhmmss”
TIME_UTC	16	Flag: time in UTC
TIME_TCOSS	32	Flag: TCOSS time (no conversions)
TIME_MAIL	64	Flag: message time zone (only for time stamps in mail entry)

The TIME_UTC flag allows the client to retrieve or set UTC times.

The TIME_TCOSS flag temporarily disables all conversions (to access the actually stored value). Setting this flag will never cause the function to fail.

The TIME_UTC and TIME_TCOSS flags may be used in two cases:

a) To do explicit conversion of time stamps by writing and reading the TIME_ACTION child of a SET_TIME_ZONE object.
b) To get UTC or unconverted TCOSS time from any time child object while the automatic time zone conversion feature is active. Calls with the TIME_UTC flag will fail if the automatic time zone conversion feature is inactive, either because “AutoTimeZone” is 0 or the connected TCOSS server is not based on UTC.

In both cases local time is selected by specifying no format flag (neither TIME_UTC nor TIME_TCOSS).

The TIME_MAIL flag may only be used for time stamps in a mail entry, e.g. TIME_ACTION or TIME_CREATED. It selects the time zone of the message. If no message time zone is found in the mail entry this flag is ignored.

CL_TEXTSTRING:

Code:

All strings on the interface are in TCOS code. [4]
String conversion from TCOS code to client codepages and vice versa has to be done by the application.
No information on the server's TCOS codepage (standard / eastern) is provided with this version of the interface.

For strings the container defines (see TCSI.H for the definitions):

the maximum length	
the default value	(Not implemented in this release of TCSI (default always empty string), default values consistent with TCSI-definition are set/assumed by TCOSS server when required.)
character set restrictions	code: 1 - 255 decimal

ASCII representation:

```
value : "string content"
```

Character codes below 20h are represented as escape sequences: backslash, 2 digits hex;
backslash is represented as `"\\"`

String values may be broken into multiple lines:

```
"string content part1"+,
"string content part2"
```

ohh_c_ts_put (h_container, id, psz_string) [5]

Puts the given zero terminated string to the specified child object. When the string is longer than the maximum object size it is truncated.
The type of the child object must be TS.

errors:	WRN_TRUNCATED	the string was longer than the maximum object size, the operation is completed anyway
	ERR_WRONG_TYPE	child is not of type TS

The ohh_c_ts_put() function had an undocumented functionality in previous releases which has been removed: If the string passed to ohh_c_ts_put() was too long and had to be truncated, with WRN_TRUNCATED as return value, the string end was adjusted to avoid that half of a double-byte character remained at the end. To do this it was assumed that the TCOSS code page was identical to the current Windows ANSI code page.

String truncation is now done according to the TCOSS code page set in INT_CODEPAGE in SET_APPL_SESSION. If the TCOSS code page is not set there a single byte code page is assumed and no extra handling is done after truncation. It is therefore recommended to always set the INT_CODEPAGE value in SET_APPL_SESSION appropriately.

ohh_c_ts_get (h_container, id, psz_string, pmaxsize)

Gets a zero-terminated string from the specified child object. When the object is too large the string is truncated to fit into a maxsize buffer (maxsize includes the terminating zero).
If the child at the given position does not exist the default string is returned.
The type of the child object must be TS.

errors:	WRN_TRUNCATED	the string did not fit into maxsize
	ERR_WRONG_TYPE	child is not of type TS

[4] see TCOS manual 5.20 or higher for reference

[5] psz: abbreviation of pointer to zero terminated string

ohh_c_ts_length (h_container, id, psize, pmaxsize)

Gets the current size and the maximum size of a string (sizes include the terminating zero).
If the child at the given position does not exist the length of the default string is returned at psize.

errors: ERR_WRONG_TYPE child is not of type TS

```
ohh_ts_maxlen      (
  TCSI_TYPE parent_type,  /* in: parent type for string lists / 0  */
  TCSI_ID   child_id)     /* in: TS_... constant                   */
```

Returns the maximum length of a string object or 0 if unknown. The parent type is only required for string lists.

Unicode string functions

Two functions, ohh_c_ts_getW() and ohh_c_ts_putW(), allow to read and write strings using wide character (16-bit) Unicode strings in UTF-16 encoding. These functions also work if TCOSS is running on a code page other than UTF-8, the necessary conversions are done internally.

The TCOSS system code page has to be set globally in the TCSI application session (new child INT_CODEPAGE in SET_APPL_SESSION) for the internal code conversions to work. If INT_CODEPAGE is 65001 (Unicode as UTF-8 used on interface) also set the legacy code page INT_CODEPAGE_LEGACY in SET_APPL_SESSION to the value read from the server session. The legacy code page is only used when creating new user profiles or when changing the password of an existing user.

```
TCSI_RET ohh_c_ts_getW(
         HOBJ hobj,         /* in: handle to TCSI object   */
         TCSI_ID ID,        /* in: child ID                */
             WORD2 *pstr,       /* out: UTF-16 string          */
         INT4 cchstrmax,    /* in: max. Unicode string length incl. term. 00
                               in 2-byte units   */
                   LPINT4 pcchstr)   /* out if != NULL: length of string excl. term. 00
                                        in 2-byte units */


TCSI_RET ohh_c_ts_putW(
         HOBJ hobj,         /* in: handle to TCSI object   */
         TCSI_ID ID,        /* in: child ID                */
   CONST WORD2 *pstr)       /* in: UTF-16 string, 00-term. */
```

Both functions return ERR_NOT_POSSIBLE if no code page is set in the application session. Note that the ohh_c_ts_getW() function expects the input parameter "`cchstrmax`" to give the size of the provided Unicode buffer in 2-byte units, and this size includes the terminating zero. It returns the length of the written Unicode string in 2-byte units, not including the terminating zero. The ohh_c_ts_getW() function always writes a terminating zero.

Note:

Seen from a container all the following functions do the same. They are sometimes referred to as **put** operations in the following:

ohh_o_close (), ohh_o_save (), ohh_c_close_at (), ohh_c_save_at ()

3.6 SET

Defines an object that holds a set of other objects. [6]
The members of the set must all have a different name.

[6] The class SET inherits all the functions of class CONTAINER.

general errors:

ERR_WRONG_NAME	This type of set does not have a sub object with the given name. (This is different from the condition that the child object with the name given is known within this set, but does not exist.)

ohh_*c_any function ( )* [7]

For class SET in all functions of container the **id** parameter is replaced by the parameter **name**.

errors: none

3.6.1.2 SET_OBJ_HANDLER

The object handler itself. This object is always open. A fixed handle to it (H_OBJ_HANDLER) exists.

Child objects:	**attributes**	**default**	**value-restr**
SET_APPL_SESSION	-	-	-

The application session. This object always exists. It can be opened more than once (as any child). When opened the application session is empty.

Restriction: Only 1 application session can have a server session. The other application sessions may be used to create temporary objects without accessing a server.

Note: TCSI does not check if the restriction, that only one application session may access a server, is observed. The applications using TCSI may get unwanted results if they do not meet the restriction.

TS_WORKST_DESCR	R/O	-	-

The description of the workstation. Defined during workstation installation (setup).

3.6.1.3 SET_APPL_SESSION

This is the object that has to be opened first by the application to attach to the interface.
It holds the list of servers, the userid, password, application id.

Additionally, for testing, the current number of open objects per application session is provided. This helps to locate missing close() or forget () statements in the application.

The application session is anchored. It can only be opened from the SET_OBJ_HANDLER object.

Child objects:	**attributes**	**default**	**value-restr**

[7] All the functions of class CONTAINER are redefined here.

L_SERVERS	–	–	–

The list of server entries.

TS_APPL_ID	–	–	–

The name and the version of the application. This information is recorded in the server at login.

INT_CLIENT_TYPE	–	CT_TCFW	CT_TCFW / CT_TCFXDPRO/ CT_TCTRANS

The type of the client. This information is recorded at the server at login.

INT_LICENSE_TYPE	–	CT_TCFW	CT_TCFW / CT_TCFXDPRO/ CT_TCTRANS

The type of license accessed, different from the client type only in case of a link or archive server. Default: INT_CLIENT_TYPE value.

TS_CPU_NO	–	–	–

The CPU number of a link or archive server.

INT_MAX_SIZE_FILES	–	–	–

Disk size of archive server in GB.

INT_CLIENT_VERSION	–	–	–

The program version of the client. Is checked for compatibility at the server at login.

INT_OPEN_OBJECTS	R/O	–	–

The number of open objects that were opened or created within this application session, including the application session itself.

TS_SERV_ID	–	–	–

The id of the default server. Used to find linked objects within SET_ATT_OBJ.

TCSI may display an icon showing the status of the remote server session. The following three objects define the window and the relative position of the indicator. When either of these objects does not exist the session status indicator is not displayed.

INT_WINDOW_HNDL	–	–	–
INT_IND_X_POS	–	–	–
INT_IND_Y_POS	–	–	–

ohh_c_create (h_appl_session, type, phand)

Creates the specified object and returns a handle to it. Any type of object can be created (exception: anchored objects). The created objects have no parent.

The application session is the only object that provides an object create function.

errors: none

ohh_o_forget (h_appl_session)

Closes the application session, discards all open objects that may still be left for this session.

Note:
As a good programming practice close all other open objects and check the object count to be one, before closing your application session.

errors: none

Automatic Time Zone Conversion

A client application using TCSI may switch on automatic conversion of time stamps by setting the following registry value:

"**TCSI\AutoTimeZone**" (REG_DWORD) 0 = off / 1 = on, default is 0 (off)

The automatic conversion of time stamps applies to the functions ohh_c_time_get() and ohh_c_time_put().

If automatic conversion of time stamps is selected, the SET_TIME_ZONE object from the server session, if it exists, is automatically saved to the application session SET_APPL_SESSION. All subsequent calls of ohh_c_time_get() and ohh_c_time_put() will then work with the local time zone of the logged-in user.

This feature must not be activated if several server sessions are opened below the same application session, by different users or connecting to different servers or working with different threads. It works only for a single TCOSS server session and an optional archive server session below a particular application session.

When connecting to a TCOSS release based on local time the automatic conversion of time stamps is not possible and the "AutoTimeZone" registry value has no effect.

Note:

The time stamps handled by the functions ohh_b_ascii_get and ohh_b_ascii_put (ASCII backup – restore), ohh_b_bin_get and ohh_b_bin_put (binary backup – restore), ohh_b_sax_getW and ohh_b_sax_putW (XML get and put) are not affected by the new feature. These time stamps are always in TCOSS time.

Code Pages in Application Session

Code page values defined in SET_APPL_SESSION:

INT_CODEPAGE … code page used on interface, 65001 = Unicode as UTF-8

INT_CODEPAGE_LEGACY … legacy code page: 0, 1, 932 etc.

Both values are used for login and for the conversions done inside the new Unicode string functions. The legacy code page INT_CODEPAGE_LEGACY is only required if INT_CODEPAGE is 65001. It is used for computing the password hash in a way which is compatible to old clients, at login and when editing user profiles. It need not be set for handling users which have only ASCII characters in User ID and password.

The INT_CODEPAGE value is also used by the text preview (L_CONTENT_VIEW_TXT) to correctly convert cover variables into the appropriate text code page and to insert cover variables as Unicode into RTF cover sheets.

Compatibility Mode Login

Clients which are not updated, and clients which choose not to use Unicode, continue to work with the legacy TCOSS system code page. This mode is selected by not setting INT_CODEPAGE in SET_APPL_SESSION to 65001 before login, or by not setting it at all.

If TCOSS is running on Unicode the INT_CODEPAGE value in the SET_SERVER_SESSION will be the legacy code page, and TCOSS will convert sent and received data to and from Unicode so that the Unicode TCOSS actually behaves like a TCOSS running on a code page.

An application could log in in compatibility mode, but still use the new Unicode string functions by setting INT_CODEPAGE in SET_APPL_SESSION (see below).

Login of Unicode enabled Client

Clients which can handle Unicode signal this by setting INT_CODEPAGE in SET_APPL_SESSION to 65001 before login. If TCOSS is running on Unicode and the User ID contains non-ASCII characters the client is also required to set

INT_CODEPAGE_LEGACY in SET_APPL_SESSION to the appropriate value before writing TS_USER_ID and TS_PASSWORD.

The new query server properties function should be used to find out TCOSS code page settings before login. Only a TCOSS running on Unicode will allow Unicode on the interface.

It is also possible to try a Unicode login without knowing whether TCOSS is running on Unicode, if User ID and password are all ASCII. In this case the returned INT_CODEPAGE in the server session should be checked, if it's not 65001 TCOSS is not running on Unicode and INT_CODEPAGE in SET_APPL_SESSION sould be updated accordingly. If INT_CODEPAGE in the server session is 65001 also read the INT_CODEPAGE_LEGACY value from the servcer session and set it in SET_APPL_SESSION.

3.6.1.4 SET_SERV_ENTRY

It is an entry in the list of servers defining a server. It holds the server ID, the path to the server, the server session and the clearing cause.

Child objects:	**attributes**	**default**	**value-restr**
TS_SERVER_ID	–	–	–

The ID of the server.

TS_PATH	–	–	–

The path to the server. Is interpreted as path to a local server if the string contains a backslash ('\'), otherwise the path points to a remote server.

Local path format: a path in DOS format: drive letter, colon, backslash, any file path
e.g.: `C:\MYFOLDER`

TCSI also offers the possibility to access a local TCOSS file structure to read and write files by specifying a server path in the format "0:\+TOS", e.g. for the WCONFIG utility. This path format allows access to different TCOSS instances in an ASP environment. The process registry subkey (e.g. "TCOSS01") is added to the path after a backslash separator:

path	TCOSS instance	process started with options
"0:\+TOS"	TCOSS	TCOSS /t
"0:\+TOS\TCOSS01	TCOSS01	TCOSS01 /t:TCOSS01 /k:TCOSS01
"0:\+TOS\TCOSS02	TCOSS02	TCOSS02 /t:TCOSS02 /k:TCOSS02
...	...	...

Remote path format: any string not containing a backslash. The remote path is transparently passed on to the TCOSS Transport module (TCTI). See TCTI module for definition.

Child objects:	**attributes**	**default**	**value-restr**
TS_USER_ID	–	–	–

The ID of the user as defined in the user entry

TS_PASSWORD	–	–	–

The user's password as defined in the user entry. (Always transferred to server in encrypted form.)

TS_PASSWORD_NEW	–	–	–

The new password, allows changing the password together with logon. For this operation to succeed, the user must have general write access right to all user profiles or at least the right to change the own password (flag R_CHANGE_PW_OWN in INT_RIGHTS_USERPROF).

If TS_PASSWORD_NEW is not specified, the password remains unchanged.

The logon with password change is the only possibility to logon when the password has expired.

INT_ID_SOURCE	–	–	LAN_ID / TYPED_IN

The ID of the user as defined in the user entry.

TS_WORKST_DESCR	–	–	–

If the workstation description is set by the application before logging in, its value will be used as login parameter instead of the workstation description provided by default (computer name, as shown in SET_OBJ_HANDLER).This feature is used by TC/Link-WM to fill the workstation description with a

string identifying the connected GSM box.

SET_SERVER_SESSION A – –

The server session.

TS_CLEARING_CAUSE R/O – –

The clearing cause as provided by the transport layer.
If a server session is refused locally by TCSI because of non-acceptable TCOSS version the clearing cause is filled as:
"server_TCOSS_version;acceptable_versions"
Version formats as in TCCLIENT.INF.
Note: when the session is cleared by TCOSS the clearing cause is not set

INT_SESSION_HNDL – – –

Here the application may store the session handle, so that other windows may access the already opened session.

ohh_c_open (h_set_server_entry, SET_SERVER_SESSION, phandle)

Opens a server session and returns a handle to it.
When the server can not be reached TS_CLEARING_CAUSE holds information from the transport layer about the problem, otherwise TS_CLEARING_CAUSE is deleted.

errors:	ERR_NOCONN	the server could not be reached
	ERR_USER_ID	invalid user ID
	ERR_PASSWORD	wrong password

Query Server Properties

This function allows to query server properties, in particular its code page setting, without user credentials and without creating a session. It's a new function requiring a new TCOSS release, if used with an old TCOSS release it will return error 613 (ERR_SESSION_LOST).

The server properties are represented by a new child object with name and type L_SERVER_ATTRIBUTES in SET_SERV_ENTRY. They can be opened after TS_PATH has been set appropriately.

Server properties layout:

L_SERVER_ATTRIBUTES

SET_ATTRIBUTE

UN_NAME / CL_TEXTSTRING …"INT_CODEPAGE"

UN_VALUE / CL_INTEGER … code page value

SET_ATTRIBUTE

UN_NAME / CL_TEXTSTRING …"INT_CODEPAGE_LEGACY"

UN_VALUE / CL_INTEGER … legacy code page value

SET_ATTRIBUTE

UN_NAME / CL_TEXTSTRING …"INT_MPL"

UN_VALUE / CL_INTEGER … minimum password length

SET_ATTRIBUTE

UN_NAME / CL_TEXTSTRING …"INT_LUNO"

UN_VALUE / CL_INTEGER … password complexity requirements

The server properties may be extended with additional attributes in future releases.

The “INT_CODEPAGE” value gives the codepage which is actually used by TCOSS. The value is 65001 if TCOSS is running on Unicode, or 0, 1, 932 etc. if TCOSS is running on a code page.

The “INT_CODEPAGE_LEGACY” value gives the legacy code page supported by a TCOSS instance running on Unicode. If TCOSS is not running on Unicode the “INT_CODEPAGE_LEGACY” value is identical to the “INT_CODEPAGE” value.

The values “INT_MPL” and “INT_LUNO” are available since TCOSS 7.96.03. They are used since TCSI 2.80.00 in order to verify if the new password fulfills the complexity requirements according to System Account Policy Directory in TCOSS file +MAIL5V/App99. They should be ignored as opaque values by clients.

Code Page Handling in Local Folders

Local folders are accessed in TCSI by logging in to a server with a file path in the TS_PATH variable instead of a network path. They provide a local message store. Local folders are only used by TCfW, mostly for support purposes.

The local folder implementation is not code page aware. Accessing local folders which have been created in some code page with a Unicode client is not supported. But it will be possible to create a new local folder with a Unicode client and have full Unicode support in the folder.

The SET_SERV_ENTRY object, which is used for log in to the local folder, will be extended with an INT_CODEPAGE_LOCAL field. The client should set it to the configured code page before login, in case of Unicode it should be set to 65001 (CP_UTF8). The SET_SERVER_SESSION object, which is returned on successful log in, will hold an INT_CODEPAGE field, giving the local folder’s code page. The local folder’s code page is determined at creation and can’t be changed afterwards.

When logging in to a local folder the client will have to check the code page returned in SET_SERVER_SESSION and handle 3 cases:

1) No code page is returned: Then it’s a local folder created by an old, non-Unicode enabled version of the client. The local folder’s code page is unknown. The client should issue a warning before displaying its contents.
2) The returned code page does not match the code page specified on log in: Then the local folder has been created in a different code page and the client should issue a warning before displaying its contents.
3) The returned code page is identical to the code page specified by the client: The local folder’s code page is correct and it can be used without restrictions.

The following restrictions apply to a local folder with a non-matching code page:

- Fields in the content listing may show wrong characters.
- Opening a message from the content listing may fail.
- Messages, when opened, may show incorrect characters in string fields, e.g. in the subject. The text blocks will be displayed correctly though.

3.6.2 SET_SERVER_SESSION

Represents the active session between the client and the server.

It holds the mail system, the message store, the user store, the recipient store and the services store and the server state.

Session related errors:

These errors may happen with this object and all objects that have been opened below from it in the hierarchy (the permanent stores, the folders, the permanent store contents and the entries of permanent stores).
When an entry is opened from a permanent store it (with all its sub objects) is bound to the session (the errors below may happen).

1. ERR_NO_SERV_CONNECTION — the link is lost and cannot be reestablished
2. ERR_SERVER_BUSY — the server was busy for a number of retries (should never happen)
3. ERR_ACT_SESSION_LIMIT — the limit to the number of active sessions is reached ("try later")
4. ERR_SESSION_LOST — the session was passive for more than 24 hours

On 1, 2, 3:

User message "Abort / Retry / Cancel".
When the error stays permanent (in spite of retries) same behavior as 4.

On 4:

The objects anchored to the server session do not work anymore. Entries opened under this session may not be fully accessible anymore. You can still work with the accessible part of those entries (e.g. store them to your local server).

Child objects:	**attributes**	**default**	**value-restr.**
SET_PS_MAIL_SYSTEM	A	–	–
SET_PS_MAIL_ARCHIVE	A	–	–
SET_PS_MESSAGE_STORE	A	–	–
SET_PS_USER_STORE	A	–	–
SET_PS_RECIP_STORE	A	–	–
SET_PS_SERVICES_STORE	A	–	–

The permanent stores. Some of the stores may not exist, depending on the server.

SET_SYSTEM	A	–	–

The server’s system.

INT_CODEPAGE	R/O	–	–

The code page used on interface, 65001 = Unicode as UTF-8

INT_CODEPAGE_LEGACY	R/O	–	–

The legacy code page: 0, 1, 932 etc.

INT_N_OF_UNREAD	R/O	–	–

The number of unread messages in the user's in-box (at the time the session was opened)

INT_QUEUELEN	R/O	–	–

The total number of messages in the user's in-box (at the time the session was opened)

INT_DAYS_PW_VALID	R/O	–	–

	The number of days the password is still valid (including the current day). The value PW_NEVER_EXPIRES indicates that the password never expires.		
TS_COMPANY	R/O	–	–
	The server's customer ID.		
SET_TIME_ZONE	R/O	–	–
	The user's local time zone.		
SET_XML_SERVER	–	–	—
	Provides direct streaming access to the connected TCOSS or TC/Archive server's OHL (object handler low) interface.		
TS_ROLE	–	–	—
	Provides the role switching functionality. Set TS_ROLE to switch to a particular role. Delete TS_ROLE or set it to an empty string to clear the current role to continue working with the logged-in user.		

3.6.3 SET_PS Permanent Store

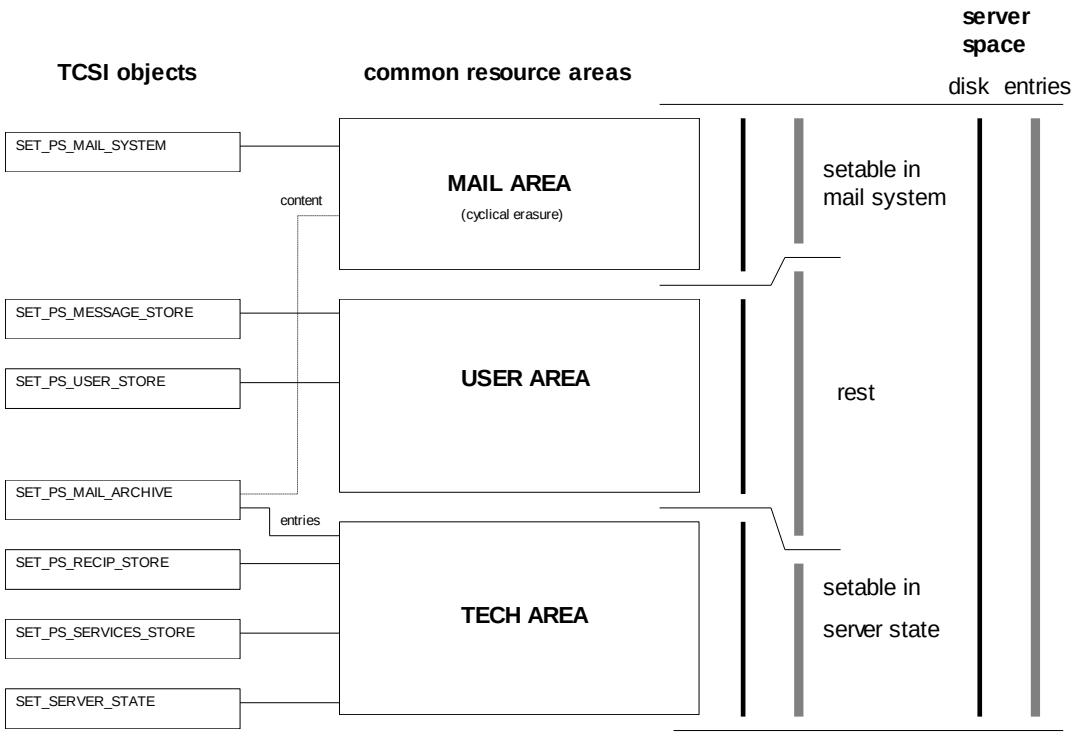
A permanent store allows storing entries permanently. An entry is the stored object itself, not just a reference.

The sub object permanent store content (SET_PS_CONT) holds the entries. All permanent operations on entries are done on this object.

To select existing entries a store holds a folder. This folder represents a view on the contents of the store. The folder holds a filter (it defines a view on the store) and a (filtered) list of display entries. A display entry does not hold all the sub objects of the stored entry (see SET_PS_FOLDER), it can be used to select the stored entry.

Child objects	**attributes**
SET_FOLDER	When you open this object more than once you can have a number of different views on the store at the same time
SET_PS_CONT..PERM	Holds the entries

Resource assignment for permanent stores on TCOSS servers:



Note on TOS Folders:

MAIL AREA:	"+MAIL"	client mail
	"+MAIL5V"	TCOS 5 files (nn99, KKnn, ...
	"+MAIL5I"	invisible help files (as used with TTX conversion facility)
	"+MAILSYS"	mail entry file (COMINFOFILE)
		recipient directory (holds system and user's directories)
		archive entry file
		registration store file
		service store file
USER AREA:	"+USER"	the user-definition files of the user store object, one file per user

	"username"	for each user on the server a TOS folder with "userid" exists, it holds all the files of the user's message store
TECH AREA:	"+TECH"	only one TOS folder in tech area; contains: program files config files

3.6.3.2 SET_PS_MAIL

The mail stores hold entries of type SET_ENTRY_MS_MAIL . The UN_CONTENT of those entries must be a L_ENV_CONT (or it may not exist). These entries are called 'mail entries' or simply 'envelopes' below.

To read the status of mail entries they can be found in the IN folders of the recipients and the OUT folder of the originator.

3.6.3.3 SET_PS_MAIL_SYSTEM

The mail system allows posting envelopes. When an envelope is posted the system creates mail entries according to the posted envelopes contents and the in-actions for internal receivers (see diagram).
These (pending = not delivered yet) entries stay within the mail system until termination (delivery or retries expired). On termination the entries are automatically either deleted or transferred to the mail archive. They can be canceled (but not deleted) via TCSI. Canceling of an entry will terminate it and transfer it to the archive.

For details of allowed status changes see SET_PS_CONT_MAIL.

Scheduling:

The in mail folder gives the mail entries filtered for a special recipient. The entries are automatically scheduled according to priority and delivery time.

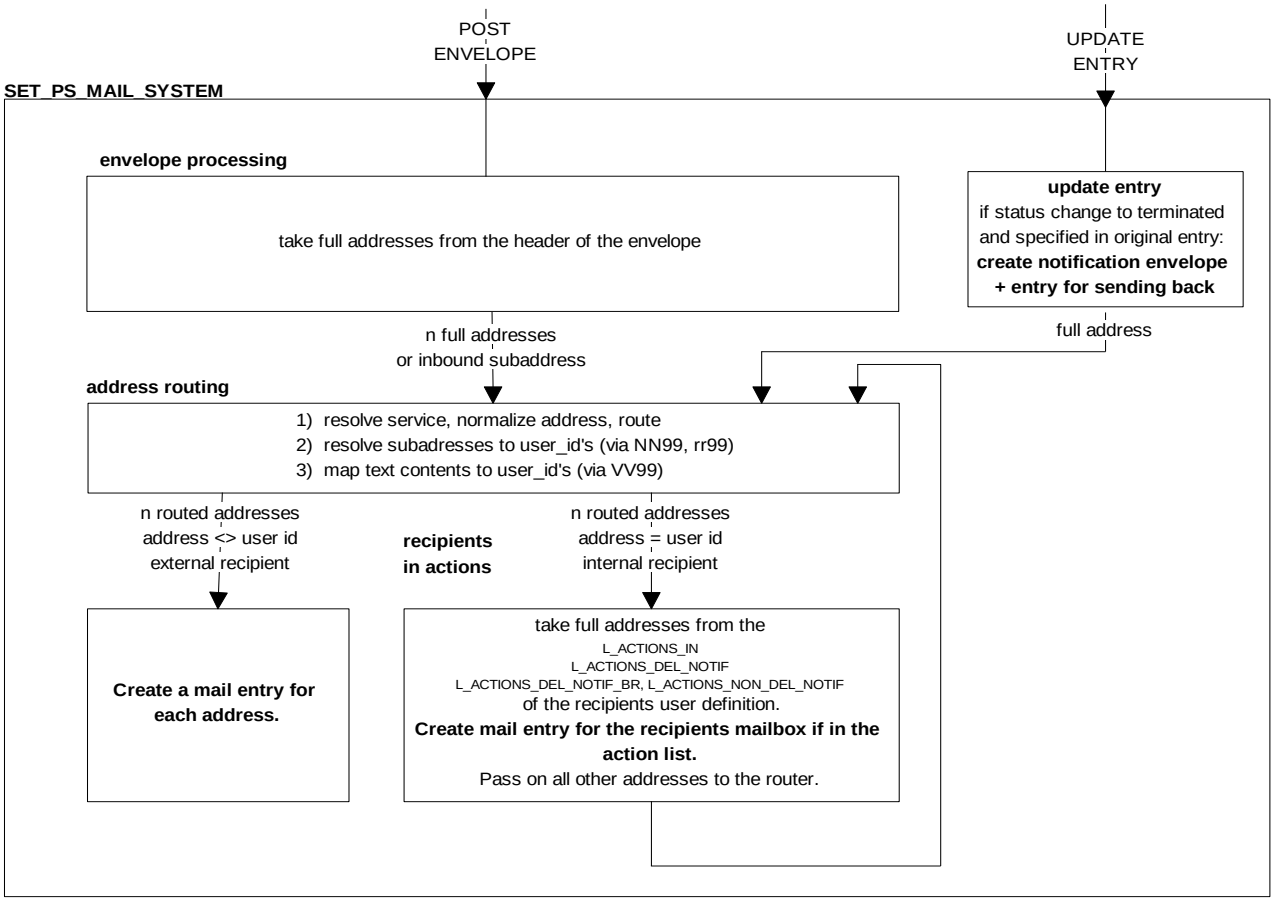
To access the contents of an envelope (UN_CONTENT) the mail entry has to be opened not from the folder but from the sub object SET_PS_CONT_MAIL.

For updating (state change, etc.) the mail entries can be changed and closed again. The content is always read only for mail entries.

When the client changes the state from pending to terminated, the mail system may create a notification. A notification consists of a mail entry addressed to the originator of the message. It may hold an optional L_ENV_CONT at UN_CONTENT with a verbal notification and backreception.

Message waiting:

The mail system keeps track of the empty / not empty state of the user's in mail. When the state changes an entry as defined in the user profile is created. These entries may be queried by a client to control message waiting indicators (e.g. at PBX's).



The in mail actions defined for a user create mail entries. The user is the originator for all those entries.
Therefore the originator of the message only finds the mail entry to the recipient in his out folder, not the in mail actions of the recipient.

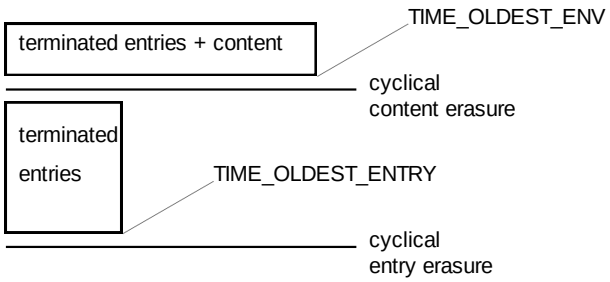
Child objects: attributes default value-restr

see: class SET_PS

3.6.3.4 SET_PS_MAIL_ARCHIVE

The short term archive contains terminated mail entries. Entries in the short term archive are automatically deleted by two cyclical erasure mechanisms:

The first deletes the content parts of terminated entries (UN_CONTENT), the second deletes terminated entries. So, for old entries the content part may not exist anymore.



Child objects:	attributes	default	value-restr

see: class SET_PS

3.6.3.5 SET_PS_MESSAGE_STORE

The message store allows storing all kinds of files. The message store holds entries of type SET_ENTRY_MS. The files will stay here until manual erasure.

Mail entries can also be put into a message store. In that case all the additional fields of the mail entry are lost.

Child objects:	attributes	default	value-restr

see: class SET_PS

3.6.3.6 SET_PS_USER_STORE

The user store holds the entries of the server's users.

Child objects:	attributes	default	value-restr

see: class SET_PS

3.6.3.7 SET_PS_RECIPIENT_STORE

The recipient store holds the entries of the recipients known at the server. For each user in the server's user store also an entry in the recipient store exists.

Child objects:	attributes	default	value-restr

see: class SET_PS

3.6.3.8 SET_PS_SERVICES_STORE

The services store holds the definition of the services.

Child objects:	attributes	default	value-restr
see: class SET_PS			

3.6.3.9 SET_PS_REGISTRATION_STORE

The registry store holds the registration information of all the registered client installations. A client application is registered automatically when it logs on to the TCOSS server for the first time after installation on the workstation. Deinstallation is possible via TCSI.

Child objects:	attributes	default	value-restr
see: class SET_PS			
plus:			
SET_STATE	R/O	-	-
The state of the registration store.			

3.6.3.10 SET_PS_ARCHIVE

The archive store exists only on a TC/Archive Server. It holds all permanent objects of the long term archive.

Child objects:	attributes	default	value-restr
see: class SET_PS			
plus:			
SET_STATE_ARCHIVE	R/O	-	-
The state of the archive			
SET_FOLDER_ARCVOL	-	-	-
The archive volume folder for the administration of archive volumes			

3.6.4 SET_PS_CONT Permanent Store Content

The permanent store content holds the entries of a permanent store. The type of the entries depends on the type of the permanent store.

The selection of the child objects is done by specifying a selector object, not a predefined integer-name as with a normal set. The type of the selector object depends on the type of the store. The entry of a store itself is accepted as selector object by all types of stores.

When a permanent store content is closed the entries that were opened from it do not lose their parent. When such an entry is closed the permanent store content at the server is updated. The entry only loses its parent when the server session is closed or lost.

General errors:

ERR_INVALID_SELECTOR	(replaces ERR_WRONG_NAME) The selector object specified is invalid. (This is different from the condition that the selected object does not exist in the store)

ohh_c_any function ( )

For all set functions the name parameter is replaced by the handle to the selector object.

errors: none

Creating a new entry:

To create a new permanent entry you first create a temporary one and fill in the entry id. Then you do a save_at() or close_at() to this object, giving the handle to the entry as selector and as object to be stored:

***ohh_close_at / ohh_save_at* (h_set_ps_cont, h_entry, h_entry, bool_overwrite)**

The first h_entry is the selector object; the second is the object to be stored.

Similar to the standard SET the overwrite parameter controls whether an existing entry with the same name is overwritten or not.

errors:

ERR_INVALID_SELECTOR	The selector object is either of wrong type or has invalid contents.
ERR_STORE_FULL	Capacity of store is exceeded.

Note: To change the name of an existing entry you have to create a new one (with the new name in the selector) and delete the old one.

Changing of entries:

To access the contents of an entry (UN_CONTENT) the entry has to be opened not from the folder but from this object.

Changing of entries is done with an open – change – close/save sequence. This will give the ERR_CHANGED if the permanent object has been changed since opening.
When you change the selector part of your entry this is ignored.

***ohh_close / ohh_save* (h_set_ps_cont_mail)**

errors:

ERR_INVALID_ENTRY	the entry is corrupted
ERR_STORE_FULL	Capacity of store is exceeded.

3.6.4.2 SET_PS_CONT_MAIL

For the mail system, the behavior of the content is modified a little – you cannot do a save_at() to create a new entry, and you cannot overwrite an entry with save_at(). See below.

selector object type:

SET_ENTRY_MS_MAIL selector object must be obtained from folder

ohh_close / ohh_save / ohh_close_at / ohh_save_at (...)

Within PS_CONT_MAIL overwriting is not possible. The overwrite flag is ignored.

A function of the mail system allows to check and verify the routing of a particular address to a valid recipient in the TCOSS server, similar to the ..CHECK command on the dot-dot interface. The routing check is done without applying location-based routing rules.

The new function is implemented as an open function in the mail system:

ret = ohh_c_open(h_set_ps_cont_mail_sys, h_full_address, ph_set_entry_ms_mail, ptype);

The first parameter of the call is a handle to the mail system's permanent store content (SET_PS_CONT_MAIL_SYS).

The second parameter is a handle to a SET_FULL_ADDRESS as selector entry, specifying the address to be checked.

The function returns a handle to a mail entry (in the third parameter) or an error code if routing of the specified address fails. In case of success the fields TS_RECIPIENT, TS_LOCALIZED_ADDR and TS_NORMALIZED_ADDR of the returned mail entry contain useful information. Other child objects of the mail entry may not exist or contain default values.

When testing fax or telex inbound routing with the new function, the separation character between number and answerback ('-' for fax, '/' for telex) will cause problems. Inbound routing requires an exact match between input number string and stored prefix, but the input number string generated from an address of type fax or telex always has the separation character at the end, given that the input answerback is empty.

As workaround for this problem the separation character can be removed by appropriate entries in the routing directory "Arr99", e.g. in the "**NORMALIZE" section:

FXI:~-,FXI:~, removes '-' at end of number for fax inbound routing test via TCSI
TXI:~/,TXI:~, removes '/' at end of number for telex inbound routing test via TCSI

Posting an Envelope

To post an envelope you close_at() an object of type SET_ENTRY_MS or SET_ENTRY_MS_MAIL to this set. As the handle to the selector object you specify the AFTER_END handle. This operation can be seen as putting the envelope into the 'mail slot'.

To be accepted by the mail system the posted entry must fulfill the following conditions:

1) UN_CONTENT must be of type L_ENV_CONT
2) the first element of L_ENV_CONT must be a valid header (type SET_HEADER) with a valid originator and at least one valid recipient

The field TS_FILE_NAME of the posted entry is copied to TS_ENV_NAME_POSTED of all created entries.

Default resolution:

When an envelope is posted the priority, date, time, cost center, options and termination of the general section of the header are taken as default values for the respective values of the following recipients. Similarly, INT_SENDING_COPY defines the default value of the BACKRECEPTION flag of INT_OPTIONS.

Mail entries created by TCSI can not be seen from the TCOS 5 contents. This is not a problem, since the query command does work for new entries.

***ohh_close_at* (h_set_ps_cont_mail, AFTER_END, h_set_entry_ms.., ...)**

Note: 'overwrite' need not be specified since the mail slot is always empty.

errors:

ERR_INVALID_RECVR	one or more receiver(s) in header could not be accepted
ERR_INVALID_ORIGINATOR	originator missing or corrupted

Normally, when an envelope with several recipients is posted to TCOSS, it will only be accepted if all recipients are valid. A single invalid recipient causes the whole transaction to fail.

The flag HANDLE_INVALID_REC in the INT_OPTIONS child of the header and recipient entries changes the standard TCOSS reaction to invalid recipients. If it is set, recipients which turn out to be invalid are replaced with the fixed recipient string “+INVALID:” and a retry to route and generate a send order is made. If the user “+INVALID” exists or “+INVALID:” is routed to another existing user, the generation of a send order will succeed.

If HANDLE_INVALID_REC is set in the INT_OPTIONS child of the envelope header it applies to all recipients. If it is not set in the header it may be set individually for each recipient in the recipient entry.

Please note that this feature does not change the actual recipient specified in the envelope header. Only the recipient in the mail entry is set, it can be seen as kind of automatic re-routing.

This option is intended to be combined with the “AUTO_REJECT” feature to produce non-delivery notifications for invalid recipients.

Posting an Envelope for Correction

To correct a mail entry you post the envelope again with the corrected header entry. As selector object you specify the mail entry to be corrected (from the display folder or PS_CONT_MAIL).

The entry being corrected is negatively terminated (INT_STATE = ST_CORRECTED or INT_STATUS = NEG_TERM), with the following use of termination flags:
(ARC_POS or ARC_NEG), DEL_ENTRY_NEG, no backreception event, no notifications generated

The operation fails if the entry to be corrected can not be found.

Except correction of the old entry the operation behaves exactly like normal posting. The application has to reset the active flags for all recipients except the one being corrected.

Posting-for-correction with state ST_DISTRIBUTED

For message distribution the INT_STATE of the selector entry can be set to ST_DISTRIBUTED (or INT_STATUS set to DO_POS_TERM_NO_NOTIF). In this case the original entry will be positively terminated and no notifications are generated.

Posting-for-correction with state ST_REJECTED

For the message authorization feature the INT_STATE of the selector entry can be set to ST_REJECTED. In this case the original entry will be negatively terminated and **non-delivery notifications are generated** if requested.

***ohh_close_at* (h_set_ps_cont_mail, h_set_entry_ms_mail, h_set_entry_ms, ...)**

Note: 'overwrite' need not be specified since the mail slot is always empty.

errors:

ERR_INVALID_RECVR	at least one receiver in header could not be accepted
ERR_NOT_FOUND	entry to be corrected could not be found
ERR_INVALID_ORIGINATOR	originator missing or corrupted

Posting While Receiving (Streaming to TCOSS)

Requirements

- The received voice data should be transferred directly to TCOSS via TCSI, with a minimum of overhead to get high performance.
- The received message should be stored permanently on the TCOSS server. If the connection to the TCOSS server is lost during reception or the TCOSS server is shut down, at least part of the voice mail should be delivered to the intended recipients.
- In case of interrupted reception, the message should be marked with error code “SP”.

Implementation idea

TC/Voicemail should generate a complete envelope, including a header with all recipients and a binary attachment for the voice data, right at the beginning of the record function. The binary attachment may be empty or already contain the first seconds of the voice message. A TIME_REC_END value which gives the current reception end time relative to the reception start time i.e. the current duration may be written to the message store entry holding the envelope.

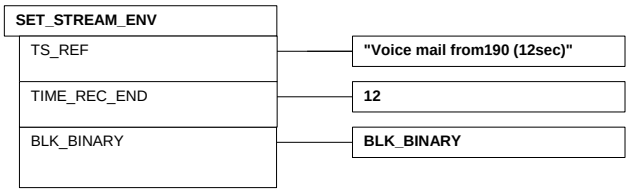
This envelope is saved to the TCOSS server. After the first save operation, voice data may be added or modified with special functions. At defined intervals, check points are inserted to ensure permanent storage of the data. At the end of the recording, the envelope is closed and its send orders are activated in TCOSS. Alternatively, it would also be possible to discard the complete message.

Implementation details

An envelope with a header and an attached object holding a binary block is created normally in TCSI. The envelope content list (L_ENV_CONT) may also contain other objects in between, as long as the header is the first element and the attachment with the binary block is the last element of the list. This envelope is saved to the mail system permanent store with

ohh_c_save_at(h_ps_cont, SET_STREAM_ENV, h_entry_ms, TRUE)

The envelope is transferred to TCOSS, all send orders are created, but the file is not closed. Also the connection to the client-server channel on TCOSS stays open. TCOSS sends back a new object “SET_STREAM_ENV”, which contains the TS_REF field from the envelope header.



In TCSI, if the ohh_o_save() succeeds, one has a handle to the new “SET_STREAM_ENV” object, which allows the following functions:

ohh_c_stream_put write data to the binary attachment, with position, length

	overwrite existing data or append new data to the end of the block
ohh_c_ts_put	change the TS_REF content, the length of the field can't be changed
ohh_c_time_put	change the reception duration in TIME_REC_END
ohh_o_save	check point, save binary attachment, TS_REF and duration, file stays open
ohh_o_close	save binary attachment, TS_REF and duration, close file, activate send orders
ohh_o_forget	cancel reception, discard file and send orders, no traces left in TCOSS

Restrictions:

- No other TCSI calls to objects related to the same server session are allowed between the initial ohh_c_save_at() of the envelope and the final ohh_o_close() or ohh_o_forget(). Exempt are calls which are processed locally on the client without server access.
- The intervals between calls of ohh_c_stream_put() or ohh_o_save() of the SET_STREAM_ENV have to be shorter than 120 seconds (client-server channel inactivity timeout).
- If a linked content (UN_CONTENT_LNK) of an attached object without parent relation to a permanent store (e.g. a newly created SET_ATT_OBJ) is opened in the same process while some other thread is streaming to TCOSS, this open may fail with error 613 (ERR_SESSION_LOST). The reason is that a session accessing the same server is "borrowed" from a global table, and this session may be active with streaming to TCOSS, so it cannot be activated on a different client-server channel.

The function ohh_c_stream_put () provides optimized stream write access.

```
TCSI_RET ohh_c_stream_put( HOBJ h_container,   // Handle to SET_STREAM_ENV
                           TCSI_ID id,         // BLK_BINARY
                           LPCHAR pbuffer,
                           INT4 bufsize,
                           INT4 position)
```

The function writes *bufsize* bytes from the buffer pointed to by *pbuffer* into the envelope's last object (which has to be a BLK_BINARY) at offset *position*. If the specified *position* is AFTER_END or if it is beyond the current end of the block, the data will be appended at the end of the block, no error or warning is returned for this case.

With every call to ohh_c_stream_put() the data is transferred immediately to TCOSS. Data is not buffered in TCSI to have fewer calls with larger data blocks to TCOSS. It is important that there is regular activity on the connection: the TCOSS client-server channel has an inactivity timeout of 120 seconds.

In case of timeout, connection interrupted or unexpected commands to TCOSS during reception, the voice message will be closed with the binary block truncated at the last checkpoint size and delivered to its recipients. It is critical that the connection is not lost during reception, as it is not possible to make retries by connecting again.

If the message subject is written with ohh_c_ts_put to the TS_REF child object, it is stored locally in TCSI. Only with the ohh_o_save() or ohh_o_close() function, the subject is updated in the envelope header as well as in the message store entry, where a truncated copy (the first 31 characters) of the subject is kept by TCOSS.

The length of the TS_REF field in the envelope header can't be changed after the initial ohh_c_save_at() of the envelope. There will be no error return code if a subject of different length is specified in this state, the subject will be truncated or padded with spaces to retain its original length.

The ohh_o_forget() function of the streaming envelope may not be able to do a complete cleanup of the message on the TCOSS server, especially if the server connection was lost during streaming. The return value of the ohh_o_forget() function will be "OK" if all TCSI internal temporary objects could be removed, independent of whether the cleanup on TCOSS succeeded or not.

If any error is reported during streaming, i.e. if ohh_c_stream_put(), ohh_o_save() or ohh_o_close() returns an error, it is important that ohh_o_forget() is called on the handle to the SET_STREAM_ENV. Otherwise the TCTI attachment used for communication with the TCOSS server is not freed.

The final ohh_o_close() or ohh_o_forget() function also closes the connection to the client-server channel on TCOSS.

Changing of Entries

To access the contents of an envelope (UN_CONTENT) the mail entry has to be opened not from the folder but from this object.

For updating (state change, etc.) the mail entries can be changed and closed again. The content is always read-only for mail entries. Changing of entries is only possible with an open – change – close/save sequence, save_at() is not supported here.

Update restriction per field has to be implemented in the future when third-party gateways use the interface.
E.g.: The originator is not allowed to positively terminate an entry.

***ohh_close* / *ohh_save* (h_set_ps_cont_mail)**

errors:

ERR_INVALID_ENTRY	the entry is corrupted
ERR_INVALID_STATUS	the requested status change was denied (see below)

When you do the changing with save(), the values in the entry are updated during the save operation.

Note: Changing of the status may delete the entry from the mail system. Even in this case the object stays open after save() - but it does not exist in the mail system anymore.

There are 2 status variables in the mail system:
1) INT_STATE: to be used for new developments, TCOSS 6.05.00 or higher required
   INT_STATE_MASKED: INT_STATE without flags for filtering in mail folders
2) INT_STATUS: supported for compatibility reasons, available with all TCOSS releases

INT_STATE_MASKED overview:

INT_STATE without flags	value	description
ST_POSTED	150	new entry put into mail system
ST_WAIT_CONV	160	waiting for document conversion
ST_CONVERTING	170	document conversion in progress
ST_CONV_END	180	document conversion complete
ST_WAIT_SEND	190	put into in-box of gateway user or public line channel
ST_WAIT_RETRY	200	wait after unsuccessful send attempt
ST_SENDING	300	sending (on public line channel or to gateway user)
ST_BREAK0	310	sending ended with BREAK=0 (checkpoint)
ST_BREAK1	320	sending ended with BREAK=1 (checkpoint)
ST_BREAK2	330	sending ended with BREAK=2 (checkpoint)
ST_BREAK3	340	sending ended with BREAK=3 (checkpoint)
ST_BREAK4	350	sending ended with BREAK=4 (checkpoint)
ST_BREAK5	360	sending ended with BREAK=5 (checkpoint)
ST_CANCELLED	400	cancelled by originator
ST_CONV_FAILED	405	document conversion failed
ST_TIMEOUT	410	latest delivery time limit expired
ST_END_OF_RETRIES	420	no more send attempts
ST_REJECTED	430	rejected by authorizer
ST_CORRECTED	450	corrected with posting-for-correction
ST_SENT	500	sent to next node, not yet delivered
ST_DISTRIBUTED	600	distributed with posting-for-correction
ST_ROUTED	610	transferred to next TCOSS node, terminated
ST_DELIVERED = ST_RECEIVED	650	delivered to recipient's mail box, archiving of in-box entries
ST_UNREAD	700	new entry in in-box
ST_AUTO_FORWARDED 1)	710	automatically forwarded (unread)
ST_READ	800	read by recipient
ST_REPLIED 1)	810	replied by recipient
ST_PROGRESSED 1)	820	progressed by recipient

1) state value for future extension, not used in current implementation

State values ST_DELIVERED or higher may be combined with flag ST_TERMINATE. ST_TERMINATE indicates that this state update is the final one.

State ST_SENDING is set by the mail system with the open-for-sending and open-for-transfer functions, sending on public line channels and with the ..QUERY command.

State ST_UNREAD is set by the mail system after a state update to ST_DELIVERED without flag ST_TERMINATE.

The state value ST_ROUTED replaces BREAK=7. Setting state ST_ROUTED results in state ST_SENT (for normal send orders) or ST_ROUTED (for routing send orders or notifications).

The state flag ST_NUMBER_LOCKED indicates that the recipient number has been locked by an open-for-sending or open-for-transfer function.

Handling of remote users by gateway client:

The gateway client sets state ST_SENT after the message transfer if further state updates or notifications can be expected. This removes the mail entry from the gateway send queue. The last update reflecting the state of the message on a remote node is marked with flag ST_TERMINATE.
In case the message has to be transferred again after state ST_SENT has been set, the gateway may update the state to a value lower than ST_SENT (e.g. ST_BREAK2). This will put the mail entry back into the gateway send queue.

INT_STATE flag ST_WAIT_READ

The flag ST_WAIT_READ may be used by the gateway client to store the information whether a particular mail entry is waiting for a delivery or a read (receipt) notification.

The flag ST_WAIT_READ may be set in INT_STATE together with state ST_SENT. Setting of this flag means that the message is waiting for a read notification. If the next notification for that message contains INT_STATE = ST_DELIVERED with flag ST_TERMINATE set, the flag ST_TERMINATE will be ignored (the send order will wait for the final notification with INT_STATE = ST_READ | ST_TERMINATE).

INT_STATE flag ST_POSSIBLE_RETRY

This flag is set when any send attempt caused a retry. It remains set even if any further attempt succeeds. It should be used by clients to show state values lower that ST_WAIT_RETRY (that may happen after a new alternative number is selected) as retry instead of new.

Handling of normal (not TC_REGISTERED, no INSTANT) send order:

INT_STATE_MASKED	altern. number	delivery notification 1)	non-delivery notification 1)	send archive 1)	reception archive 2)
ST_CANCELLED			NOTIF_NEG	ARC_POS | ARC_NEG	
ST_TIMEOUT			NOTIF_NEG	ARC_NEG	
ST_END_OF_RETRIES	yes		NOTIF_NEG	ARC_NEG	
ST_REJECTED	yes		NOTIF_NEG	ARC_NEG	
ST_CORRECTED				ARC_POS | ARC_NEG	
ST_SENT					
ST_DISTRIBUTED				ARC_POS	
ST_ROUTED				ARC_POS	
ST_DELIVERED = ST_RECEIVED		NOTIF_POS		ARC_POS	NO_IN_ARC (if not set)

1) controlled by flags in INT_TERMINATION child of mail entry

2) controlled by flag in INT_ACTIONS child of recipient user entry

Relation between INT_STATE_MASKED and INT_STATUS:

INT_STATE_MASKED	value	INT_STATUS read	INT_STATUS write
ST_POSTED	150	NEW_ENTRY or TOUCHED [1)]	DO_ACTIVATE
ST_WAIT_CONV	160	NEW_ENTRY	
ST_CONVERTING	170	NEW_ENTRY	
ST_CONV_END	180	NEW_ENTRY or TOUCHED [1)]	
ST_WAIT_SEND	190	NEW_ENTRY or TOUCHED [1)]	
ST_WAIT_RETRY	200	TOUCHED	
ST_SENDING	300	SENDING_ACTIVE	
ST_BREAK0	310	SENDING_ACTIVE	
ST_BREAK1	320	SENDING_ACTIVE	
ST_BREAK2	330	SENDING_ACTIVE	
ST_BREAK3	340	SENDING_ACTIVE	
ST_BREAK4	350	SENDING_ACTIVE	
ST_BREAK5	360	SENDING_ACTIVE	
ST_CANCELLED	400	NEG_TERM	DO_CANCEL
ST_CONV_FAILED	405	NEG_TERM	
ST_TIMEOUT	410	NEG_TERM	
ST_END_OF_RETRIES	420	NEG_TERM	
ST_REJECTED	430	NEG_TERM	
ST_CORRECTED	450	NEG_TERM	(posting for correction)
ST_SENT	500	AT_NEXT_NODE	
ST_DISTRIBUTED	600	POS_TERM	DO_POS_TERM_NO_NOTIF
ST_ROUTED	610	POS_TERM	
ST_DELIVERED = ST_RECEIVED	650	POS_TERM	
ST_UNREAD	700	NEW_ENTRY	
ST_AUTO_FORWARDED 2)	710	NEW_ENTRY	
ST_READ	800	TOUCHED	
ST_REPLIED 2)	810	TOUCHED	
ST_PROGRESSED 2)	820	TOUCHED	

1) Value TOUCHED is returned if the state is reached as first attempt with the next alternative number.

2) state value for future extension, not used in current implementation

Status changes allowed at entry update:

new status: / old status:	NEW	TOUCHED	AT_NEXT_NODE	POS_TERM	NEG_TERM	DO_CANCEL	DO_ACTIVATE
NEW	-	Yes, set by user client in In-box and MDA	Yes, used by MDA and Gateway that expects notification	Yes, by recipient's client	Yes, by recipient's client. Mail activates alternative numbers	Yes, by originator's client. Mail does not activate alternative numbers, next status will be NEG_TERM Entry moved to archive.	Yes, by originator's client. Mail will activate first alternative, RETRIES=9, STATUS= TOUCHED
TOUCHED	No	-	Yes, as above	Yes, as above	Yes, as above	Yes, as above	Yes, as above
AT_NEXT_NODE	No	No	-	Yes	Yes, as above	Yes, as above	Yes, as above
POS_TERM	No	No	No	No	No	No	No
NEG_TERM	No	No	No	No	No	No	No

Notes:

- DO_CANCEL, DO_ACTIVATE and DO_POS_TERM_NO_NOTIF are statuses that are only set by the client. They will never be read back from the mail system.

Opening Entries for Sending (Query)

When more than one client fetches messages from the same in-queue (to send or forward the messages) the following must be guaranteed:

1. the entries have to be presented in the order of priority
2. each entry is just handled by a client at a time

This is done by the open-for-sending operation.

To do an open-for-sending you:

- create a SET_FILTER_MS_MAIL
- set TS_RECIPIENT to the queue you want to query (no wildcards)
- use SET_FILTER_MS_MAIL as selector object

Open will give you the first entry in the selected queue. The state of the returned mail entry is set to ST_SENDING (status SENDING_ACTIVE). The entry is not locked, but removed from the in-queue and put into a separate queue with timeout supervision (1 hour timeout). If the timeout expires before any update (with a close, save or posting-for-correction function) is done, the entry is put back into the in-queue with state ST_WAIT_RETRY and the INT_SUSP_DUPL is set to YES.

ohh_c_open **(h_set_ps_cont_mail, h_set_filter_ms_mail, ...)**

errors: none

Optimization:

To avoid the problem that the voice link runs out of internal TCSI handles if messages with long recipient lists are processed, the following optimizations have been implemented.

The idea of the optimization is that the recipient list in the message's header is not built up in memory because it is skipped when received from the server.

The new child object INT_SKIP_TYPE has been defined in SET_FILTER_MS_MAIL. The voice link will set the INT_SKIP_TYPE value to the type constant L_RECIPIENTS before calling the open-for-sending function.

An open-for-sending function with INT_SKIP_TYPE set in the selector object SET_FILTER_MS_MAIL returns, if successful, a handle to a message which has been stripped of all child objects of the type specified in INT_SKIP_TYPE. The stream optimization parameters Xlevel, Xblock, Xtotal and Xmode are not active in this case. The INT_SKIP_TYPE value is also passed on to the server to allow future implementation of the object skipping feature in TCOSS.

The send trace shows a line indicating that object skipping is active instead of stream optimization. The receive trace writes a line for each actually skipped child object.

Job Handling

Any document with one or more recipients may now be handled as "job" by TCOSS. Typically job handling will be used for messages with several hundred recipients. It involves the following features:

- job folder view
- job start actions
- job end actions
- change priority of all send orders of a job
- cancel all send orders of a job
- correct individual send order(s) of a job
- get a list of all users entitled to post jobs

Setting the job options of a message

All job options of a message are set when the message is posted using the INT_OPTIONS child object of the message header.

New flags in the INT_OPTIONS child of SET_HEADER:

JOB_START (=128)	requests job start actions
JOB_END (=256)	requests job end actions
JOB_HANDLING (=512)	requests general job handling

The JOB_HANDLING flag marks a message as job causing it to appear in the job folder. It has to be set if the JOB_START or the JOB_END flag is used.

Setting the JOB_START flag requests the job start actions as defined in the user profile of the message originator.

Setting the JOB_END flag requests the job end actions as defined in the user profile of the message originator.

Note: The job options are set once for the whole message using the INT_OPTIONS child object of the message header. It is not possible to set job options for individual recipients using the INT_OPTIONS child object of the recipient entry.

Defining job start and job end actions in the user profile

The user profile content (SET_US_CONTENT) has been extended with two additional action lists holding the job start and job end actions:

L_ACTIONS_JOB_START	job start actions
L_ACTIONS_JOB_END	job end actions

Both lists may contain entries of type SET_ACTION or SET_ENTRY_RS.

As for other action lists, the presence of the job start and job end action lists in the user profile has to be signalled by setting the appropriate flag in the INT_ACTIONS child of the user store entry:

New flags in the INT_ACTIONS child of SET_ENTRY_US:

JOB_START (=128)	job start actions list contained in user profile
JOB_END (=256)	job end actions list contained in user profile

Job folder views

Two new folder views are provided for the job management:

- an overview with one entry per job
- a details view containing all mail entries of a particular job

The following flags in the INT_FOLDER_TYPE child object of SET_FOLDER_MAIL_SYS have been defined:

JOB_FLDR_OVERVIEW (=64)	job folder overview
JOB_FLDR_DETAILS (=128)	job folder details

Both flags have to be used alone, not combined with each other or with other folder type flags.

Job overview folder

The job overview is requested by setting the JOB_FLDR_OVERVIEW flag in INT_FOLDER_TYPE. Filtering of the job overview folder is optimized for the message originator (TS_ORIGINATOR child of SET_DISP_FILTER set to a specific user ID without wildcards).

The display list returned by the job overview folder contains mail entries, one for each job. The mail entry shown for a job is not one of the mail entries generated for each recipient, it is an extra entry generated by the mail system to provide the job folder and the job end event handling. To clearly mark this entry as different from regular mail entries its message type is set to the constant JOB.

The following child objects of the job overview folder mail entries contain useful information:

TS_FILE_NAME	...12-digit decimal number, may be used as job ID
TS_ORIGINATOR	...user ID of message originator
TS_ORIGINATOR_INFO	...descriptive string of message originator
TS_RECIPIENT	...equal to TS_ORIGINATOR
INT_MSG_TYPE	...new constant JOB (=25)
INT_ER_RECIPIENT	...total number of send orders (=recipients) of this job
INT_ER_ALT_ADDR_LEFT	…number of open send orders
INT_DEL_TYPE	…number of failed send orders

It is possible to open the associated message using an entry from the job overview folder as selector entry in the mail system’s permanent store content (SET_PS_CONT_MAIL_SYS).

Job details folder

The job details folder is requested by setting the JOB_FLDR_DETAILS flag in INT_FOLDER_TYPE. The display filter has to contain the job ID (TS_FILE_NAME) value from a specific job. Additional filter values may be used, the filtering on TS_FILE_NAME and TS_RECIPIENT is optimized.

The job details folder returns all mail entries of a job, one for each active recipient in the message header, and in the same order as the recipients.

Note: The job details are only provided for a specific job. It is not possible to get the details of several or all jobs in a single call by using wildcards for the job ID.

Job start actions

Job start actions are requested by setting the JOB_START flag (combined with the JOB_HANDLING flag) in the INT_OPTIONS child of the message header. The actions itself are defined in the user profile of the message originator.

The job start actions consist of one or more additional send orders for the message, generated according to the action list in the user profile. The job start actions, if requested and defined, are triggered by the successful posting of the message.

The following child objects of the job start mail entries contain useful information:

TS_FILE_NAME	... 12-digit decimal number, ”job ID”
TS_ORIGINATOR	... user ID of message originator
TS_ORIGINATOR_INFO	... descriptive string of message originator
INT_ER_RECIPIENT	... total number of send orders (=recipients) of this job
INT_EVENT_TYPES	... flag JOB_START set

Job end actions

Job end actions are requested by setting the JOB_END flag (combined with the JOB_HANDLING flag) in the INT_OPTIONS child of the message header. The actions itself are defined in the user profile of the message originator.

The job end actions consist of one or more additional send orders for the message, generated according to the action list in the user profile. The job end actions, if requested and defined, are triggered as soon as all send orders of the message are (positively or negatively) terminated.

At the same time the job end actions are triggered, the job disappears from the job overview folder. Actually the additional mail entry generated for the job overview folder is activated to trigger the job end actions. If any of the job end send orders defined in the user profile can’t be created (e.g. because of an invalid recipient), the mail entry triggering the actions stays in the user’s in-box, otherwise it is auto-terminated.

The following child objects of the job end mail entries contain useful information:

TS_FILE_NAME	... 12-digit decimal number, ”job ID”
TS_ORIGINATOR	... user ID of message originator
TS_ORIGINATOR_INFO	... descriptive string of message originator
INT_ER_RECIPIENT	... total number of send orders (=recipients) of this job
INT_DEL_TYPE	... number of failed send orders
INT_EVENT_TYPES	... flag JOB_START set

The number of successful send orders may be calculated as total number of send orders minus number of failed send orders.

Termination handling of job send orders

Send orders belonging to a job are deleted immediately or not at termination depending on the setting of the DEL_ENTRY_POS and DEL_ENTRY_NEG flags in INT_TERMINATION. Usually job send orders will only have the DEL_ENTRY_POS flag set. This means that send orders are deleted (and disappear from the job details folder) at positive termination, but stay visible at negative termination as long as the job is active.

The flags DEL_ENV_POS and DEL_ENV_NEG of the job send orders should be set in accordance with the DEL_ENTRY_POS and DEL_ENTRY_NEG flags, because setting the DEL_ENV_POS / NEG flag automatically implies the DEL_ENTRY_POS / NEG flag.

All send orders of the job are deleted at the end of the job, independent of the DEL_ENTYR_POS and DEL_ENTRY_NEG flag settings. The end of the job is defined as the time when its last send order terminates.

The extra send order with message type JOB (INT_MSG_TYPE = JOB) providing the job overview folder is deleted from the mail system at the end of the job. It may be written into the short term archive depending on the setting of the ARC_POS flag in the INT_TERMINATION child object of the message header.

Termination setting of job start / end actions

Job start and end actions defined in the user profile should have an INT_TERMINATION value with at least the flag DEL_ENV_POS set so that the message file is deleted after termination of the action send orders.

Correcting job send orders

Any send order belonging to a job may be corrected with the usual posting-for-correction procedure. But different form regular messages, correcting a job send order does not create a new file in the mail system. The newly posted message is not stored, only its header is evaluated and new send orders for the existing message file are created.

Actually the correction of job send orders works in the same way as correction of messages in the “+MAIL5V” folder which are created with “..commands”. In the case of jobs this correction mode is selected automatically by the JOB_HANDLING flag in the INT_OPTIONS child object of the mail entry.

Corrected send orders will be displayed in the job details folder at the end of the list, the original (incorrect) send order disappears from the job details.

Reactivating or canceling job send orders

It is possible to reactivate or cancel individual send orders of a job in the usual way (setting INT_STATE to ST_POSTED or ST_CANCELLED respectively). Of course this is only possible as long as the job is active, because all send orders are deleted at the end of the job.

Changing the priority of send orders

It is now possible to change the priority level of individual send orders by setting INT_STATE to ST_PRIORITY_CHANGE (=151) and inserting the new priority.

Canceling a job

All send orders belonging to a job may be cancelled with a single TCSI action on the job entry from the job overview folder.

To do so, the message is opened from the mail system’s permanent store content (SET_PS_CONT_MAIL_SYS) using the entry from the job overview folder as selector entry. The INT_STATE value of the mail entry is set to ST_CANCELLED and the message is closed. The INT_LAST_USER_ACTION value may also be set to CANCELLED.

The system will then try to cancel all open send orders of the job. On send orders currently active a “cancel during transmission” is tried, but some send orders may be in a phase where cancel during transmission is no longer possible.

The cancel operation on the job overview mail entry always succeeds, even if some of the job’s mail entries could not be cancelled. The job overview mail entry itself is not cancelled, but it will disappear automatically once all mail entries belonging to this job are terminated. The job end actions will be triggered, independent of whether the job is cancelled or runs out normally.

Changing the priority level of a job

The priority level of all send orders belonging to a job may be changed with a single TCSI action on the job entry from the job overview folder, in a way analogous to the job cancel procedure.

The message is opened from the mail system’s permanent store content (SET_PS_CONT_MAIL_SYS) using the entry from the job overview folder as selector entry. The INT_PRIORITY value of the mail entry is set to the new priority level. The priority change action is signaled by setting INT_STATE to ST_PRIORITY_CHANGE (=151). The INT_LAST_USER_ACTION value and the other read / write child objects of the mail entry may also be set. Then the message is closed.

The system will try to adapt the priority level of all send orders of the job. Send orders which are active at the time of the priority change are locked and their priority is not changed.

The administrator may do retries to catch the send orders which were active and therefore not updated. The system keeps the priority of all send orders in memory; the priority change action is optimized to skip the send orders which already have the desired priority.

Update of job overview send order

All updates done to a job overview send order are applied in a loop to all send orders of the job, but not to the job overview send order itself (as in the above examples of cancel and priority change).

Also all read / write child objects of the mail entry, which exist in the job overview send order, are taken and inserted into all job send orders. Read / write child objects of the mail entry are:

INT_STATE
INT_PRIORITY (only if INT_STATE = ST_PRIORITY_CHANGE)
INT_LAST_USER_ACTION
TIME_ACTION
TIME_LATEST
INT_OPEN_RETRIES
INT_SUSP_DUPL
TS_LAST_MDA_ACTION
TS_LAST_MDA_NOTE
TS_DOC_NR
TS_COST
TS_DURATION

If any of the above child objects should not be set in all job send orders, it has to be deleted from the job overview send order before doing the update.

Re-running a job

The functionality to re-run a job, i.e. pick all negatively terminated send orders of a job from the short-term archive and compile them into a new job, will be implemented outside TCOSS in a separate module (TC/NOTIF).

The message which is finally posted to re-run a job will contain the original job ID in the TS_FILE_NAME child object of the message store entry. If posting of the message succeeds the original job ID will appear in the ENV_NAME_POSTED field of all generated mail entries, while TS_FILE_NAME will contain a new, automatically assigned job ID.

Job overview folder in the short-term archive

If the ARC_POS flag had been set in the INT_TERMINATION child object of the message header, job overview mail entries are written to the short term archive.

A job overview folder may be obtained from the short term archive by setting INT_FOLDER_TYPE to JOB_FLDR_OVERVIEW. The job overview folder without any additional filtering is optimized in the short-term archive.

Job details folder in the short-term archive

An optimized job details folder with folder type JOB_FLDR_DETAILS is not implemented in the short-term archive.

Instead one may obtain a similar view by using folder type OUT_FLDR and a display filter with the following filter values (TS_FILE_NAME and TS_ENV_NAME_POSTED are used alternatively):

TS_ORIGINATOR ... message originator

TS_FILE_NAME ... job ID

TS_ENV_NAME_POSTED ... original Job ID, to get entries form original job and re-runs

---

To get messages and notifications from the TCOSS server, the open-for-transfer function is used.

a) Prepare a SET_SELECTOR_MS_MAIL, which holds the parameters of the open-for-transfer function**.**

Set the following child objects:

TS_RECIPIENT: In-queue to be selected (user-id of the gateway client)

INT_OPTIONS: select if the original content should be included in delivery or non-delivery notifications (set to DEL_NOTIF | NON_DEL_NOTIF by a gateway which needs the original content for correlation)

Optionally the following child objects may also be set:

INT_MAX_ENTRIES: Maximum number of returned mail entries (default: 128)

INT_TIMEOUT: Timeout of returned mail entries in seconds (default: 3600s = 1 hour timeout)

Example:

```
ohh_c_create( h_appl_session, SET_SELECTOR_MS_MAIL, &h_selector_ms_mail);
ohh_c_ts_put( h_selector_ms_mail, TS_RECIPIENT, "TCGATE");
ohh_c_int_put( h_selector_ms_mail, INT_OPTIONS, DEL_NOTIF | NON_DEL_NOTIF);
```

b) Call the open-for-transfer function:

```
ret = ohh_c_open( h_set_ps_cont_mail_sys, h_selector_ms_mail, &h_transfer_env, &dum);
if (ret == OK) { ...
```

The function return code is OK or ERR_OBJ_NOT_EXIST, if the selected in-queue was empty.

c) The open-for-transfer function returns (if ok) a handle to a transfer envelope, which is processed by the gateway.

Layout of transfer envelope:

```
SET_TRANSFER_ENVELOPE
  UN_CONTENT / L_ENV_CONT   (the message content)
  L_ENTRIES                 (list of active recipients)
    SET_ENTRY_MS_MAIL
    SET_ENTRY_MS_MAIL
    ...
```

In the UN_CONTENT child all linked attached objects are converted into embedded attached objects.

Note that if the same message is sent to more than one recipient of the same in-queue (same gateway), the message is only transferred once, together with a list of recipients. The gateway may limit this grouping function by setting the maximum number of recipients it will accept in one transfer (child INT_MAX_ENTRIES of SET_SELECTOR_MS_MAIL). The grouping function may be switched off by setting INT_MAX_ENTRIES to 1.

The open-for-transfer function may return messages or notifications. The actual type of the entries is determined by the INT_MSG_TYPE child. Messages and notifications are never mixed in one list. In the current implementation notifications (INT_MSG_TYPE = NOTIF) are always sent separately, only recipients of a message (INT_MSG_TYPE = NORM or ROUTING) are grouped together.

The returned mail entries are not locked on the TCOSS server but removed from the in-queue and put into a separate queue with timeout supervision. INT_STATE is set to ST_SENDING. Normally the state will be updated by a transfer notification to the TCOSS server. If the timeout expires before a notification is received, the mail entry is put back into the in-queue of the gateway with INT_STATE set to ST_WAIT_RETRY (and may be returned again by an open-for-transfer function). The INT_SUSP_DUPL flag is also set.

d) Generate a notification to the TCOSS server to update the mail entry's state and to acknowledge the message (or notification) transfer. (see separate chapter on notifications).

If a notification was transferred from the TCOSS server to the gateway, terminate the operation by setting INT_STATE = ST_DELIVERED | ST_TERMINATE in the notification to TCOSS.

If a message was transferred, and if it could be delivered immediately to the mail user, set also INT_STATE = ST_DELIVERED | ST_TERMINATE.

Otherwise set INT_STATE = ST_SENT and provide a final (positive or negative) notification at a later time.

Each mail entry returned by the open-for-transfer function requires a notification, which may be grouped together into one transfer envelope or sent separately. (In the current implementation the number of notifications which will be accepted in one transfer envelope is limited to 128.)

Passing Transfer Envelopes to the Mail System

Function ohh_c_save_at in the mail system will accept a transfer envelope and return a transfer notification:

ohh_c_save_at(h_set_ps_cont_mail_sys, AFTER_END, h_transfer_env, TRUE);

After the save_at (if the return value is OK), h_transfer_env is a handle to a transfer notification, which gives the result of the transfer for each mail entry of the original transfer envelope. This function, together with the open-for-transfer, is used for the TC-TC routing feature.

The transfer envelope content, if any, is stored unchanged in a mail file (its header is not evaluated; it may not even have a header).

Each mail entry in L_ENTRIES generates a new mail entry (routing of message or routing of notification) or causes a mail entry waiting for notification to be updated (delivery notification arrived at entry node or transfer notification at next node).

The following default values for come child objects of SET_ENTRY_MS_MAIL have been implemented to simplify the input of notifications:

INT_MSG_TYPE:	NOTIF
TS_NORMALIZED_ADDR:	own TCOSS node
TS_NORMALIZED_ORIG:	"SY:" (any valid originator would do)

In case a new mail entry is generated, some of its child objects are the result of special processing of the input mail entry (to support the TC - TC routing feature):

INT_MSG_TYPE: ROUTING if input was NORM or ROUTING, NOTIF if input was NOTIF

INT_TERMINATION: DEL_* | NOTIF_* for NORM or ROUTING, DEL_* for NOTIF

TS_ORIGINATOR, TS_NORMALIZED_ORIG: "SY:"

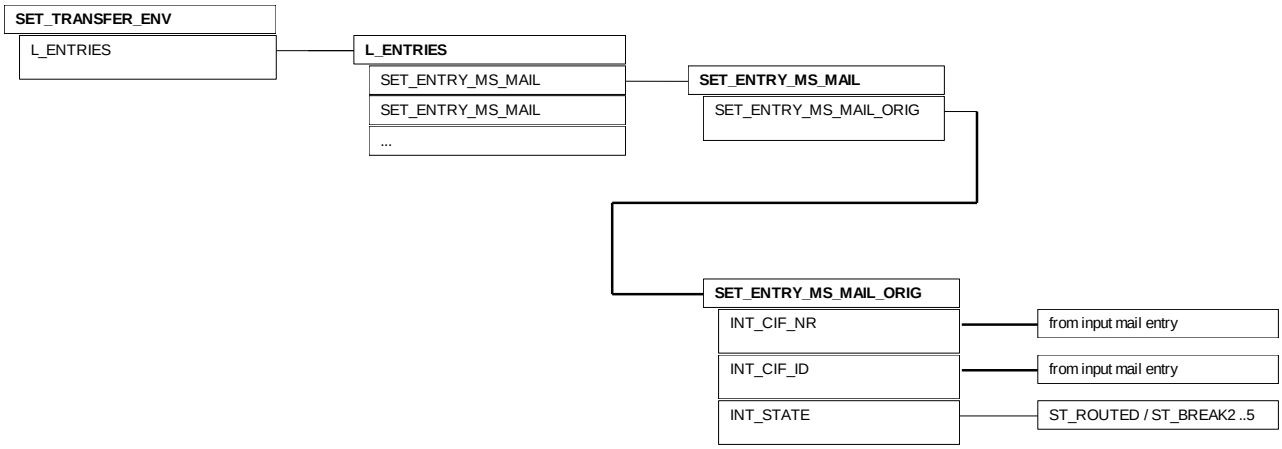
TS_RECIPIENT, TS_NORMALIZED_ADDR, TS_LOCALIZED_ADDR: result of rr99 process on input normalized address

TS_NODELIST: input node list extended with current node

Some children of SET_ENTRY_MS_MAIL are transferred unchanged from the input mail entry: TS_REF, INT_NPAG, INT_DOC_CLASS, TS_RECIPIENT_INFO, TS_ORIGINATOR_INFO, TS_ENV_NAME_POSTED, INT_DEL_TYPE, INT_PRIORITY, INT_SUSP_DUPL, TS_COST_CENTER, INT_ER_RECIPIENT, INT_ER_ALT_ADDR_LEFT, INT_OPTIONS and SET_ENTRY_MS_MAIL_ORIG with all its children.

All other children get the default value of a new entry.

Layout of transfer notification returned after save_at of a transfer envelope:



The INT_STATE child of SET_ENTRY_MS_MAIL_ORIG gives the result of the transfer for the corresponding input mail entry (state ST_ROUTED = ok). The input mail entry referred to is identified by the children INT_CIF_NR and INT_CIF_ID.

Put Notifications into TCOSS Server

a) Prepare a transfer envelope containing one or more (max. 128) notifications:

SET_TRANSFER_ENV

L_ENTRIES

SET_ENTRY_MS_MAIL with child SET_ENTRY_MS_MAIL_ORIG

SET_ENTRY_MS_MAIL with child SET_ENTRY_MS_MAIL_ORIG

....

Set the following child objects in SET_ENTRY_MS_MAIL_ORIG:

INT_CIF_NR: INT_CIF_NR of original mail entry returned by the open-for-transfer function

INT_CIF_ID: INT_CIF_ID of original mail entry

INT_STATE: ST_SENT ... acknowledge transfer to gateway (further notification follows)

ST_DELIVERED | ST_TERMINATE ... final delivery notification

ST_BREAK2 .. negative, TCOSS will do 8 retries

ST_BREAK4 .. negative, TCOSS will do 1 retry

ST_END_OF_RETRIES .. negative, no retries (non delivery notification)

Optionally, additional information may be returned to the TCOSS server by setting the following child objects of SET_ENTRY_MS_MAIL_ORIG:

TS_COST (cost of sending for TCOSS cost accounting feature)

TIME_ACTION (time of delivery)

TS_LAST_MDA_ACTION (2-character error code defined by TCOSS)

TS_LAST_MDA_NOTE (string describing e.g. reasons for non-delivery)

b) Pass the notification to the TCOSS server with

ohh_c_save_at(h_set_ps_cont_mail_sys, AFTER_END, h_transfer_env, TRUE);

c) Forget the transfer envelope (it is changed by the save_at function) with

ohh_o_forget(h_transfer_env);

Extended Correlation Information

The correlation information in notifications may be of 3 types:

1) INT_CIF_NR and INT_CIF_ID: This correlation information works for all notifications. It is always used for the first notification.

2) TS_FILE_NAME and INT_SEARCH_ID: If INT_CIF_NR and INT_CIF_ID are missing, the correlation to the original mail entry is done with TS_FILE_NAME and INT_SEARCH_ID.

The INT_SEARCH_ID must have been set in a previous notification of type 1 with INT_STATE = ST_SENT. INT_SEARCH_ID must not be zero.

The gateway user, which sets INT_SEARCH_ID, must also ensure that it is unique for a particular mail entry (among all mail entries for the same document). It may use e.g. a checksum of the recipient or an ID generated by the connected mail system. (It is useless to include the file name in a checksum, as the file name is checked separately.)

Some bits of INT_SEARCH_ID will be reserved on the link interface to encode the link type. This ensures that IDs from different links do not overlap.

3) TS_LAST_MDA_NOTE and INT_SEARCH_ID: If INT_CIF_NR, INT_CIF_ID and TS_FILE_NAME are missing, the correlation to the original mail entry is done with TS_LAST_MDA_NOTE and INT_SEARCH_ID.

TS_LAST_MDA_NOTE and INT_SEARCH_ID must have been set in a previous notification of type 1 with INT_STATE = ST_SENT. INT_SEARCH_ID must not be zero.

INT_SEARCH_ID speeds up the access to the original mail entry and should be unique in almost all cases. The combination of TS_LAST_MDA_NOTE and INT_SEARCH_ID must be unique on the mail server. The gateway user is responsible for setting unique correlation information.

Transfer Client

A transfer client sets up sessions on two TCOSS servers and transfers envelopes and notifications. Example of command sequence for one direction (from server1 to server2):

```
res = ohh_c_open( h_set_ps_mail_1, h_set_selector_ms_mail, &h_env, ..)
if (res == OK) {
  res = ohh_c_save_at( h_set_ps_mail_2, AFTER_END, h_env, ..)
  if (res == OK)
    ohh_c_save_at( h_set_ps_mail_1, AFTER_END, h_env, ..)
  ohh_o_forget( h_env)
  }
```

The ohh_c_open function on server1 returns a handle to a transfer envelope or ERR_OBJ_NOT_EXIST, if the selected queue was empty. With the first ohh_c_save_at the transfer envelope is sent to server2, the result of this operation (in the form of a transfer envelope containing transfer notifications) is passed back to server1 with the second ohh_c_save_at.

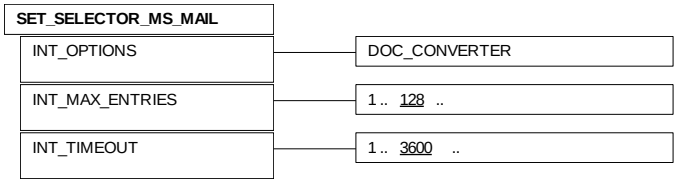
If the transfer to server2 was not ok, server1 gets no response. A timeout handling of the activated mail entries on server1 ensures retries and setting of the INT_SUSP_DUPL flag.

Interface to Document Converter

Polling the Document Converter Queue

The DC prepares a SET_SELECTOR_MS_MAIL with INT_OPTIONS set to DOC_CONVERTER. Optionally INT_MAX_ENTRIES and INT_TIMEOUT may be set as well.

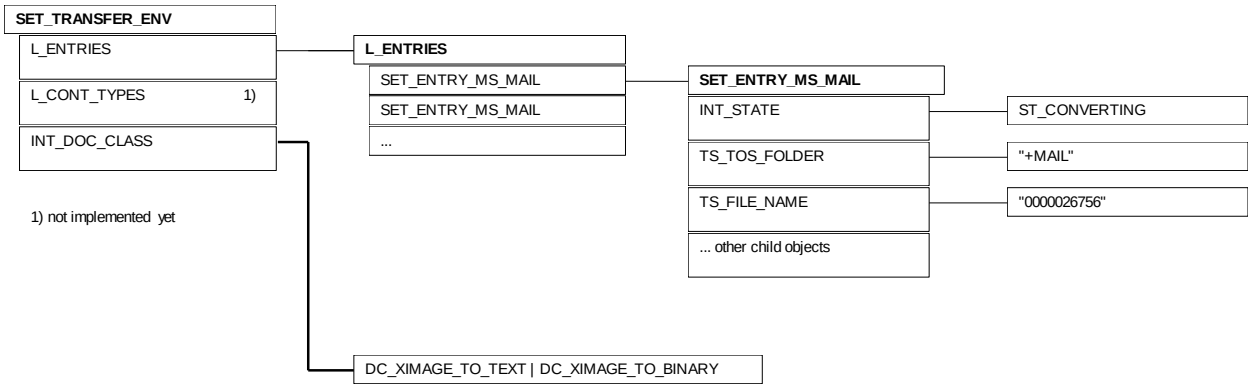
In addition to the DOC_CONVERTER flag in INT_OPTIONS the DC may specify one or more flags of the group [DC_XIMAGE_TO_TEXT, DC_XIMAGE_TO_BINARY, DC_MUST_SIGN]. TCOSS returns only documents that match the specified flags. If the DOC_CONVERTER flag is specified alone, the conversions DC_XIMAGE_TO_TEXT and DC_XIMAGE_TO_BINARY will be selected for compatibility reasons.



The DC calls the open-for-transfer function:

```
ret = ohh_c_open( h_set_ps_cont_mail_sys, h_selector_ms_mail, &h_transfer_env, &dum);
```

The function return code is OBJ_NOT_EXIST if the document converter queue was empty. Otherwise OK is returned and a handle to a transfer envelope:



The returned transfer envelope contains a list of mail entries which all refer to the same document.

Converting a document

The DC prepares a SET_ENTRY_MS and copies TS_TOS_FOLDER and TS_FILE_NAME from one of the mail entries. It then opens the document from the message store:

ret = ohh_c_open(h_set_ps_cont_message_store, h_entry_ms, &h_ env, &dum);

The DC performs the conversions and closes the document:

ret = ohh_o_close(h_env, FALSE);

If there are no changes to the document (already converted or no conversions necessary or conversion failed) the close is replaced by:

ret = ohh_o_forget(h_env);

Updating mail entries

After the DC has done the conversion of the document it updates INT_STATE and TS_LAST_MDA_ACTION (2-byte error code) of all returned mail entries in L_ENTRIES.

The values of INT_STATE and TS_LAST_MDA_ACTION reflect the success of the conversion operation:

INT_STATE	description	TS_LAST_MDA_ACTION
ST_CONV_END	conversion successful	„ „
ST_BREAK1 ..5	retry conversion at a later time	„S1“
ST_CONV_FAILED	document conversion failed	„S1“

Finally the transfer envelope is closed to make the changes to the mail entries permanent:

ret = ohh_o_close(h_transfer_env, FALSE);

It is also possible to cancel all temporary changes to the mail entries with

ret = ohh_o_forget(h_transfer_env);

In that case the default error handling of the mail entries by TCOSS will apply (retry after 1 hour).

Examples of state sequences

without document conversion: ST_POSTED -> ST_WAIT_SEND -> ...

with conversion: ST_POSTED -> ST_WAIT_CONV -> ST_CONVERTING -> ST_CONV_END -> ST_WAIT_SEND -> ...

conversion retry: ST_POSTED -> ST_WAIT_CONV -> ST_CONVERTING -> BREAK1..5 -> ST_WAIT_CONV -> ...

conversion fails: ST_POSTED -> ST_WAIT_CONV -> ST_CONVERTING -> ST_CONV_FAILED

Error handling

The DC will detect and handle the cases

a) no conversion necessary
b) document already converted

A timeout of one hour (default value) applies to all mail entries returned by the initial open-for-transfer function. If the document converter fails to close the transfer envelope within one hour, the mail entries will be put back into the converter queue.

Restrictions

- There is only one type of document converter
- Conversion is done only if the address had been entered using a service, e.g. ..S,N=FAX$66133 or posting via TCSI.
- Linked attached objects are not converted.
- Document conversion is started at send time.
- If a document has a number of send orders which require document conversion and others which don't, two versions of the same document may be sent and archived. Send orders which do not require conversion will not wait for the conversion to be completed.
- If a document requires conversion and has send orders with different send times, the document will turn up for conversion several times (once for each send time).
- The list of alternative contents has to be sorted (with the TCI alternative before the text) for the correct working of the alternative content selection by the TAM channels.
- Inbound routing via NN99 requires a change in TCfW (specify service with user ID) to allow conversion of inbound documents
- There is no check if a document, which would normally require conversion, has been converted already (The alternative content list, if any, is not evaluated).

3.6.4.3 SET_PS_CONT_MAIL_ARC

Derived from PS_CONT_MAIL.

Posting of envelopes or changing of entries is not possible. The delete function (with folder entry as selector object) is supported. Delete removes the message immediately from the archive. If the message can not be deleted an error is returned.

***ohh_c_delete* (h_set_ps_cont_mail_arc, h_set_entry_ms_mail_arc)**

Removes the message immediately from the archive. Returns also OK if the child was not existing.

rrors: ERR_NOT_PERMITTED open entries exist or message still in mail.

Storing a User-Defined Log Entry in the Short Term Archive

ohh_c_save_at(*h_set_ps_cont_mail_arc*,.../* handle to permanent store content */
h_set_entry_ms_mail_arc , .../* handle to selector entry */
h_set_entry_ms_mail_arc,.../* handle to entry to be stored */
TRUE)

This is a procedure similar to the posting of a transfer envelope. It is done in the mail archive and the mail archive entry holds a user-defined SET_ATTRIBUTE.

The selector entry is not relevant, except that it must be of type SET_ENTRY_MS_MAIL_ARC. One may use the handle of the entry to be stored also as selector.

A log user may be specified in the mail archive entry, by default the logged-in user is written to the TS_LOG_USER field of the log entry. The time of archiving is set automatically, if TIME_ACTION is specified it will be ignored. The same applies to the INT_MSG_TYPE child which always gets the constant value LOG_ENTRY. Other child objects of the mail archive entry (all except TS_LOG_USER, TIME_ACTION, INT_MSG_TYPE and UN_CONTENT) are not stored.

After the ohh_c_save_at call the mail archive entry will hold all values as archived, plus a binary block object for internal use which points to the position of the entry within the short term archive.

Restriction: The TCSI stream size of the archive entry including the user-defined attribute must not exceed 4 KB. If the object is larger the ohh_c_save_at() function fails returning error code ERR_B_STREAM_SIZE.

Restriction: All strings prefixed with “INT_”, “TS_” or “SET_” are reserved for internal use and should not be specified as user-defined field or structure names. This restriction is caused by the fact that a number of fields like “TS_RECIPIENT” are indexed by TC/Archive and specified in the same way as user-defined attributes in an archive search filter.

Recommendation: User-defined attributes should have exactly two hierarchy levels as in this example:

```
UN_CONTENT/SET_ATTRIBUTE
  UN_NAME        ... “EXAMPLE_LOG” (structure name)
  UN_VALUE/L_ATTRIBUTES   ... list of fields
    SET_ATTRIBUTE
      UN_NAME   ... “field1” (field name)
      UN_VALUE   ... 35 (integer value)
    SET_ATTRIBUTE
      UN_NAME   ... “field2” (field name)
      UN_VALUE   ... “blabla” (string value)
    SET_ATTRIBUTE
      UN_NAME   ... “field3” (field name)
      UN_VALUE   ... “lall” (string value)
```

The reason for this recommendation is that attributes with two hierarchy levels are easily processed by TC/Report, indexed completely by TC/Archive and displayed by TCfW.

The user-defined SET_ATTRIBUTE log entry is packed into a mail archive entry for storing in the short term archive. This SET_ENTRY_MS_MAIL_ARC contains the following sub-objects:

SET_ENTRY_MS_MAIL_ARC

INT_MSG_TYPE	... constant LOG_ENTRY
TIME_ACTION	... time of archiving
TS_LOG_USER	... user ID (logged by)
BLK_BINARY	... position in short term archive (TCOSS internal)
UN_CONTENT/SET_ATTRIBUTE	... user-defined log entry

The new folder type OUT_FLDR_LOG_ENTRIES gives a folder view of all log entries, filtering on TS_LOG_USER is optimized. The log user in the filter is handled like an originator, i.e., the log folder may be combined with other out-folder types by adding OUT_FLDR_LOG_ENTRIES as flag.

The SET_ATTRIBUTE content of the log entries is already contained in the entries of the folder view and may be opened directly, a separate open in the permanent store content with the folder entry as selector is not required.

3.6.4.4 SET_PS_CONT_MS

SET_ENTRY_MS mandatory fields: TS_TOS_FOLDER, TS_FILE_NAME

3.6.4.5 SET_PS_CONT_US

SET_ENTRY_US mandatory fields: TS_USER_ID

3.6.4.6 SET_PS_CONT_RS

SET_ENTRY_RS mandatory fields: TS_SECTION, TS_RECP_ID

SET_DL mandatory fields: TS_SECTION, TS_RECP_ID

3.6.4.7 SET_PS_CONT_SS

selector object types:

SET_ENTRY_SS mandatory field: TS_NAME

3.6.4.8 SET_PS_CONT_REGS

Deleting of a child de-installs the workstation.

selector object types:

SET_ENTRY_REGS...selector object must be obtained from folder

The ohh_c_open function in the registry store content has an additional functionality: If the registration referred to by the selector entry does not exist and if there is a valid, not fully used license of the specified type, a new registration will be created. It may be used to check a license without opening a new server session (e.g. image license check).

Registrations are deleted after 30 days of inactivity.

At TCOSS startup all registrations which refer to invalid licenses are deleted.

If a user is deleted in TCOSS (delete recipient type user in recipient store), all registrations referring to this user are also deleted.

3.6.4.9 SET_PS_CONT_ARCHIVE

selector object types:

SET_ENTRY_ARCHIVE mandatory fields: INT_VOLUME, INT_ENTRY_NR

3.6.5 SET_STATE_N_MAX

The standard state for permanent stores.

Child objects:

INT_N_OF_ENTRIES	R/O	The current number of entries in the store.
INT_MAX_ENTRIES	R/O	The maximum number of entries the store can hold

.

3.6.6 SET_STATE_N_MAX_REG

The state of the registration store.

Child objects:

the child objects of SET_STATE_N_MAX plus:

INT_NREG_TCFW		R/O	The current number of registrations for TCfW.
INT_STATUS	-	-	holds a number of flags:

LICSTAT_NEW (=1)	TCOSS operates in new licensing mode
LICSTAT_TEST (=2)	TCOSS operates in test mode, no license checks are done
LICSTAT_RESTR (=4)	TCOSS operates in restricted mode, not sending and receiving

Two functions are activated by setting a single INT_STATUS bit and closing or saving SET_STATE_LICENSE_STORE:

INT_STATUS = LICSTAT_NEW	... switch TCOSS to new licensing mode
INT_STATUS = LICSTAT_TEST	... switch TCOSS to test mode

The licensing test mode disables all license checks for a test phase of 7 days. The test phase is followed by a recovery phase of 30 days with regular license checking. The test phase may be activated in both old and new licensing mode.

3.6.7 SET_STATE_LICENSE_STORE

The state of the license store.

Child objects:

the child objects of SET_STATE_N_MAX plus:

TIME_END_OF_TEST		R/O	The test phase end time.
INT_NREG_PRO	-	R/O	The current number of registrations for TC FAX DESK PRO.

3.6.8 SET_STATE_ARCHIVE in SET_SYS_RESOURCES

The state of a message server's short term archive.

Child objects:

INT_N_OF_FILE_ENTRIES	R/O	-	-

The current number of file entries in the archive (the number of file entries for terminated envelopes in the mail area).

INT_SIZE_OF_ENVS	R/O	-	-

The current total size of the envelopes in the archive (the total size of terminated envelopes in the mail area).

TIME_OLDEST_ENV	R/O	-	-

The oldest envelope still in the archive.
An entry still has its envelope when its TIME_ACTION is greater than this value.

TIME_OLDEST_ENTRY	R/O	-	-

The oldest entry still in the archive.

3.6.9 SET_STATE_ARCHIVE in SET_PS_ARCHIVE

The state of an archive server.

Child objects:	**attributes**	**default**	**value-restr.**
INT_ACTIVE	-	WAIT / CONTINUE	-

The current state of the task fetching new messages from the TCOSS message server. This task may be halted / continued by setting INT_ACTIVE to WAIT / CONTINUE and saving SET_STATE_ARCHIVE.

INT_VOLUME	R/O	-	-

The total number of archive volumes (online and offline).

TIME_CREATED	R/O	-	-

Start date and time of the archive, this is also the time of the oldest entry in the archive.

TIME_OLDEST_ENTRY	R/O	-	-

The oldest entry on disk (online).

TIME_OLDEST_ENV	R/O	-	-

The oldest document on disk (online).

TIME_ACTION	R/O	-	-

Date and time of the most recent entry in the archive.

INT_N_OF_ENTRIES	R/O	-	-

The number of entries on disk (online).

INT_MAX_ENTRIES	R/O	-	-

The total number of entries in the archive (online and offline on CDs).

INT_N_OF_FILE_ENTRIES	R/O	-	-

The number of documents on disk (online).

INT_MAX_FILE_ENTRIES	R/O	-	-

The total number of documents in the archive (on disk and offline on CDs).

3.6.10 SET_SYSTEM

Holds the central server values such as the number of active user sessions, the CPU number, the key used for licensing and so on.

Child objects:	**attributes**	**default**	**value-restr.**
SET_LICENSE	-		

The TCfW license object. To set the license values at the server the license object is saved here. If the license object holds a valid key for the server the server will accept it. You can check this by opening this child and checking the values. When a license object is saved here the validity of the keys is checked by the server.
If the key is invalid the number of maximum registrations will fall back to 5.

SET_LICENSE_PRO	-		

The TC FAX DESK PRO license object. Same function as SET_LICENSE.
If the key is invalid the number of maximum registrations will fall back to 0.

TIME_SYS_RESOURCES	-		

The server's system resources.

TIME_SERVER	-		

The server's system time.

TS_PROGRAM_VERSION	R/O	-	-

The server's program version.

INT_N_OF_USER_SESSIONS	R/O	-	-

The current number of user sessions. For user sessions a 24-hour non-activity time-out is defined. When the time-out expires all objects locked within the server for a particular session are released.

L_NUMBER_SERIES	R/O	-	-

The list of number series.

L_CHANNEL_STATUS	R/O	-	-

The list of the statuses of the channels.

SET_NODE	R/O	-	-

The root of the hardware tree. The hardware tree represents the node structure of the TCOSS system the way it was configured.

TS_TC_MSG_ID	-	-	-

Provides unique message IDs. Every time the TS_TC_MSG_ID field is read with the ohh_c_ts_get() function, a new unique TCOSS message ID value is returned. It has the same format as the automatically generated TCOSS message IDs.

INT_ALERTING_ACTIVE	-	ACTIVE	ACTIVE / INACTIVE

The global alerting active / inactive setting

3.6.11 SET_LICENSE

Holds the values needed for TCfW licensing. This object is only supported for compatibility reasons. See SET_ENTRY_LICENSE for the complete licensing description.

Child objects:	attributes	default	value-restr.
TS_CPU_NO	R/O	-	-
The CPU number of the server. (12-character hex)			
INT_MAX_REGISTRATIONS	R/O	-	-
The current maximum number of registrations allowed for this server.			
TIME_KEY_VALID	R/O	-	-
The expiration time for the key.			
TS_CPU_KEY	-	-	
The key making the number of registrations valid.			
TS_NAME	-	-	
A descriptive string of the license.			

3.6.12 SET_LICENSE_PRO

Same child objects as SET_LICENSE. Holds the values needed for TC FXD PRO licensing. This object is only supported for compatibility reasons. See SET_ENTRY_LICENSE for the complete licensing description.

3.6.13 SET_NUMBER_SERIE

Holds the values for a number series.

Child objects:	attributes	default	value-restr.
TS_NRS_ID	R/O	-	‘A’ .. ‘Z’
The id of the number series.			
TS_FROM	-	-	-
The start value of the series.			
TS_TO	-	-	-
The end value of the series.			
TS_ORDER	-	-	-
The current value of the series.			
TS_CYCLE	-	-	-
The cycle of the series.			

3.6.14 SET_CHANNEL_STATUS

Holds the channel status.

Child objects:	**attributes**	**default**	**value-restr.**
TS_CHANNEL	R/O	-	“00” .. “31”
The decimal channel number.			
INT _ACTIVITY	-	-	CONTINUE / WAIT / QUERY / SERVER
The activity status of the channel.			
TS_REPRESENTS_1	R/O	-	-
Normally the channel group that is polled by the channel.			
TS_ANSWERBACK	R/O	-	-
The channel’s answerback.			
INT _STATUS_IN	R/O	-	IDLE / RECEIVING / BACKRECEIVING
The incoming status.			
INT _ STATUS_OUT	R/O	-	IDLE / SENDING
The outgoing status.			
INT_ERROR	R/O	-	NO / LINE_ERROR / TIMEOUT / NOT_LOADED
The error status of the channel.			

NO	no error
LINE_ERROR	e.g. telex line not connected
TIMEOUT	timeout condition on the TAM-TUM interface
NOT_LOADED	user module not loaded

INT_ NODE_NR	R/O	-	-
The number of the node where the user module operates.			
INT _STATUS	-	-	0 / DO_RELOAD_CONFIG
When DO_RELOAD_CONFIG is put to this child the channel reloads its configuration.			
INT_OPTIONS	-	-	RECEPTION_OFF (0) / RECEPTION_ON (1)
Reception enable / disable state, may be changed			
INT_TYPE	R/O	-	CHANTYPE_LOCAL,
Remote connection type (as set in config line 49, 1st and 2nd position), possible values: CHANTYPE_LOCAL (0), CHANTYPE_REMOTE_TUM (1), CHANTYPE_REMOTE_TAM (2). additional flag: CHANTYPE_DEDICATED (256), may be combined with CHANTYPE_REMOTE_TUM and CHANTYPE_REMOTE_TAM			
TS_NAME	R/O	-	-
line ID or line group as set in config line 37			
TS_SERVICE	R/O	-	-
The currently active service. It is empty if the channel is in idle state. If the channel is sending or receiving, the TS_SERVICE field is set to “FAX” or “VOICE” showing which service is currently active. Channels which support only one service like telex lines leave the TS_SERVICE field always empty.			

3.6.15 SET_NODE

The description of one node in the hardware tree.

Child objects:	attributes	default	value-restr.
INT_NODE_NR	R/O	-	-
The node number.			
TS_TYPE	R/O	-	-
The node hardware type. For a master node this is „Master“ or „Prim. Master“ or „Sec. Master“. For a slave node interface type, master - dot - slave number - dot - slot, e.g. „TC20(1.1.L0)“			
INT_LINK_DOWN	R/O	-	-
The link number in the node above, pointing down to this node.			
INT_LINK_UP	R/O	-	-
The local link number, pointing up to the node above.			
INT_ACTIVITY	R/O	-	STOPPED/ WAITING / NOBOOT / BOOTED / STARTED / MASTER/ DO_RESTART
The current activity of the node.			
SET_MEMORY	R/O	-	-
The node’s memory state.			
L_DISKS	R/O	-	-
The list of disks connected to the node.			
L_SUBNODES	R/O	-	-
The list of sub-nodes connected to the node.			

3.6.16 SET_MEMORY

Holds the memory status of a node.

Child objects:	attributes	default	value-restr.
INT_TOTAL_FREE	R/O	-	-
The total free memory at this node.			
INT_LARGEST_BLOCK	R/O	-	-
The largest free memory block at this node.			

3.6.17 SET_DISK_STATUS

Holds the disk status.

Child objects:	**attributes**	**default**	**value-restr.**
INT_DISK_NR	R/O	-	-

The disk number.

INT_DISK_STATUS	-	-	DISK_OK | MIRRORED | UPDATING

The status of the disk.

DISK_OK	this disk is OK
MIRRORED	the mirror disk is up to date
UPDATING	this disk is being updated from its mirror disk

When a desynchronization between the mirror disks occurs (both disks have been used independently) mirroring does not start automatically.
To start mirroring manually a value must be put to this child with the UPDATING flag set. This will result in loss of all data written to this disk after single operation started.

TIME_LAST_MIRROR	R/O	-	-

The time when single operation started.

INT_DISK_NR	R/O	-	-

The number of blocks changed since single operation started.

3.6.18 SET_SYS_RESOURCES

Holds the system resources of the server.

Child objects:	attributes	default	value-restr.
SET_AREA_MAIL	R/O for all below ADMIN	-	-
The descriptor of the mail area.			
SET_AREA_USER	R/O	-	-
The descriptor of the user area. (User = Total - Mail - Tech)			
SET_AREA_TECH	R/O for all below ADMIN	-	-
The descriptor of the tech area.			
SET_AREA_TOTAL	R/O	-	-
The descriptor of the total server area. Depending on the server's disk size.			
SET_STATE_MAIL	R/O	-	-
The state of the mail store.			
SET_STATE_ARCHIVE	R/O	-	-
The state of the archive store.			
SET_STATE_RS	R/O	-	-
The state of the recipient store.			

3.6.19 SET_AREA

The description of a TOS area.

Child objects:	attributes	default	value-restr.
INT_N_OF_FILE_ENTRIES	R/O	-	-
The current number of file entries in the area.			
INT_MAX_FILE_ENTRIES	-	-	-
The maximum number of file entries in the area.			
INT_SIZE_OF_FILES	R/O	-	-
The current total size of the files in the area.			
INT_MAX_SIZE_FILES	-	-	-
The maximum total size of files in the area.			

3.6.20 SET_TIME_ZONE

Describes a specific time zone.

SET_TIME_ZONE	
TS_TIME_ZONE	… “CET”
TS_TO	… “CEST”
TS_DESCRIPTION	… “Central European Time …”
INT_ADD_TO_UTC	… offset from UTC in seconds
TIME_ACTION	… for time conversion functionality, not persistent
TS_PREFIX	… for mapping functionality, not persistent
L_DAYLIGHT_SAVINGS	
SET_DAYLIGHT_SAVING	
TIME_START	… 2003-03-30 02:00
TIME_END	… 2003-10-26 03:00
SET_DAYLIGHT_SAVING	
TIME_START	… 2004-03-28 02:00
TIME_END	… 2004-10-31 03:00
…	
SET_TIME_ZONE	… TCOSS time zone definition

The SET_TIME_ZONE may contain a nested SET_TIME_ZONE child object which describes the conversion between TCOSS time and UTC (update of existing TCOSS system).

The TIME_ACTION child object is not part of the definition. It may be used to convert time stamps according to this time zone.

Example of local time -> UTC conversion:

ohh_c_time_put(h_set_time_zone, TIME_ACTION, &timevalue, TIME_INT);

ohh_c_time_get(h_set_time_zone, TIME_ACTION, &timevalue, TIME_INT | TIME_UTC);

3.6.21 SET_VOLUME

Concepts:

Data on the Archive Server is organized into CD-sized archive volumes. Archive Volumes can be written to CD, or restored to the archive. Each Archive Volume has a unique volume number. Volume numbering starts with system installation.

Child objects:	**attributes**	**default**	**value-restr.**
INT_VOLUME	R/O	-	-
The volume number: 1,2,3,...			
INT_STATE_ARCHIVE	-	-	-
The state of the volume. It contains the following flags (additional flags may be defined):			
STA_OPEN	volume open, new messages may be added		
STA_ON_DISK	volume is on-line (stored on the archive server's hard disk)		
STA_ENTRIES_ON_DISK	only the entries are on-line, messages are off-line		
STA_IDXVOL_ON_DISK	the index volume covering this volume is on-line		
STA_INDEX_VOLUME	volume is an index volume		
STA_FILE_ERROR	some or all files of an on-line volume could not be opened		
INT_VALID_PGNO	R/O	-	-
Only in Index volumes: The number of the first volume covered			
INT_VALID_PGCNT	R/O	-	-
Only in Index volumes: The number of volumes covered			
TS_SECTION	R/O	-	-
The on-line path. It shows the file path where the volume subdirectory is located on the archive server.			
TS_PATH	-	-	-
The path to an off-line volume on CD, e.g. Z:\AV000075CD1 (CD of volume 75 is in a local jukebox)			
TIME_CREATED	R/O	-	-
The start date and time of the volume, i.e. the date / time of its oldest entry			
TIME_ACTION	R/O	-	-
The end date and time of the volume, i.e. the date / time of its most recent entry			
INT_N_OF_ENTRIES	R/O	-	-
The number of entries stored in this volume.			
INT_N_OF_FILE_ENTRIES	R/O	-	-
The number of documents stored in this volume.			
INT_N_OF_CDS	R/O	-	-
The total number of CDs of this volume which were written and verified.			
INT_ACTIONS	W/O	-	-
The administrator actions to be performed on this volume, may contain one of the following flags:			
ACTION_CD_WRITE_IMMEDIATE	start automatic writing of CD immediately		
ACTION_CD_WRITE_SCHEDULE	start automatic writing of CD at scheduled time (at night)		
ACTION_CD_PREPARE	for manual CD writing: create encrypted volume copy		
ACTION_CD_VERIFY	for manual CD writing: verify CD		
ACTION_CD_MAKE	combined with ACTION_CD_VERIFY: increment CD count		
ACTION_CD_CLEANUP	for manual CD writing: delete encrypted volume copy		
ACTION_VOL_RESTORE	restore volume from CD		

ACTION_VOL_RESTORE_ENTRIES	restore only index and entries from CD		
INT_LAST_USER_ACTION	R/O	-	-
The last action set in INT_ACTIONS (INT_ACTIONS is write-only).			
INT_ERROR	R/O	-	INACTIVE / ACTIVE / OK / ERR_...
The result of the last action.			

3.6.22 SET_FOLDER

Concepts:

A folder represents a view on the contents of a permanent store. The folder holds a view-filter and a (filtered) display list of entries.

The selected view to the store can give a very long list of entries. The display list, on the other hand, has a limited length to save client, server and communication resources.
The application may request a display list, specifying the filter, the starting or end point and the maximum size of the list.

When the SET_FOLDER_DISPLAY is opened it represents the current state of the store. (When you do not need the display list anymore - just forget() it.). Often, for scrolling, you want to get a new display list that starts just after the last one. To request this you copy the last entry displayed to the SET_START_ENTRY and open the SET_FOLDER_DISPLAY again. The new display list will start with the next entry.

Note:
The folder entries are used for display and selection only. They have the same type as the content entries (see SET_PS_CONT) but contain only child objects of type INT_, TIME_ or TS_ that are direct child objects of the content entry.

Child objects:	**attributes**	**default**	**value restr.**
SET_FOLDER_DISPLAY	R/O	-	-
The display list plus the 'end reached' flag. If requested, a mail system folder display may also contain the number of entries in this view.			
SET_DISP_FILTER	-	-	-
The filter to select a view on the store. The type of the filter (the exact values that the filter holds) depends on the type of the store.			
SET_START_ENTRY	-	-	-
Sets the start / end position for the display list			
INT_IND_X_POS	-	-	0 – 1 000 000
Sets the start position in the absence of a start entry. Support depends on type of folder.			
INT_MAXDISP_DIR	-	0	-
Maximum size and scrolling direction for display list.			
INT_SORT_BY	-	-	-
The type constant of the child element used for sorting the display list. It is ignored if the permanent store does not support sorting on this element.			
INT_FOLDER_TYPE	-	IN_FLDR	IN_FLDR, OUT_FLDR, IN_ARCH, IN_MIXED, CHANGES, ADDRESS_MAP_FLDR
The values INT_FLDR .. IN_MIXED are used for mail and mail archive folders:			
Mail: IN_FLDR / OUT_FLDR determine the priority of listing according to recipient / originator.			
Archive:	OUT_FLDR selects the sending archive. May be combined with the flag OUT_FLDR_ALL_ATTEMPTS to see all send attempts. OUT_FLDR_LOG_ENTRIES selects all log entries.		
	IN_FLDR, IN_ARCH or IN_MIXED select the reception archive		
	The reception archive may combine archive and mail entries. IN_FLDR: Only entries from archive without corresponding mail entries IN_ARCH: All archive entries		

IN_MIXED: Archive entries replaced by the corresponding mail entry if possible

In case of folder type IN_MIXED the resulting folder display list may contain entries of types

SET_ENTRY_MS_MAIL and SET_ENTRY_MS_MAIL_ARC. To access the message content, it has to be opened from the appropriate permanent store (SET_PS_CONT_MAIL_SYS or SET_PS_CONT_MAIL_ARC.

The values IN_FLDR, CHANGES and ADDRESS_MAP_FLDR are used for recipient store folders:

IN_FLDR: normal address book
CHANGES: New, changed and deleted entries for address book synchronization
ADDRESS_MAP_FLDR: Address mapping

INT_COUNT_ENTRIES - - YES / NO

The application sets this object to YES, if the server should determine the number of entries according to the filter values set by the application. Since this function is very costly in terms of CPU load it must be explicitly enabled.

The server returns INT_N_OF_ENTRIES in SET_FOLDER_DISPLAY if INT_COUNT_ENTRIES is set to YES. It contains the maximum number of entries available for the given filter.

This functionality is available only for the mail store. There are certain restrictions on supported filter values for this functionality:

1. Valid only in SET_FOLDER_MAIL_SYS
2. A single recipient must be specified without wildcards
3. Document Classes are filtered without restrictions
4. The State of the document must be NEW or TOUCHED
5. No other filter values must be used!

Setting the view:

To set the filter you create an object of type SET_FILTER_type_of_store and fill it with the filter values. Then you store the filter object to SET_DISP_FILTER.

Requesting a display list:

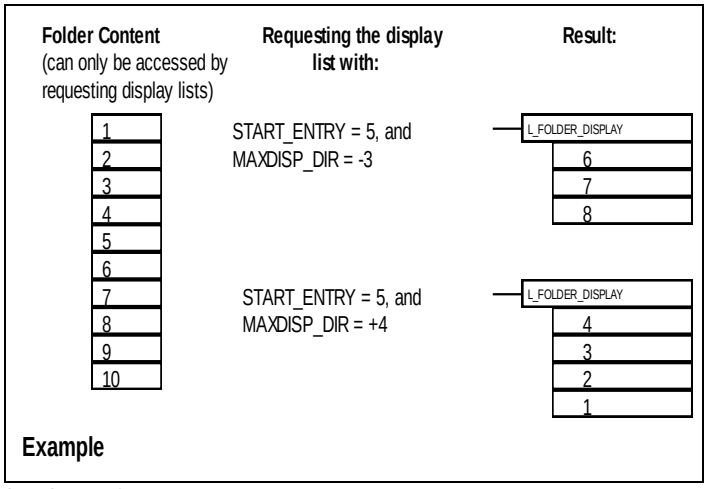
Select start/end point:

To set the start of the display list to the beginning of the selected view delete the SET_START_ENTRY object.

To set the start/end of the list to any other point in the view you need a SET_ENTRY_type_of_stored object. You store this to SET_START_ENTRY. The next display list requested will start just after / before this entry. (During scrolling you will use the last displayed entry to set the start point for the new display list.)

Select size and direction:

Set INT_MAXDISP_DIR to the required value.



n > 0: The display list will have maximum n entries, starting just **after** SET_START_ENTRY.

n < 0: The display list will have maximum n entries, starting just **before**.

0: The list will hold only the freshly updated SET_START_ENTRY, when SET_START_ENTRY does not exist anymore in the view the display list is empty.

When the view is empty, the list is always empty.

When SET_START_ENTRY does not exist |n| specifies the maximum number of entries in the display list.

Opening the list:

Each time you open the SET_FOLDER_DISPLAY you get a current display list from permanent store. You now have access to the display list and the 'end reached' flag.

3.6.22.2 SET_ FOLDER _MS

Users with standard USER rights have access only to user's message stores. (Entries in folders starting with '+' will not show up - regardless of the filter values set.)

Users with at least ADMIN rights have access to all TOS folders (user and system folders).

3.6.22.3 SET_FOLDER_MAIL_SYS

The folder content exists only at the server. It cannot be directly opened – it can only be accessed by requesting a display list.

Order of entries in the folder content:

(Sort priority from left to right)

IN - Folder

Recipient	Message Priority	Time of reception	last touched
AAA	low	now	now
ZZZ	high	earlier	earlier

Out - Folder:

Originator	last touched
AAA	now
ZZZ	earlier

The behavior of the filter values INT_DOC_CLASS and INT_DOC_CLASS_U in the mail store is different compared to the standard behavior. These filter objects are not considered as lower bound and upper bound of a range of allowed integer values, but as lower and upper bounds for every single bit in a bit field.

Example:

INT_DOC_CLASS = 0x00000100

INT_DOC_CLASS_U = 0x000001FF

Means, bits 0 to 7 can be both 0 and 1, bit 8 has to be 1, and all other bits have to be 0, i.e., allowed are values with a bit pattern of: 0x000001XX

3.6.22.4 SET_FOLDER_MAIL_ARC

The folder content exists only at the server. It cannot be directly opened – it can only be accessed by requesting a display list.

Order of entries in the folder content:

Archive folder:

Time of archiving
now
earlier

3.6.22.5 SET_FOLDER_ARCHIVE

The Archive server's search folder contains some additional child objects:

Additional child objects:	attributes	default	value-restr.
INT_OPTIONS	-	-	OPTION_IMMEDIATE

The search folder options. Currently only one flag is defined: OPTION_IMMEDIATE. If it is set, the ohh_c_open() function of SET_FOLDER_DISPLAY returns immediately (not waiting for INT_TIMEOUT to expire), if at least one entry is found. All readily available entries are returned immediately even if there are less entries than the requested number.

INT_TIMEOUT	-	3	-

The timeout for returning a search result in seconds. After this timeout a search result list is returned, even if it is empty or shorter than requested.

3.6.22.6 SET_FOLDER_ARCVOL

The Archive server's volume folder gives a view of the archive volumes. The volumes are sorted by volume number. The volume list may be filtered using INT_STATE_ARCHIVE and INT_STATE_ARCHIVE_U to obtain e.g. a list of all index volumes.

3.6.22.7 SET_FOLDER_TIME_ZONE

The time zone folder in the mail system allows to list all time zone definitions.

Get Time Zone by Name

The time zone folder may be used to get the complete time zone definition from a time zone name, e.g. to resolve the time zone name in the user profile.

This is done opening a SET_FOLDER_DISPLAY in the time zone folder after setting the following parameters:

INT_MAXDISP_DIR = 0 (get start entry)

SET_START_ENTRY SET_TIME_ZONE containing only the TS_TIME_ZONE string

If the TS_TIME_ZONE string does not refer to an existing time zone the system default time zone is returned. If there is no system default time zone the returned display list is empty.

Time Zone Mapping Functionality

A special function of the time zone folder allows to access the mapping functionality provided by the MAP2ZONE section in A:rr99. This feature may be used for the synchronization of user profiles, determining the user's time zone from some other information like a phone area code which can be mapped to a time zone.

The mapping functionality is accessed by opening a SET_FOLDER_DISPLAY in the time zone folder after setting the following parameters:

INT_MAXDISP_DIR = 0 (get start entry)

SET_START_ENTRY SET_TIME_ZONE containing only the TS_PREFIX string

The time zone is identified by mapping the TS_PREFIX string to a time zone according to the A:rr99 MAP2ZONE section and returned as a single SET_TIME_ZONE child of the folder display list. If mapping of the TS_PREFIX string to a time zone fails the system default time zone is returned, if there is no system default time zone the returned display list is empty.

Example:

A:rr99
**MAP2ZONE
201,EST
202,EST
203,EST
204,CST
205,CST
206,PST
207,EST
208,MST
…

The MAP2ZONE section is used in this example to map US phone area codes to time zones. The phone area code, e.g. "204" can be filled into the TS_PREFIX child object of the start entry, and after opening the folder display it contains a list with a single SET_TIME_ZONE object holding the time zone definition, the TS_TIME_ZONE field contains the corresponding name, e.g. "CST".

3.6.22.8 SET_FOLDER_RS

There are 3 types of folders in the recipient store (selected by INT_FOLDER_TYPE). The normal address book view is selected with INT_FOLDER_TYPE = IN_FLDR or INT_FOLDER_TYPE not existing.

Address Book Synchronization

A recipient store folder with INT_FOLDER_TYPE = CHANGES returns all new, changed or deleted entries and allows keeping an up-to-date copy of the address book in some outside data base.

The INT_CHANGE child is included in all entries and specifies the kind of change made to the entry: CHANGE_NEW, CHANGE_UPDATE or CHANGE_DELETE.

The INT_CHANGE_IDX child is included in all entries of a CHANGES folder and serves as pointer into the TCOSS internal ring buffer. (Normally only the value in SET_LAST_ENTRY will be used.)

Operation:

a) Open a recipient folder list with INT_FOLDER_TYPE = CHANGES, INT_MAXDISP_DIR = -1 and no SET_START_ENTRY to get a starting point for the update checks (save SET_LAST_ENTRY).

b) Load all recipients opening recipient folder lists with INT_FOLDER_TYPE = IN_FLDR.

c) Check periodically for changes in the recipient store by opening a folder list with INT_FOLDER_TYPE = CHANGES, INT_MAXDISP_DIR = 5 (any positive value) and the SET_LAST_ENTRY from the last periodic check (or from step a for the first check) as SET_START_ENTRY. Filtering on TS_SECTION and TS_RECP_ID is possible.

d) If the CHANGES folder list cannot be opened with return code ERR_OBJ_NOT_EXIST (301), discard all recipients and go back to step a (too many changes occurred since the last check or TCOSS was rebooted).

Restrictions:

a) Filtering in a CHANGES folder is only possible for TS_SECTION and TS_RECP_ID.

b) The SET_START_ENTRY in a CHANGES folder must contain INT_CHANGE_IDX, TS_SECTION and TS_RECP_ID.

c) Distribution lists are handled in the same way as recipients. Distribution lists are included in the CHANGES folder without the L_RECIPIENTS child (like in a normal recipient folder).

d) All changes are lost after a TCOSS restart.

Implementation:

TCOSS enters all new, changed or deleted entries into a ring buffer in memory.

The buffer size is set to one tenth of the configured maximum number of recipients. On Transputer-based systems it is set to 3.

Optimization: If a recipient is saved without any change, it will not get a new version ID and it is not entered into the ring buffer of changed entries.

ddress Mapping Folder

The address mapping folder lists all addresses (active and inactive) of all recipients of type USER. The inactive addresses are used for the inbound routing.

Three additional child objects appear in the recipient store entries of an address mapping folder and may be used for filtering (They are also defined as child objects of SET_FILTER_RS).

INT_ACTIVE	active flag (R/O)
TS_SERVICE	service of address (R/O)
TS_FREE_ADDR	free address string without service prefix (R/O)

The string contained in TS_FREE_ADDR is extracted from the address structure using the same procedure that is applied to an address when an envelope is posted, except that the service prefix and the answer back string are excluded.

The address mapping folder is sorted by:

1. service name
2. free address string
3. active flag
4. user (TS_RECP_ID)

An index will be kept in the recipient store to allow fast access to all addresses. The filtering on the active flag and on service and free address string with or without wildcards will be optimized (done in memory without disk access).

If a particular user has two or more identical addresses set, the address will show up only once in the address mapping folder and no warning or error will be generated in this case.

The maximum number of addresses to be stored in the address mapping index will be three times the maximum number of users set in the system configuration. The string space for the addresses will be allocated assuming a mean address string length of 15 (not counting the service name and the terminating hex 00 character).

If the addresses of a new or edited recipient of type USER cannot be stored in the address mapping index or in the address string space, error code ERR_STORE_FULL (=308) is returned.

3.6.22.9 SET_FOLDER_US

There are two types of folders in the user store (selected by INT_FOLDER_TYPE). The normal user view is selected with INT_FOLDER_TYPE = IN_FLDR or INT_FOLDER_TYPE not existing.

User Profile Synchronization

A user store folder with INT_FOLDER_TYPE = CHANGES returns all new, changed or deleted entries and allows keeping an up-to-date copy of the user profiles in some outside data base.

The INT_CHANGE child is included in all entries of a CHANGES folder and specifies the kind of change made to the entry: CHANGE_NEW, CHANGE_UPDATE or CHANGE_DELETE.

The INT_CHANGE_IDX child is included in all entries of a CHANGES folder and serves as pointer into the TCOSS internal ring buffer. (Normally only the value in SET_LAST_ENTRY will be used.)

Operation:

a) Open a user folder list with INT_FOLDER_TYPE = CHANGES, INT_MAXDISP_DIR = -1 and no SET_START_ENTRY to get a starting point for the update checks (save SET_LAST_ENTRY).

b) Load all user profiles opening user folder lists with INT_FOLDER_TYPE = IN_FLDR. (Store TS_FILE_NAME as an additional reference to the user profile)

c) Check periodically for changes in the user store by opening a folder list with INT_FOLDER_TYPE = CHANGES, INT_MAXDISP_DIR = 5 (any positive value) and the SET_LAST_ENTRY from the last periodic check (or from step a for the first check) as SET_START_ENTRY.

d) If the CHANGES folder list cannot be opened with return code ERR_OBJ_NOT_EXIST (301), discard all user entries and go back to step a (too many changes occurred since the last check or TCOSS was rebooted).

Restrictions:

a) Filtering in a CHANGES user folder is not possible.

b) The SET_START_ENTRY in a CHANGES user folder must contain INT_CHANGE_IDX and TS_FILE_NAME.

c) The INT_FILE_ID, INT_FILE_SIZE and TIME_CREATED child objects are set to dummy values.

d) The long user IDs in TS_USER_ID, TS_GROUP and TS_REPRESENTATIVE of a new or changed entry are incorrect (set to the TCOSS internal short user name like „#001000F“) if those users are deleted before the CHANGES folder is read.

e) Entries with CHANGE = CHANGE_DELETE contain dummy values in all child objects except INT_CHANGE, INT_CHANGE_IDX, TS_USER_ID and TS_FILE_NAME. The TS_USER_ID is incorrect for users with more than 8 characters, so TS_FILE_NAME should be used to identify the user which has been deleted.

f) All changes are lost after a TCOSS restart.

g) With a TCOSS restart all user entries are put into the TCOSS internal ring buffer with CHANGE = CHANGE_NEW and INT_CHANGE_IDX is set to a new value.

Implementation:

TCOSS enters all new, changed or deleted entries into a ring buffer in memory. The buffer size is set to one tenth of the configured maximum number of users.

3.6.22.10 SET_FOLDER_DISPLAY

Concepts:

see SET_FOLDER

Child objects:	**attributes**	**default**	**value restr.**
L_FOLDER_DISPLAY	-	-	-

The display list.

INT_END_REACHED		UNKNOWN	YES

If YES the display list has reached the end of the view in the direction specified.

SET_LAST_ENTRY	-	-	-

The last entry examined during filtering at the server. May be copied to SET_START_ENTRY in SET_FOLDER to start filtering for the next folder display from here.

INT_N_OF_UNREAD	R/O	-	-

The number of unread messages in the user's in-box (in case that a single user has been selected without using wildcards), only in mail folders (SET_FOLDER_MAIL_SYS and SET_FOLDER_MAIL_ARC). Does not exist in all other cases.

INT_QUEUELEN	R/O	-	-

The total number of messages in the user's in-box (in case that a single user has been selected without using wildcards), only in mail folders (SET_FOLDER_MAIL_SYS and SET_FOLDER_MAIL_ARC). Does not exist in all other cases.

SET_VOLUME	-	-	-

Only in an archive search result. If a SET_VOLUME is contained in the folder display it indicates that the search stopped because an offline volume was unavailable. SET_VOLUME gives the number of the archive volume which is needed to continue the search. The path to that volume is specified by setting the TS_PATH child and closing SET_VOLUME. (The path may also be set using the volume folder).

INT_IND_X_POS	R/O	-	0 – 1 000 000

The start position, normalized to numbers between 0 and 1 000 000. The position is indicated relative to the search direction. If the complete list is returned in one call, the start position is zero.

INT_IND_Y_POS	R/O	-	0 – 1 000 000

The end position, normalized to numbers between 0 and 1 000 000. The position is indicated relative to the search direction. If the complete list is returned in one call, the end position is 1 000 000.

INT_N_OF_ENTRIES	R/O	-	-

The total number of entries in the view (exact or approximate).

INT_ERROR	R/O	-	-

The deviation of the number of entries value (0 = exact value)

3.6.23 SET_ENTRY

Concepts:

A data object handled by a permanent store. An entry is not just a reference – it holds all the data.

An entry has a hidden unique stamp and identifies itself.

For each type of permanent store a subclass of SET_ENTRY exists.

3.6.23.2 SET_ENTRY_MS (Message store entry)

The entry of a message store.

Child objects:	**attributes**	**default**	**value-restr.**
TS_TOS_FOLDER	-	-	-

Only used within message store.
In the message store there is a section for each user. The sections are selected by the user ids – so a user id is also a section id.
Sections within a message store (the names of fixed sections start with "+"):

"userid "	for each user on the server a section with "userid" exists it holds all the files of the user's message store
"+FIS"	the Fax Information System section

The following sections may only be accessed by the TCOSS client for system maintenance:

"+MAIL"	client envelopes
"+MAIL5V"	TCOS 5 files (nn99, KKnn, ...)
"+MAIL5I"	invisible files (as used with TTX conversion facility)
"+MAILSYS"	mail entry file, archive entry file, recipient store file, service store file, registration store file
"+USER"	the user-definition files of the user store object, one file per user
"+TECH"	holds program and config files for file names see TCOS internal description.

TS_FILE_NAME	-	-	-

The name of the document.
Within a message store the name must be unique per user.

In mail folders the name is just an inf. field that shows the name of the envelope entered. It has no effect on the function of the mail engine and may even be empty.

TS_REF - - -

A textual reference to the content of the message or the file. This field normally appears in the cover of a message and is the main criterion for selection of entries from the folder. If TS_REF in the envelope header is filled, TS_REF in the mail entry is a limited length (max. 32 char.) copy of it.

INT_NPAG - - -

The number of pages in the document.
Can be supplied by the application. Is not set automatically by the object handler. When the application changes the envelope contents (L_ENV_CONT) it shall either delete this object or set it to the correct value again. When UN_CONTENT is not of type L_ENV_CONT this field has no meaning and should be deleted.

INT_DOC_CLASS – – RESTR || TEXT_DOC || XIMAGE || BINARY_DOC || XVOICE || DC_MUST_SIGN

The class of the document. Is set automatically by TCOSS whenever an envelope is stored in the message store or posted in the mail system.

INT_FILE_SIZE R/O

The size of the file or the size of the envelope content.

TIME_CREATED -

When this child does not exist when an entry is created in a message store or put to the mail system it is created automatically with the current time. If a message is changed and saved, and the TIME_CREATED child is not explicitly set or deleted, it will retain its original value. In case of messages received from public lines TIME_CREATED gives the reception start time.

TIME_REC_END -

Date / time when reception ended. Only exists in documents received on public line channels and in voice mails.

UN_CONTENT R/O when opened from the mail system

May be of any object type for message stores.
Must be of type L_ENV_CONT to be accepted by the mail system.

SET_CONTENT_VIEWS R/O

Presents different views of the envelope contents.
Can be opened only if UN_CONTENT is of type L_ENV_CONT. Otherwise the
error: ERR_WRONG_TYPE is returned.

INT_LINELEN not stored permanently LINELEN_UNLIMITED 0

May be set temporarily to act as a parameter for SET_CONTENT_VIEWS. If it is set to LINELEN_UNLIMITED before SET_CONTENT_VIEWS is opened, no line make up will be done for any text block in the view. The page make up may also be incorrect in that case. Other features of the content view are not affected.

INT_USAGE_COUNT R/O

The number of times an entry has been opened from the SET_PS_CONT_MS.
The usage count is set to 0 when an entry is created in a message store.
Not existent for entries in the mail system.

TS_DOCUMENT_ERR	R/O	-	any 2 byte error code (as in TCOS 5.xx)

The 2 byte error code as used in TCOS 5.xx. May be set on reception.

TS_TC_MSG_ID	-	-	-

The TCOSS message ID, refers to a message with all its recipients. Normally it will be automatically generated by TCOSS and contain a 16-digit hexadecimal number. When a message is posted via TCSI the TS_TC_MSG_ID may be specified by the client (this should be done if a message is corrected or forwarded).

TS_REC_CHANNEL	-	-	-

The receiving channel number, 2 characters, e.g. “07” or “B3”. Gives the channel number on the media server if the message was received on a remote channel, otherwise it’s the local TCOSS channel number.

TS_REC_SERV_ID	-	-	-

The ID of the media server if received on a remote channel, otherwise empty.

TS_REC_QUEUE	-	-	-

Holds the link queue name (which is an existing TCOSS user ID) in case of message posting by the TC/Link application. The TS_REC_QUEUE field is filled by TC/Link before a message is posted and will appear as attribute with the name “TS_REC_QUEUE” in the message submit log entry. Although the TS_REC_QUEUE field is stored internally in the message entry it is currently not displayed in a message store or mail folder.

3.6.23.3 SET_ENTRY_MS_MAIL (Mail entry)

The mail entry is derived from the message store entry with the following restrictions:

Within the mail system the usage count is not supported and the content must be of type envelope content.

Child objects:	**attributes**	**default**	**, value-restr.**
INT_STATE			ST_POSTED, ST_WAIT_SEND, ST_WAIT_RETRY, ST_SENDING, ST_BREAK0 .. ST_BREAK5, ST_CANCELLED, ST_END_OF_RETRIES, ST_CORRECTED, ST_SENT, ST_DISTRIBUTED, ST_ROUTED, ST_DELIVERED, ST_UNREAD, ST_READ ST may be combined with flags (ST_TERMINATE etc.)

The state of the mail entry. When the state of a mail entry is updated the mail system may perform certain actions. See mail system.

INT_STATE_MASKED	R/O	INT_STATE without flags (INT_STATE & ST_MASK)

This child, together with INT_STATE_MASKED_U, is used for filtering on a range of state values in a mail folder.

INT_STATUS	NEW_ENTRY	NEW_ENTRY, TOUCHED, AT_NEXT_NODE, SENDING_ACTIVE, POS_TERM, NEG_TERM, DO_CANCEL, DO_ACTIVATE, DO_POS_TERM_NO_NOTIF

The (old) status of the mail entry, supported for compatibility with previous releases. See mail system.

INT_DEL_TYPE	R/O	TO_	TO_ / CC_ / BCC / AUTHORIZE

The delivery type determines how the recipient is shown in the cover.

TS_RECIPIENT	R/O	-	-

Holds the user id of the recipient.

TS_RECIPIENT_INFO	R/O	-	-

Holds a shortened string representation of the recipient:
Fullname (if available), Service, number (if no recipient shortcode available)

TS_REC_INFO_ORIG	R/O	-	-

Holds the originally specified recipient information (see TS_RECIPIENT_INFO).
This value is passed unchanged to send orders generated by actions defined in a user profile.

TS_RECIPIENT_FULLNAME	R/O	-	-

Holds the full name of the recipient.

TS_RECIPIENT_SERVICE	R/O	-	-

Holds the service of the first active recipient address.

TS_ORIGINATOR	R/O	-	-

Holds the user id of the originator.

TS_NORMALIZED_ORIG	R/O	-	-

Holds the normalized string representation of the originator, including
personification. Alternative numbers, if present, are delimited with backslashes.

TS_ORIGINATOR_INFO	R/O	-	-

Holds a shortened string representation of the originator:
shortcode, fullname, list of services (if available from envelope)

TS_ORIGINATOR_SERVICE	R/O	-	-

Holds the service of the first originator address.

TS_ENV_NAME_POSTED	R/O	-

The name of the envelope that was put into the mail system.

TIME_ACTION		-

The time of the next or last action. Set first by the mail system to the earliest delivery time. May be changed by the client to a later time for further send attempts. When the entry is terminated this field holds the time of termination.

TIME_LATEST		-	-

The latest delivery time. If it has the value FUTURE (0x7FFFFFFF) the latest delivery time is infinite.

TIME_INTENDED		-

The time when the message should have been sent, which may be the sending time specified by the user or (by default) the time of posting. If the user specified a sending time which had already passed at the time the message was posted, TIME_INTENDED gives the time of posting.

TIME_SCHEDULED	-	-	

The planned time of the first or next send attempt.

TIME_SELECTED	-	-	

The time when the message was actually selected for sending.

TS_DOC_NR	-	-	

The document and page number as in TCOS 5.

INT_LAST_USER_ACTION	-	NONE	NONE, MSG_READ, SAVED, PRINTED, DISTRIBUTED, REPLIED, FORWARDED,

			CORRECTED, CANCELLED

The last action done by any user client.

INT_PRIORITY	R/O	NORM	LOW, NORM, HIGH, SUPERIOR_1, … SUPERIOR_10

The priority as specified in the header.

TS_LAST_MDA_ACTION	R/O	-	-

The last action done by the message delivery agent.
Currently the TCOS 5 error code.

TS_LAST_MDA_NOTE	R/O	-	-

Any note set by the message delivery agent regarding the last action.
As: answerback received, error codes received on the line etc.

INT_OPEN_RETRIES		-	-

The open retries left. Set initially to 9 when the message is posted. Normally counted down by MDA's. Can be set to any value by the user client to reactivate sending.

INT_SUSP_DUPL	R/O	NO	NO, YES

Suspected duplication flag. May be also set by the client. Is normally printed in the cover page.

TS_COST_CENTER	R/O	-	-

The cost center.

TS_COST	R/O	-	-

The cost of sending in unspecified units.

INT_DURATION	R/O	0	-

The duration of sending in seconds.

TS_CHANNEL	R/O	-	-

Number of TCOSS channel actually used for sending, 2 characters, e.g. “07” or “B3”

TS_SERV_ID	R/O	-	-

ID of media server if sent on a remote channel, otherwise an empty string.

TS_LOCALIZED_ADDR	R/O	-	-

The localized address. This is the address that can be dialed at the local node.

TS_NORMALIZED_ADDR	R/O	-

The normalized address. This is the address with international prefix.

TS_CORREL_1, ... _4	R/O	-

The custom fields 1..4. The total length of all four fields is limited to 320 characters.

SET_TIME_ZONE	R/O	-	-

The time zone used for resolving mask and cover variables.
The message’s time zone is determined when the message is posted using (with decreasing priority):

a) the recipient’s user profile
b) the normalized recipient number zone mapping (MAP2ZONE section in A:rr99)
c) the originator’s user profile
d) the normalized originator number zone mapping (MAP2ZONE section in A:rr99)
e) the system default time zone

Envelope references:

The following two fields give a reference to the full address in the header of the envelope.
When the header does not exist (as can happen with envelopes entered via UAS) they are not existent.

INT_ER_RECIPIENT R/O -

This entry points to a recipient in the header. The first recipient has position 0.

INT_ER_ALT_ADDR_LEFT R/O -

Gives the number of alternative addresses left.
This field is changed by the mail system when a client sets the INT_STATE to ST_END_OF_RETRIES (or INT_STATUS to NEG_TERM). It then switches to the next alternative address.

INT_TERMINATION R/O - flags as given below

ARC_POS	create archive entry on positive termination
ARC_NEG	create archive entry on negative termination
NOTIF_POS	create notification entry on positive termination
NOTIF_NEG	create notification entry on negative termination
DEL_ENTRY_POS	delete mail entry on positive termination
DEL_ENTRY_NEG	delete mail entry on negative termination
DEL_ENV_POS	delete envelope on positive termination
DEL_ENV_NEG	delete envelope on negative termination
BR_POS	create backreception entry (message) on positive termination
BR_NEG	create backreception entry (message) on negative termination
REM_ENV_ASAP	remove envelope as soon as possible, do not wait for cyclical erasure
INSTANT	immediate positive termination (on first state change)
TC_REGISTERED	positive termination only after recipient has read the message
GATEWAY	send order remains visible in the sender's out-box until it is terminated by the recipient (like sending to GATEWAY user)

Defines the termination state and the actions to be taken on positive or negative termination.
The termination flags as specified in the header.
Informs the mail system on the required actions on POS_TERM or NEG_TERM.
Cannot be changed by the client.

TS_NODELIST R/O - -

A string of 1 character node ID's. Gives access to TCOS 5.xx routing process.

INT_EVENT_TYPES R/O IN_MAIL IN_MAIL | DEL_NOTIF | NON_DEL_NOTIF | BACK_REC | MWON | MWOFF | FROM_ACTION

The events that trigger the recipient's in actions. Should not be used by client, use MSG_TYPE instead.
FROM_ACTION is set by the server if the entry was generated by any server action (IN-action, etc.)

INT_OPTIONS R/O - HIGH_RES | BACKRECEPTION | HEADERLINE | AUTO_TERMINATION | HANDLE_INVALID_REC

The send options as specified in the header.

INT_MSG_TYPE R/O NORM NORM / ROUTING / NOTIF

The type of the message. Within a routing chain the type at the first node is NORM, at next ones it is ROUTING, for the notification it is NOTIF at all nodes.

SET_ENTRY_MS_BR R/O

The envelope entry of the backreception. May exist only with original send orders. Holds only TOS folder and filename, can be used to open backreception document.

SET_ENTRY_MS_MAIL_ORIG R/O

The original entry of the message. Exists only within a notification mail entry.
It does not exist in a display folder entry and if the original entry is already archived.
Its child objects have the following values:

TS_TOS_FOLDER	not existing	
TS_FILE_NAME	not existing	
TS_REF	of original envelope	limited to 24 characters if the original message has been erased cyclically

These child objects do not exist if the original message has been erased cyclically:

INT_NPAG	of original envelope	
INT_DOC_CLASS	of original envelope	
INT_FILE_SIZE	of original envelope	
TIME_CREATED	of original envelope	
UN_CONTENT	of original envelope	
SET_CONTENT_VIEWS	of original envelope	
INT_USAGE_COUNT	of original envelope	
TS_DOCUMENT_ERR	of original envelope	
INT_STATE	of original mail entry	includes flag ST_TERMINATE
INT_STATUS	of original mail entry	
INT_DEL_TYPE	not existing	
TS_RECIPIENT	of original mail entry	
TS_RECIPIENT_INFO	of original mail entry	
TS_ORIGINATOR	of original mail entry	
TS_ORIGINATOR_INFO	-	
TS_ENV_NAME_POSTED	of original envelope	
TIME_ACTION	of original mail entry	
TS_DOC_NR	of original mail entry	
INT_LAST_USER_ACTION	not existing	
INT_PRIORITY	of original mail entry	
TS_LAST_MDA_ACTION	of original mail entry	
TS_LAST_MDA_NOTE	of original mail entry	
INT_OPEN_RETRIES	not existing	
INT_SUSP_DUPL	not existing	
TS_COST_CENTER	of original mail entry	
TS_COST	of original mail entry	
INT_DURATION	of original mail entry	
TS_LOCALIZED_ADDR	of original mail entry	last 9 characters cut off if result of action (set in user definition)
TS_NORMALIZED_ADDR	not existing	
INT_ER_RECIPIENT	of original mail entry	
INT_ER_ALT_ADDR_LEFT	of original mail entry	
INT_TERMINATION	not existing	
TS_NODELIST	of original mail entry	
INT_EVENT_TYPES	not existing	

INT_OPTIONS	not existing
INT_MSG_TYPE	not existing
SET_ENTRY_MS_BR	not existing
SET_ENTRY_MS_MAIL_ORIG	not existing

INT_CIF_NR	-	-
INT_CIF_ID	-	-
INT_CIF_VID	-	-

Together these values form a unique id for the entry. A mail entry that has these values set may be used as a selector entry for PS_MAIL_CONT.

3.6.23.4 SET_ENTRY_MS_MAIL_ARC (Mail-Archive entry)

The entries of the TCOSS server's short term archive, derived from SET_ENTRY_MS_MAIL.

Currently it matches exactly the mail entry; in the future some restrictions may be defined for the archive entry.

3.6.23.5 SET_ENTRY_ARCHIVE (Archive entry)

The entries of the archive server's long term archive, derived from SET_ENTRY_MS_MAIL.

Additional child objects:	**attributes**	**default**	**value-restr.**
INT_VOLUME	R/O	-	1,2,3,...
The archive volume number.			
INT_ENTRY_NR	R/O	-	1,2,3,...
The number of the entry within an archive volume.			
INT_SEARCH_ID	R/O	-	-
The search ID. It is taken from SET_START_ENTRY in case that an archive search is continued and used to find the existing background search thread (optimization for faster search results).			
INT_ERROR	R/O	-	0 / SEARCH_TOO_COMPLEX
If it is contained with value SEARCH_TOO_COMPLEX in entries of an archive search result, it indicates that these entries may not match all filter criteria because some were too complex to evaluate completely. This may happen when searching with wildcards, e.g. searching for "A*" in the text content.			

3.6.23.6 SET_ENTRY_US (User Store)

This is the user definition. It holds the user's rights, the different actions of the mail system for this user and the user's client setup.

Child objects:	attributes	default	value-restr.
TS_USER_ID	-	-	-

The alpha user id (up to 127 characters).

TS_FILE_NAME	-	-	-

The TCOSS internal file name (internal short user ID), restricted to 8 characters. May be used in a CHANGES folder as a reference to an already deleted user.

TS_PASSWORD	W/O	-	-

Password hash for backup and restore functionality.

It may also be used to set a new clear-text password, but this function is depreciated. It is provided for compatiblity with old clients only. Such clients should be changed to use TS_PASSWORD_NEW to set a new password.

TS_PASSWORD_NEW	W/O	-	-

It should be used to set a new password. The alphanumeric password has to be defined as figures only for users that want to login from phone or fax. In that case the user_id_num and the password together form the pin code. The password must be provided in unencrypted form. This field is never send back to the client.

INT_DAYS_PW_VALID		-	0 ..n, PW_NEVER_EXPIRES, PW_CHANGE_REQU

The number of days the password is still valid (including the current day). The value PW_NEVER_EXPIRES indicates that the password never expires.
The INT_DAYS_PW_VALID child is read-only, except for the value PW_CHANGE_REQU: By setting INT_DAYS_VALID to PW_CHANGE_REQU when a user profile is stored, the user will be forced to change the password with the next logon. The value PW_CHANGE_REQU is write-only, it will be read back as 0 = password expire
INT_ACCOUNT_LOCKED

The counter of bad logon attempts (r/o), combined with flags
ACCOUNT_LOCKED (0x80000000) ... account locked (r/o)
ACCOUNT_DO_UNLOCK (0x20000000) ... unlock account (w/o)
ACCOUNT_DO_LOCK (0x40000000) ... lock account (w/o)

The INT_ACCOUNT_LOCKED child shows whether an account is locked or not. It also gives the current value of the bad logon attempts counter. If it does not exist, the server (or the client's TCSI32.DLL) does not support account locking.

An account is unlocked by setting the INT_ACCOUNT_LOCKED child to ACCOUNT_DO_UNLOCK when the user profile is stored. The account is also unlocked implicitly when the user profile is stored with a changed password.

An account is locked by setting the INT_ACCOUNT_LOCKED child to ACCOUNT_DO_LOCK when the user profile is stored.

Changing the account lock state requires the "write user profiles" right (R_USERPROF_WRITE, R_UPROF_GROUP_WRITE or R_UPROF_OWN_WRITE, depending on relation to user).

INT_AUTO_LOGIN_ENABLE - NO YES, NO

Defines if an auto login is accepted for that user. With auto login the LAN - user id is used for login automatically.

TS_REPRESENTATIVE - - -

Another user that is allowed to access the folders of the user defined here.

TS_GROUP - - -

The group the user belongs to.

TS_LOCATION - - -

The user's location (for the location-based routing feature).

TS_TIME_ZONE - - -

The user's local time zone name.

INT_RIGHTS - - TECH, ADMIN, USER

The general rights setting. More detailed rights may be defined in the client setup.

INT_RIGHTS_RW R/O - -

The READ / WRITE right settings. Bits 0 ..13 (mask 0x3fff) are set automatically by TCOSS according to the R_READ | R_WRITE value of INT_R_FIS, INT_R_MESSAGE, INT_R_SYS, INT_R_USERADD, INT_R_USERPROF, INT_R_GROUPADD in client right settings (SET_WS_CL_RIGHTS).

Possible flags: R_FIS_READ, R_FIS_WRITE, R_MESSAGE_READ, R_MESSAGE_WRITE, R_SYS_READ, R_SYS_WRITE, R_USERADD_READ, R_USERADD_WRITE, R_SYSADD_READ, R_SYSADD_WRITE, R_USERPROF_READ, R_USERPROF_WRITE, R_GROUPADD_READ, R_GROUPADD_WRITE.

The R_FIS_READ flag corresponds to the R_READ flag in INT_R_FIS, the R_FIS_WRITE flag to the R_WRITE flag in INT_R_FIS, etc..

These flags in INT_RIGHTS_RW have to be changed directly.

flag	**value**	**access right granted**	**checked by**
R_USERADD_GROUP_READ	0x40000	read the private address book of other users in same group	server
R_USERADD_GROUP_WRITE	0x80000	write to the private address book of other users in same group	server
R_USERADD_ALL_READ	0x100000	read the private address book of all users on TCOSS instance	server
R_USERADD_ALL_WRITE	0x200000	write to the private address book of all users on TCOSS instance	server
R_GROUPMSG_READ	0x400000	read message folder of group user	server
R_GROUPMSG_WRITE	0x800000	write message Folder of group user	server

A users access to the own address book is controlled via the R_USERADD_READ, R_USERADD_WRITE flags. If users have the R_USERPROF_READ right, they are also permitted to read all users private address books. If users have the R_USERPROF_WRITE right, they are also permitted to write to all users private address books.

INT_RIGHTS_YN - - -

The YES / NO right settings. The individual flags are set automatically by TCOSS if the related INT_R_...-child object is contained in SET_WS_CL_RIGHTS.

Possible flags: R_SERVER, R_SERVICES, R_OPERATOR, R_REGLIC, R_EXTVIEW, R_DIRECTNO, R_CHGCOST, R_INALL, R_OUTALL, R_MSGALL, R_FIRST_PAGE, R_PRINT, R_SAVEAS, R_TERMINATE,

R_AUTOTERMINATION, R_OVERVIEW_IN, R_OVERVIEW_OUT, R_OVERVIEW_MSG, R_OVERVIEW_IN_GROUP, R_OVERVIEW_OUT_GROUP, R_OVERVIEW_MSG_GROUP, R_OPEN_IN_GROUP, R_OPEN_OUT_GROUP, R_OPEN_MSG_GROUP, R_AUTHORIZE, R_CHG_HEADER, R_CHG_CONTENT, R_AUTH_W_SIGN, R_CORRECT_RECP, R_MSG_KEY, R_SERVICE_RESTR

The R_SERVER flag is set if INT_R_SERVER in SET_WS_CL_RIGHTS has the value „YES“, if it has any other value the flag is reset, if INT_R_SERVER does not exist the flag is not changed by TCOSS. The other flags are handled accordingly.

INT_RIGHTS_YN_2 - - -

The extended YES / NO right settings. Possible flags: R_CORR_GROUP, R_CORR_ALL, R_IS_ROLE, R_DISALLOW_MARKCOMPLETE, R_DENY_WRITE_COVERSHEET

INT_RIGHTS_USERPROF - - -

The extended rights to the user profiles.

Possible flags: R_INACTIONS_OWN, R_INACTIONS_GROUP, R_INACTIONS_ALL, R_ADDRESS_OWN, R_ADDRESS_GROUP, R_ADDRESS_ALL, R_CHANGE_PW_OWN, R_CHANGE_PW_GROUP, R_CHANGE_PW_ALL, R_PW_NEVER_EXPIRES …

The following flags have been defined to control write access to individual fields of the user profile. These flags are active for users who have no general right to write a given user profile. For each field there are three flags which apply to a user's own profile, users in the same group or all users respectively.

flag	**value**	**write access to user profile field**	**checked by**
R_TIMEZONE_OWN R_TIMEZONE_GROUP R_TIMEZONE_ALL	0x08 0x10 0x20	TS_TIME_ZONE	server
R_GROUP_OWN R_GROUP_GROUP R_GROUP_ALL	0x10000 0x20000 0x40000	TS_GROUP	server
R_REPRES_OWN R_REPRES_GROUP R_REPRES_ALL	0x80000 0x100000 0x200000	TS_REPRESENTATIVE	server
R_COSTC_OWN R_COSTC_GROUP R_COSTC_ALL	0x400000 0x800000 0x1000000	TS_COST_CENTER	server
R_LANGUAGE_OWN R_LANGUAGE_GROUP R_LANGUAGE_ALL	0x2000000 0x4000000 0x8000000	TS_LANGUAGE	server
R_US_CONTENT_OWN R_US_CONTENT_GROUP R_US_CONTENT_ALL	0x10000000 0x20000000 0x40000000	SET_US_CONTENT except SET_WS_CL_RIGHTS	server

The R_US_CONTENT_OWN right is granted automatically to all users for compatibility reasons.

INT_RIGHTS_APPL - - -

The extended application right settings.

R_REQU_REPORT	the right to request a report, checked by TC/Report
R_POST_JOB	the right to post jobs, checked by client application
R_META_MAIL	the right to use the meta mail feature
R_ROUTING_DB_TCWEBADMIN	customer or helpdesk administrator right

For the All-For-One key system support 3 flags have been defined:

R_SPLIT	split messages in distributor mode
R_VIEW_KEYS	view keys for key system in header
R_EDIT_KEYS	edit keys for key system in header

The following flags have been defined to control read access to individual fields of the user profile:

flag	value	read access to field	checked by
R_ADDRESS_OWN_READ R_ADDRESS_GROUP_READ R_ADDRESS_ALL_READ	0x2000 0x4000 0x8000	SET_ENTRY_RS with all child objects	client
R_GROUP_OWN_READ R_GROUP_GROUP_READ R_GROUP_ALL_READ	0x10000 0x20000 0x40000	TS_GROUP	client
R_REPRES_OWN_READ R_REPRES_GROUP_READ R_REPRES_ALL_READ	0x80000 0x100000 0x200000	TS_REPRESENTATIVE	client
R_COSTC_OWN_READ R_COSTC_GROUP_READ R_COSTC_ALL_READ	0x400000 0x800000 0x1000000	TS_COST_CENTER	client
R_LANGUAGE_OWN_READ R_LANGUAGE_GROUP_READ R_LANGUAGE_ALL_READ	0x2000000 0x4000000 0x8000000	TS_LANGUAGE	client
R_TIMEZONE_OWN_READ R_TIMEZONE_GROUP_READ R_TIMEZONE_ALL_READ	0x10000000 0x20000000 0x40000000	TS_TIME_ZONE	client

TS_FIS_PREFIX - -

The FIS folder prefix.

TS_LANGUAGE - -

The user's language as 2 character country code:
‘01’ English
‘02’ German
‘03’ French

INT_ACTIONS - - GATEWAY | NO_INC_ARC | AUTO_REJECT | MWON | MWOFF | IN_MAIL | DEL_NOTIF | NON_DEL_NOTIF | BACK_REC | ARC_ALL_ATTEMPTS | NUMBER_LOCKING

Flag GATEWAY: defines a gateway user if set. Gateway users work with in-box state values of public line channels (public line channels are not defined as users, but they could be defined as gateway users). New in-box entries get state ST_WAIT_SEND (instead of ST_UNREAD for a normal user). No auto-termination of notifications is done for gateway users.

Flag NO_IN_ARC: no reception archive if set

Flag AUTO_REJECT: a user which has this flag set will automatically reject all messages in its in-box. The automatic rejection affects three fields of the mail entry: INT_STATE is set to ST_REJECTED, the 2-byte error code TS_LAST_MDA_ACTION is set to “SI” and the TS_LAST_MDA_NOTE string is set to “invalid recipient”. The TCOSS user with ID “+INVALID” gets the AUTO_REJECT flag automatically.

Flag ARC_ALL_ATTEMPTS: Logging of all send attempts in the short term archive. This means e.g. for a fax line that all cases where the distant subscriber was busy or not reachable for some other reason are documented in the short term archive.

Flag NUMBER_LOCKING: Activate the number locking by TCOSS within the open-for-sending and open-for-transfer functions

All other flags signal the existence of the corresponding object in the content. These flags have to be set by the application. They are used by the mail system for optimization.

INT_CHANGE	-	-	CHANGE_NEW | CHANGE_UPDATE CHANGE_DELETE

Only in a user folder with INT_FOLDER_TYPE = CHANGES. Reports the kind of change.

INT_CHANGE _IDX	-	-	-

Only in a user folder with INT_FOLDER_TYPE = CHANGES. (TCOSS internal reference)

UN_CONTENT	-	-	-

The content of the user entry.

INT_N_OF_UNREAD	R/O	-	-

The number of unread messages in the user's in-box at the time the folder display has been opened (if the entry is an element of a folder display list) or at the time the entry has been opened or saved.

INT_QUEUELEN	R/O	-	-

The total number of messages in the user's in-box at the time the folder display has been opened (if the entry is an element of a folder display list) or at the time the entry has been opened or saved.

TS_FULLNAME	R/O	-	-

The full name from the user’s recipient store entry. Only in user store folder.

TS_DEPTM	R/O	-	-

The department from the user’s recipient store entry. Only in user store folder.

TS_SERVICE	R/O	-	-

The service from the user’s recipient store entry. Only in user store folder filtered on a specific service.

TS_FREE_ADDR	R/O	-	-

The DID number from the user’s recipient store entry. Only in user store folder filtered on a specific service.

3.6.23.7 SET_US_CONTENT

The content of the user store entry.

hild objects: **ttributes** **efault** **alue-restr.**

ET_ENTRY_RS_MWON

ET_ENTRY_RS_MWOFF

he full addresses that create mail entries for message waiting on and off if the state of the user que ue changes.

_ACTIONS_IN_MAIL

_ACTIONS_DEL_NOTIFIC

_ACTIONS_NON_DEL_NOTIFIC

_ACTIONS_BACKR

_ACTIONS_JOB_START

_ACTIONS_JOB_END

_ACTIONS_IN_RELEASE

_ACTIONS_OUT_RELEASE

These are lists of recipient store entries. For each event they define the appropriate action(s) to be taken.
Note: Actions are not triggered by messages which have flag AUTO_TERMINATION in INT_OPTIONS set.

_ACTIONS_ALERT

Holds the alerts defined for the user queue.

ET_CLIENT_SETUP

Holds the setup for the different types of clients the user may have.

3.6.23.8 SET_ENTRY_RS (Recipient Store)

Holds the definition for one recipient. A receiver has personal data and a list of alternative addresses for different mail services.
When the recipient store entry of a user is deleted, the system also deletes the corresponding user store entry and the user's private message and recipient stores. Depending on the number of entries to be deleted automatically there may be a longer response time.

Child objects:	**attributes**	**default**	**value-restr.**
TS_SECTION	-	"+TECH", "userid"	
The section of the recipient store (may not be changed after creation of the entry).			
TS_RECIP_ID	-	-	
The short code of the recipient. If the recipient exists also as user, this is equal to his user id (may not be changed after creation of the entry).			
TS_COMPANY	-	-	
Holds the company's name.			
TS_DEPTM	-	-	
The department.			
TS_FULLNAME	-	-	
The given name and the surname of the recipient.			
TS_SALUTE	-	-	
The salutation.			
TS_FREETXT	-	-	
One line of free text to characterize the receiver.			
INT_ACTIVE	-	ACTIVE	ACTIVE, INACTIVE, DELIVERED
The active flag for the header. There are two values for inactive recipients (INACTIVE and DELIVERED), which are handled differently by TCfW in case of authorization (INACTIVE recipients are set ACTIVE). Not stored permanently in the recipient store. It is also inserted as a "read-only" child in an address mapping folder and shows the active flag of a particular address.			
TS_SERVICE	R/O	-	
Only in address mapping folder. The service of a particular address.			
TS_FREE_ADDR	R/O	-	
Only in address mapping folder. The free address string without service prefix of a particular address.			
INT_DEL_TYPE	-	TO_	TO_, CC_, BCC, AUTHORIZE
The delivery type determines how the recipient is shown in the cover, and the sending priority (can be set differently for TO_ and CC_). Not existent in the recipient store.			
INT_DEL_TYPE_DEF	-	TO_	TO_, CC_, BCC, AUTHORIZE
The default delivery type (not active in an envelope's header, only for storage in the recipient store)			
INT_PRIORITY	-	NORM	LOW, NORM, HIGH, SUPERIOR_1, … SUPERIOR_10
The sending priority to be used.			
TIME_DATE	-	-	-
The date of deferred sending. The value has only date seconds (time of day = 00:00:00)			
TIME_OF_DAY	-	-	-
The time of day for deferred sending. The value has only time seconds (relative date = 0.0.00). When an envelope is posted the date for each mail entry is set to the current date. This allows to keep template envelopes with preset time of day.			
TIME_LATEST	-	-	-
The latest delivery time, absolute or relative.			

If the time value set in TIME_LATEST is less than 1 year (date between 1.1.1993 and 31.12.1993), it is taken as

a relative value and added to the intended send time. Otherwise it is an absolute time specification. If it is set to the constant FUTURE (0x7FFFFFFF) the latest delivery time is infinite.

TS_COST_CENTER	-	-	-

The cost center.

INT_OPTIONS	-	-	HIGH_RES | BACKRECEPTION | HEADERLINE | AUTO_TERMINATION | HANDLE_INVALID_REC

The send options.

INT_AUTO_TERMINATION	-		MANUAL, AUTO

Automatic termination of in-box entries.

INT_TERMINATION	-	see SET_ENTRY_MS_MAIL	

The actions to be taken on positive or negative termination.

INT_TYPE	-		RECIPIENT / USER

The type of the recipient (may not be changed after creation of the entry).

NT_OWNERTYPE	-		OT_TOPCALL / OT_MSMAIL / OT_CCMAIL / OT_NOTES / OT_HPOM / OT_HOST / OT_TCFI / OT_EXCHANGE / OT_NDS / OT_MQ / OT_SAPSC / OT_SAPAC / OT_INTERNET / OT_SMS / OT_X400 / OT_CUSTOM_1 ...OT_CUSTOM_10

The type of the owner of the recipient (for directory synchronization).

INT_DIRSYNC_ALLOWED	-		NO / YES

Specifies if this recipient may be updated by directory synchronization processes

INT_USAGE_COUNT	R/O	-

The number of distribution lists containing this recipient

SET_DL_PARENT	-	-

Indicates that this recipient was inserted into an envelope's header as member of a distribution list. Points to the parent distribution list. Not existent in the recipient store.

L_FULL_ADDR	-	-

The list of the recipient's alternative numbers. The send attempts start with the first active recipient in the list. When sending to the current address of the recipient is not successful the next alternative address will be used.

TS_CORREL_1 .._6	-	-

The entry specific correlation info.

Note: When the entry correlation fields (CORREL_1 .. 6) are used REM_ENV_ASAP may not be set in any entry of the same message. (Reason: When the original message (and its header) is deleted the entry correlation info is not available anymore.)

3.6.23.9 SET_DL (Recipient Store)

A distribution list is represented by an object of type SET_DL:

Child objects:	**attributes**	**default**	**value-restr.**
TS_SECTION	-	"+TECH", "userid"	
The section of the recipient store (may not be changed after creation of the entry).			
TS_RECIP_ID	-	-	
The short code of the distribution list.(may not be changed after creation of the entry).			
TS_COMPANY	-	-	
Holds the company's name or any other information to be used in a cover sheet.			
TS_DEPTM	-	-	
The department or any other information to be used in a cover sheet.			
TS_FULLNAME	-	-	
The name or any other information to be used in a cover sheet.			
TS_SALUTE	-	-	
The salutation.			
TS_FREETXT	-	-	
One line of free text to characterize the distribution list.			
INT_PRIORITY_TO	-	LOW, NORM, HIGH, …	
The default priority for “to:” recipients			
INT_PRIORITY_CC	-	LOW, NORM, HIGH, …	
The default priority for “cc:” recipients			
INT_STATUS	-	-	
This value may be freely used by the application, e.g. to separate temporary and permanent lists.			
INT_ACTIVE	-		ACTIVE, INACTIVE
The active flag for the header (not evaluated by TCOSS, all SET_DLs are considered inactive). Not existent in the recipient store.			
INT_DEL_TYPE	-	TO_	TO_, CC_, BCC, AUTHORIZE
The delivery type determines how the distribution list is shown in the cover, and the sending priority (can be set differently for TO_ and CC_) Not existent in the recipient store.			
INT_DEL_TYPE_DEF	-	TO_	TO_, CC_, BCC, AUTHORIZE
The default delivery type (not active in an envelope’s header, only for storage in the recipient store)			
INT_OWNERTYPE	-	OT_TOPCALL	OT_TOPCALL / OT_MSMAIL / OT_CCMAIL / OT_NOTES / OT_HPOM / OT_HOST / OT_TCFI / OT_EXCHANGE / OT_NDS / OT_MQ / OT_SAPSC / OT_SAPAC / OT_INTERNET / OT_SMS / OT_X400 / OT_CUSTOM_1 ...OT_CUSTOM_10
The type of the owner of the distribution list (for directory synchronization).			
INT_DIRSYNC_ALLOWED	-	NO	NO / YES
Specifies if this distribution list may be updated by directory synchronization processes			
INT_N_OF_ENTRIES	R/O	-	
The number of child elements (recipients or nested lists)			
INT_USAGE_COUNT	R/O	-	
The number of distribution lists containing this list			
SET_DL_PARENT	-	-	
Indicates that this distribution list was inserted into an envelope’s header as member of a distribution list. Points to the parent distribution list. Not existent in the recipient store.			
L_RECIPIENTS	-	-	
The list of the recipients. May only contain recipients (or other distribution lists) which already exist in the			

recipient store. A recipient cannot be edited within a distribution list; it can only be included in the list. Internally, references (pointers) to the actual recipients are stored in the distribution list, not the recipients themselves.

The recipient store folder (SET_FOLDER_RS) contains recipients and distribution lists sorted only by TS_SECTION and TS_RECP_ID, regardless of the type (SET_ENTRY_RS or SET_DL) of the entries.

While the complete SET_ENTRY_RS is contained in the recipient store folder, this is not true for distribution lists. The L_RECIPIENTS child of SET_DL is not given in the recipient store folder. The distribution list has to be opened from the recipient store content to access all its child objects.

When a distribution list is opened from the recipient store content (using the folder entry as selector), the internal references to all list entries are resolved and the current values inserted into the list. The L_RECIPIENTS may contain recipients and further distribution lists. For recipients the complete SET_ENTRY_RS is inserted into the list, distribution lists are included without their L_RECIPIENTS child.

In case of nested distribution lists only one level will be resolved after an open in the recipient store content. To resolve a second nesting level another open in the recipient store content, using an unresolved SET_DL entry from the previous open as selector, is done.

The selector entry for an open in the recipient store content may be taken from the recipient store folder or from a previous open. It is also possible to create a new selector entry with type SET_DL and fill TS_SECTION and TS_RECP_ID.

When a distribution list is closed or saved in the recipient store, the existence of all recipients (or other distribution lists) in the list is checked. ERR_OBJ_NOT_EXIST will be reported if the check fails. It is sufficient to specify only TS_SECTION and TS_RECP_ID of all (SET_ENTRY_RS or SET_DL) entries in L_RECIPIENTS. Additional child objects may be included but are not evaluated.

If a recipient is deleted from the recipient store, it is also deleted from all distribution lists. So if a recipient is deleted and re-created with the same recipient ID, it will be missing in all distribution lists where it had been included before. (It is seen as a different recipient.) The same applies to distribution lists which are included in distribution lists.

Usage of Distribution Lists

Distribution lists may be included in the recipients list of an envelope header. Envelopes with distribution lists may be stored in a message store (e.g. to serve as templates).

Before an envelope is posted, all distribution lists in its header have to be resolved by the application, which opens the distribution lists from the recipient store content and inserts the recipients into the envelope header's L_RECIPIENTS. The distribution lists of the first nesting level remain in the header to be accessed with cover variables. Their INT_DEL_TYPE has to be set, the L_RECIPIENTS child should be removed. The INT_ACTIVE child is not evaluated by TCOSS (all SET_DLs are considered inactive), it may be used by the application to keep track of the resolving process.

All recipient entries originating from a distribution list are marked in the envelope header by including a SET_DL_PARENT child. The SET_DL_PARENT may be empty (only its existence is checked in the cover variable evaluating procedure) or may point (with TS_SECTION and TS_RECP_ID) to the parent distribution list. It may be used by the application to revert the distribution list resolving. The application also sets the INT_DEL_TYPE child of the recipient entries equal to the INT_DEL_TYPE of the distribution list.

The inclusion of distribution lists originating from a distribution list in the envelope's header is optional. If they are included they have to be marked with a SET_DL_PARENT child and their L_RECIPIENTS child should be removed.

For the message preview, the same procedure as for posting is applied to the envelope's header before the SET_CONTENT_VIEWS is opened.

3.6.23.10 SET_ENTRY_SS (Service Store)

A service entry defines the document classes the service can handle, the internal prefix for sending and the type of the address object.

Child objects:	attributes	default	value-restr.
TS_NAME	-	-	
Holds the short name of the service.			
TS_DESCRIPTION	-	-	
The description of the service + capabilities.			
INT_DOC_CLASS	-	-	
The document class as defined in SET_ENTRY_MM.			
TS_PREFIX	-	-	
The internal prefix for the stringed address.			
INT_ADDRESS_TYPE	-	-	SET_FAX_ADDRESS SET_TX_ADDRESS SET_TTX_ADDRESS, SET_X400_ADDDRESS, SET_TC_ADDRESS, SET_POSTAL_ADDRESS SET_FREE_ADDRESS
The type of the address object for that service.			

3.6.23.11 SET_ENTRY_REGS (Registration Store)

A registration entry is created automatically when a workstation logs in for the first time.

Child objects:	attributes	default	value-restr.
TS_WORKST_DESCR	-	-	
The workstation description as specified at client installation.			
TS_USERID_LAST_LOGIN	-	-	
The id of the user that did the last login on that workstation.			
TS_TIME_LAST_LOGIN	-	-	
The time of the last login on that workstation.			
INT_CLIENT_TYPE	-	CT_TCFW	CT_TCFW / CT_TCFXDPRO / ...
The type of the client.			
INT_LICENSE_TYPE	-	CT_TCFW	CT_TCFW / CT_TCFXDPRO / ...
The type of license used.			
TS_CPU_NO	-	-	
The CPU number of the license used.			
TS_NAME	-	-	

A descriptive string of the license referred to by the registration.

TS_PREFIX	-	-

The default registry section, representing the process instance.

3.6.23.12 SET_ENTRY_LICENSE

License store entries are identified by license type and CPU number.

Child objects:	**attributes**	**default**	**value-restr.**
INT_LICENSE_TYPE	-		CT_TCFW / CT_TCFXDPRO / ...
The type of the license.			
TS_NAME	-	-	
A descriptive string of the license.			
TS_CPU_NO	-	-	
The CPU number of the license.			
INT_MAX_REGISTRATIONS	-	-	
Number of users, workstations, link types or disk size in GB depending on the license type.			
TIME_KEY_VALID	-	-	
The expiry date of the license. May be set to FUTURE for an unlimited license.			
TS_CPU_KEY	-	-	
The license key, 10 digits long.			
INT_N_OF_ENTRIES	R/O	-	
Number of currently used registrations.			

All licenses which use the TCOSS server's CPU number will remain valid for 14 days in case of a CPU number change (e.g. model 165 slave running stand-alone).

The setting of TCfW and FXDPRO licenses in SET_SYSTEM (SET_LICENSE and SET_LICENSE_PRO) will still be supported for compatibility reasons.

The maximum number of registrations of expired or invalid licenses is set to zero, except for the TCfW client license which allows 5 registrations without a valid key.

Licenses of type TCfW and FXDPRO cannot be deleted (they have to exist to be accessible via the SET_SYSTEM objects). Licenses which are in use should not be deleted. If a license is deleted from the license store and then recreated its registration count (INT_N_OF_ENTRIES child) may be incorrect until the next system reboot (already existing registrations are not counted).

License Models

A new TCOSS configuration will start running the old license model. It may be switched to the new license model with TCSI calls as described above, but there is no TCSI function to switch back to the old model.

The only way to switch from the new license model to the old one is to stop TCOSS, delete the system file "+MAILSYS/ALICENSEFILE" with TCDEL.EXE, thereby also deleting all licenses, and to restart TCOSS.

License Classes in Old Licensing Model

Client Licenses

License types: CT_TCFW, CT_TCFXDPRO, ... CT_TCOPEN

User based registration: The license type selects the appropriate license. The number of users working with a particular license is limited by the INT_MAX_REGISTRATIONS value of the license.

TCOSS Foundation Classes (short “TC/TFC”)

License type: CT_TFC

Like for other client licenses, the TC/TFC license count is “per user”, i.e. it sets a maximum number of users allowed to run TFC.

If a TFC license is actually used (by any number of users up to the maximum count value), this will be counted as one TCOSS channel on all Models 2xx. This means that all users logging in with a TFC license type may be unable to do so (error 621 “registration limit reached” reported) if there is no free TCOSS channel on a Model 2xx.

TCOSS Foundation Classes on Model 210 (short “TC/TFC210”)

License type: CT_TFC210

This license works like the regular TC/TFC license, except that it is used on a Model 210. This means e.g. that it is counted as one TCOSS channel if TCF is actually used. The existence of a model 210 license on the TCOSS server switches the mode of operation. A regular TC/TFC license is not supported on a model 210 (It may be entered, but it is not active).

Link Licenses

License types: CT_LINK_AC, ..., CT_LINK_SJ, ..., CT_LINK_WM

Workstation based registration: The license type selects the appropriate license. The number of workstations working with a particular license is limited by the INT_MAX_REGISTRATIONS value of the license.

The workstation is identified by TCOSS using the INT_LAST_SESSION_ID value (similar to the way client registrations were handled by previous releases). The last session ID will be stored by the TCSI32.DLL under a registry sub-key which contains the TCOSS server name and the license type.

TC/Link for Model 210 (short "TC/Link210")

License type: LT_LINK210

This license is only active on TCOSS servers model 210 (although it may be entered on any TCOSS server), and replaces all other link licenses on this model.

The license is entered with the TCOSS server's CPU number.

The license count is "per link type and per workstation". Multiple instances of the same type of link on the same workstation are possible and not counted separately. This license will be issued with a count of 1.This means that one link of any type (and multiple instances of the same link) may be operated on either the TCOSS server or on a 3rd party hardware, but not on both.

The idea of operation is, that on models 210 all link registrations are counted for the "TC/Link for Model 210" license, while on all other models the normal link licenses are used. The existence of a model 210 license on the TCOSS server switches the mode of operation. Normal link licenses (like TC/Link-AC, FI, LN, ...) are not supported on models 210 (They may be entered, but they are not active).

Link Server License

License type: LT_LINK_SERVER

The appropriate license is found with license type LT_LINK_SERVER and matching CPU number.

Link type based registration: The number of links of different types (INT_CLIENT_TYPE value is used) working with a particular license is limited by the INT_MAX_REGISTRATIONS value of the license

Exception: Link-FI, -SJ and -WM (running on a link server) are not registered and are therefore not counted for the registration and channel limit. This exception is implemented in TCOSS.

TC/Link 210 on Model 280 (short "TC/Link210-280")

License type: LT_LINK210_280

This license is only active on TCOSS server models 210 (although it may be entered on any TCOSS server), and replaces all link server licenses on these models. The license is entered with the CPU number of the model 280 link server.

Like for a normal link server license, the license count is "per link type". Multiple instances of the same type of link on the same link server are possible and not counted separately. It will be issued with a count of 1.This means that one link of any type (and multiple instances of the same link) may be operated on the Model 280 link server.

There is no exception for Link-FI, -SJ, -WM like on the regular link server, these links are counted like any other link for the registration and channel limit.

Archive License

License type: CT_ARCH

For TC/ARCH running on TCOSS server. No registration and no extra checks except that license must be valid.

Archive Server License

License type: LT_ARCH_SERVER

For TC/ARCH running on archive server. No registration is done. The archive server license must be valid and the licensed disk size is checked against the disk size reported by the archive server. The disk size of the archive server license refers to the total physical disk space installed on the archive server.

Archive Jukebox License

License type: LT_ARCH_JUKEBOX

For using a jukebox and fully automatic CD writing with TC/Archive. The license count refers to the number of slots in the jukebox.

This license is also issued with a license count of 1 to enable semi-automatic CD writing with the archive server’s built-in CD writer.

Licenses without Login

License types: CT_IMG_PS, ..CT_IMG_OCR

These licenses are not checked at login, the ohh_c_open function in the registry store is used instead.

Workstation based registration: The license type selects the appropriate license. The number of workstations working with a particular license is limited by the INT_MAX_REGISTRATIONS value of the license.

The workstation is identified by name (TS_WORKST_DESCR in SET_ENTRY_REGS). The applications using this type of licenses may read the workstation name from TS_WORKST_DESCR in SET_OBJ_HANDLER.

TCOSS internal Licenses

License types: LT_MOD211 .. LT_TANDEM_DISK

These licenses are checked by TCOSS at startup time. If a license is required (depends on the hardware) but not valid, TCOSS will operate in a restricted mode. Client access (e.g. for input of license) will be possible, some essential functions like sending and receiving will not be available.

Any change in the setting of TCOSS internal licenses becomes active at the next system reboot, except for additional link channels which apply immediately.

The different model licenses limit the number of license channels. The disk size is specified in the INT_MAX_REGISTRATIONS value (in GB).

The license channel value is the sum (a+b+c) of
a) number of TCOSS channels of types fax and telex
b) number of link registrations (link server or specific link, except Link-FI, -SJ, -WM on link server)
c) 1 if a TFC license is actually used

The number of link registrations and the extra channel for the TFC license are checked at login.

Only one of the model licenses (LT_MOD210 ... LT_MOD22Z) exists in the license store at the same time. If a new model license is entered, an existing model license (for the same or a different model) is overwritten. It is also possible to open or delete an existing model license from the license store by specifying any model license type (not necessarily the same model license type) in the selector entry.

The disk size set in the model license 2xx refers to the locally installed logical disk space.

The tandem disk license (type LT_TANDEM_DISK) covers the additional physical disk space used for NT-based fault tolerance (mirror or stripe set with parity). It applies if the installed physical disk space exceeds the logical disk space set in the model 2xx license. The required license size is calculated as difference between physical and logical disk space. Alternatively the disk size set in the model 2xx license may be increased to cover the physical disk size, no tandem disk license is required in this case.

The following values may be used in the fields INT_LICENSE_TYPE, some also as INT_CLIENT_TYPE:

constant	description	count	remarks	value
CT_TCFW	TCfW client	per user		0
CT_TCFXDPRO	FaxDeskPro client	per user		1
CT_TCTRANS	Transfer module	no registration		2
CT_TCLINK	Link void type	no registration		3
CT_TCJAVA	Java client	per user		4
CT_TCOPEN	Open client	per user		5
CT_TFC	TFC client	per user		6
CT_TFC210	TFC client on model 210	per user		7
CT_LINK_AC	APPLI/COM link	per workstation		8
CT_LINK_FI	File interface link	per workstation		9
CT_LINK_LN	Lotus Notes link	per workstation		10
CT_LINK_MX	Exchange link	per workstation		11
CT_LINK_SC	SAPconnect link	per workstation		12
CT_LINK_SM	SMTP link	per workstation		13
CT_LINK_X4_IS	X.400 link with ISOCOR	per workstation		14
CT_LINK_X4	X.400 link w/o ISOCOR	per workstation		15
CT_LINK_GW	Group Wise link	per workstation		16
CT_LINK_SJ	HP Scan Jet link	per workstation		17
CT_LINK_WM	Wireless messaging link	per workstation		18
CT_LINK_MQ	MQ Series link	per workstation		21
CT_LINK_BN	Baan link	per workstation		22
LT_LINK210	Link for model 210	per workstation	license type only	31
CT_IMG_PS	Postscript conversion	per workstation		32
CT_IMG_PCL	PCL5 conversion	per workstation		33
CT_IMG_GIF	GIF conversion	per workstation		34
CT_FILE_REPORTER	File reporter	per workstation		35
CT_DIRSYNC	Directory Synchronization	per workstation		36
CT_IMG_OCR	OCR conversion	per workstation		37
CT_IMG_PDF	PDF conversion	per workstation		38
CT_TC_REPORT	TC/Report	per workstation		39
CT_TC_MWA	TC/MWA	per workstation		40
LT_LINK210_280	Link 210 on model 280	per link type	license type only	57
LT_ARCH_JUKEBOX	archive jukebox support	number of slots	license type only	59
LT_LINK_SERVER	Link server	per link type	license type only	60
LT_ARCH_SERVER	archive server	disk size	license type only	61
CT_ARCH	archive	no disk size check		62
LT_MOD210	model 210 (3 channels)	disk size	license type only	63
LT_MOD211	model 211 (5 channels)	disk size	license type only	64
LT_MOD212	model 212 (8 channels)	disk size	license type only	65
LT_MOD214	model 214 (16 channels)	disk size	license type only	66
LT_MOD215	model 215 (32 channels)	disk size	license type only	67

LT_MOD216	model 216 (60 channels)	disk size	license type only	68
LT_MOD217	model 217 (88 channels)	disk size	license type only	69
LT_MOD218	model 218 (120 channels)	disk size	license type only	70
LT_MOD21Z	spare model type	disk size	license type only	71
LT_MOD221	model 221 (10 channels)	disk size	license type only	72
LT_MOD222	model 222 (16 channels)	disk size	license type only	73
LT_MOD224	model 224 (32 channels)	disk size	license type only	74
LT_MOD225	model 225 (64 channels)	disk size	license type only	75
LT_MOD226	model 226 (120 channels)	disk size	license type only	76
LT_MOD227	model 227 (176 channels)	disk size	license type only	77
LT_MOD22Y	spare model type	disk size	license type only	78
LT_MOD22Z	spare model type	disk size	license type only	79
LT_TANDEM_DISK	tandem disk license	disk size	license type only	80
LT_VOICE_AB	Voice A/B Channels	channels	license type only	81
LT_VOICE_BRI	Voice Basic Rate Channels	channels	license type only	82
LT_VOICE_PRI	Voice Primary Rate Channels	channels	license type only	83

License Classes in New Licensing Model

To enter a license of the new model it is enough to set the TS_CPU_KEY in the license store entry, all other child object values are determined internally by TCOSS.

The values of INT_LICENSE_TYPE in the new model are grouped in different ranges. The value range determines the functionality of the license.

INT_LICENSE_TYPE range	**license count unit**
256 ..511	TCOSS system
512 ..767	User
768 ..1023	Channel
1024 ..1279	Workstation
1280 ..1535	workstation and instance
1536 ..1791	2 instances on same workstation
1792 ..2047	3 instances on same workstation

Overview of licenses of the new model

Per System Licenses

These licenses are issued with a maximum count value of 1. No counting of users, workstations, etc. is done, the existence of the license is sufficient to pass all checks.

Tandem Status Agent
Least cost routing
Email reader voice access

Voice attendant
TFC
SNMP support
Directory Synchronization
Messaging firewall interface
Message wait agent
Meta Mail

Per User Licenses

These licenses restrict the number of users.

TCfW
TC/Player
Client Pro
Client Light
TC/Web
Open Client
Link-MXLink-LN
Link-GW

Per Channel Licenses

These licenses are issued for a specific number of channels (slots in the case of the jukebox license).

Fax channel
Voice channel
Fax & Voice channel
Voice IP port
SMS channel
Telex channel
Archive jukebox (number of slots)

Per Workstation Licenses

These licenses count the number of workstations; any number of instances may run on each workstation.

Archive
TC/Report
LAN print
Break message
File reporter
OCR
Postscript conversion
PCL conversion
GIF conversion
PDF conversion
Link-SM
High volume SMS link
Link-SDD
Link-X4-IS
Link-X4
Link-FI
Link-FI-100 (with additional limitation to 100 messages per day)
Link-MQ
Link-SJ
Voice link

Per Instance Licenses

These licenses count the number of instances, on the same or on different workstations.

Currently no licenses of this group are used, but the functionality is implemented in TCOSS to be available to future professional services developments.

Per Workstation and 3 Instances Licenses

These licenses count 3 instances per workstation. The number of instances on the same workstation is rounded up to multiples of 3 and then divided by 3 to get the count for a specific workstation.

Link-SC
Link-AC

3.6.24 SET_FULL_ADDRESS

A full address consists of the service, the active flag and the public address.

Child objects:	**attributes**	**default**	**value-restr.**
TS_SERVICE	-	-	
Short name of the selected service.			
INT_ACTIVE	-	ACTIVE	ACTIVE, INACTIVE
Determines if the alternative number is active during this sending. Allows to keep the full list of alternative numbers in the envelope for later use. Addresses which are not suitable to receive the contents of an envelope have to be set inactive by the application (e.g. FAX image addressed to telex). The active flag is evaluated only in envelopes.			
UN_PUBLIC_ADDRESS	-	-	
Holds one address. The address may be of type FAX, TX, TTX, X.400, TCOSS internal, free format or postal. Note: For the first implementation the full address can be found directly in the display folder of the recipient store.			

3.6.25 SET_HEADER

Holds all the information needed for sending. When an envelope is put to the mail system the sending information is taken from the header. The header remains part of the envelope even after the message is sent.

Child objects:	**attributes**	**default**	**value-restr.**
TS_REF	-	-	-
The subject of the message. When a message is posted or stored a short form of this field is put to the message entry.			
INT_PRIOR_TO	-	NORM	LOW, NORM, HIGH
The sending priority to be used for all recipients with delivery type TO_ or AUTHORIZE.			
INT_PRIOR_CC	-	NORM	LOW, NORM, HIGH
The sending priority to be used for all recipients with delivery type CC_ or BCC.			
TIME_DATE	.	.	.
The date of deferred sending. The value has only date seconds (time of day = 00:00:00)			
TIME_OF_DAY	-	-	-
The time of day for deferred sending. The value has only time seconds (relative date = 0.0.00). When an envelope is posted the date for each mail entry is set to the current date. This allows to keep template envelopes with preset time of day.			
TIME_LATEST	-	-	default depends on server configuration
The latest delivery time, absolute or relative. If the time value set in TIME_LATEST is less than 1 year (date between 1.1.1993 and 31.12.1993), it is taken as a relative value and added to the intended send time. Otherwise it is an absolute time specification. If it is set to the constant FUTURE (0x7FFFFFFF) the latest delivery time is infinite. The default value is defined by the TCOSS server configuration. The configuration allows different settings for each priority (LOW, NORM and HIGH), the value may be either infinite or a relative time in hours.			
TS_COST_CENTER	-	-	-
The cost center.			
INT_OPTIONS	-	-	HIGH_RES | HEADERLINE | AUTO_TERMINATION | HANDLE_INVALID_REC
The send options without BACKRECEPTION flag.			
INT_SENDING_COPY	-		NO, ALL, FIRST
Controls the BACKRECEPTION flag of INT_OPTIONS			
INT_TERMINATION	-		flags, see SET_ENTRY_MS_MAIL
The actions to be taken on positive or negative termination.			
SET_ENTRY_RS_ORIGINATOR	-	-	
The recipient info of the originator of the sending. Used to address replies and notifications back to the originator. Can be put to the cover by use of text variables.			
L_RECIPIENTS	-	-	
Holds the list of recipients.			

L_ACTIONS_JOB_END - -

Holds the job end actions of the message in case it has been posted as a job.

TS_COMMENTS_1 - -

Extension for TCOSS SMPT / MIME link

TS_COMMENTS_2 - -

Extension for TCOSS SMPT / MIME link

TS_CONTENT_ENCODING - -

Extension for TCOSS SMPT / MIME link

TS_CONTENT_TYPE - -

Extension for TCOSS SMPT / MIME link

TS_SEND_TIME - -

Extension for TCOSS SMPT / MIME link

TS_ENCRYPTED - -

Extension for TCOSS SMPT / MIME link

L_REFERENCES - -

Extension for TCOSS SMPT / MIME link

L_RECEIVED - -

Extension for TCOSS SMPT / MIME link

TS_KEYWORDS - -

Extension for TCOSS SMPT / MIME link

TS_MESSAGE_ID - -

Extension for TCOSS SMPT / MIME link

SET_REPLY_TO - -

Extension for TCOSS SMPT / MIME link

SET_RESENT_FROM - -

Extension for TCOSS SMPT / MIME link

TS_RESENT_MESSAGE_ID - -

Extension for TCOSS SMPT / MIME link

SET_RESENT_REPLY_TO - -

Extension for TCOSS SMPT / MIME link

SET_RESENT_SENDER - -

Extension for TCOSS SMPT / MIME link

TS_RETURN_PATH - -

Extension for TCOSS SMPT / MIME link

SET_SENDER - -

Extension for TCOSS SMPT / MIME link

TS_STATUS - -

Extension for TCOSS SMPT / MIME link

L_EXTENSIONS - -

Extension for TCOSS SMPT / MIME link

INT_USAGE_COUNT - -

Extension for TCOSS SMPT / MIME link

3.6.26 SET_PAGE

Is a possible element of the L_ENV_CONT object. A SET_PAGE effects a page break in the documents view. It may hold page parameters.

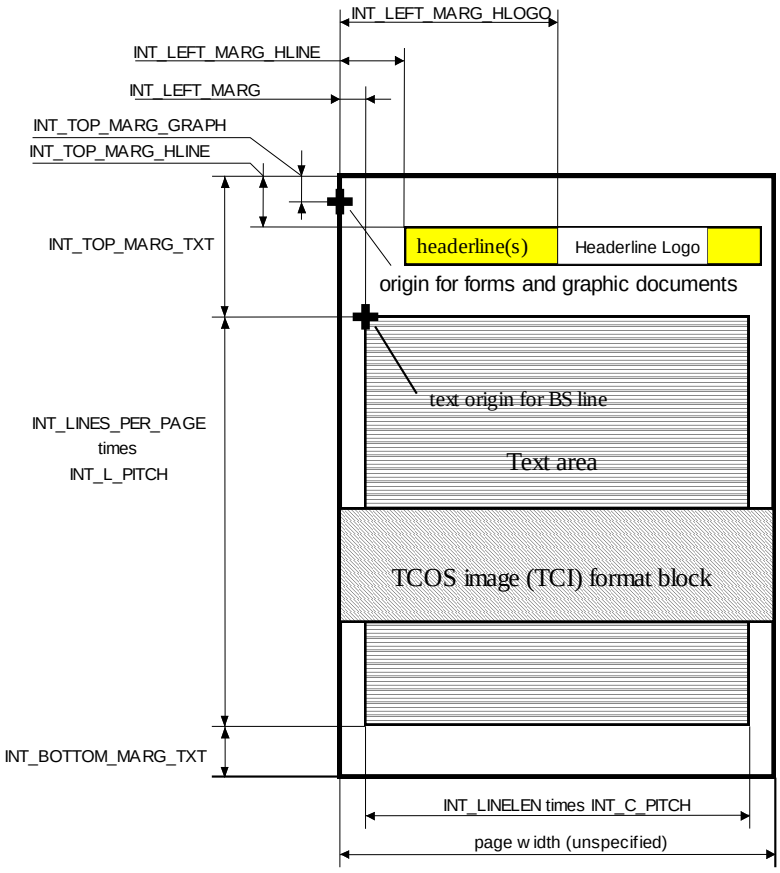
Child objects:	**attributes**	**default,**	**value-restr**
INT_PAGE_FORMAT	-	LAST_USED	LAST_USED, A4H, A4Q, BDH, BDQ, AUTO
Holds the page format. When the page format is LAST_USED the format of the previous page in the list is taken. When there is no such page, LAST_USED defaults to A4H. AUTO denotes an auto make up page. (Is only used in the view on the envelope.)			
TS_HEADERL_N	-	-	
The new header line.			
TS_HEADERL_B	-	-	
The header line from backreception.			
INT_FONT_ID	-	BIG	SMALL / BIG / TELEX_FNT
The font identifier. Selects one of two fonts each for upright and broadside pages.			
INT_TOP_MARG	-	1148 .. 3754	
The distance between the top of the page and the top of the first text line in Twips.			
INT_LINES_PER_PAGE	-		1 .. 59 .. 255
The number of lines that fit on a page between top and bottom margin.			
INT_L_PITCH	-		177 .. 1767 .. 3754
The vertical line pitch in Twips. Minimum 177 for SMALL font, 236 for BIG font.			
INT_BOTTOM_MARG_TXT	-		0 .. 663 .. 3754
The distance between the bottom of the page and the bottom of the last text line in Twips.			
INT_LEFT_MARG	-		388 .. 1798
The distance between the left edge of the page and the start of the text lines in Twips.			
INT_LINELEN	-		1 .. 95 .. 110
The line length in characters.			
INT_C_PITCH	-		85 .. 120 .. 1798
The horizontal line pitch in Twips. Minimum 85 for SMALL font, 113 for BIG font.			
INT_BS_LF_OFFSET	-		0 .. 5
The backspace linefeed offset as in TCOS 5.			
INT_TOP_MARG_HLINE	-		0 .. 227 .. 1445
The distance between the top of the page and the top of the header line in Twips.			
INT_LEFT_MARG_HLINE	-		0 .. 964 .. 14456
The distance between the left edge of the page and the start of the header line in Twips.			
INT_LEFT_MARG_HLOGO	-		0 .. 7370 .. 14456
The distance between the left edge of the page and the start of the header line logo in Twips.			

INT_ TOP_MARG_GRAPH - 0 .. 1445

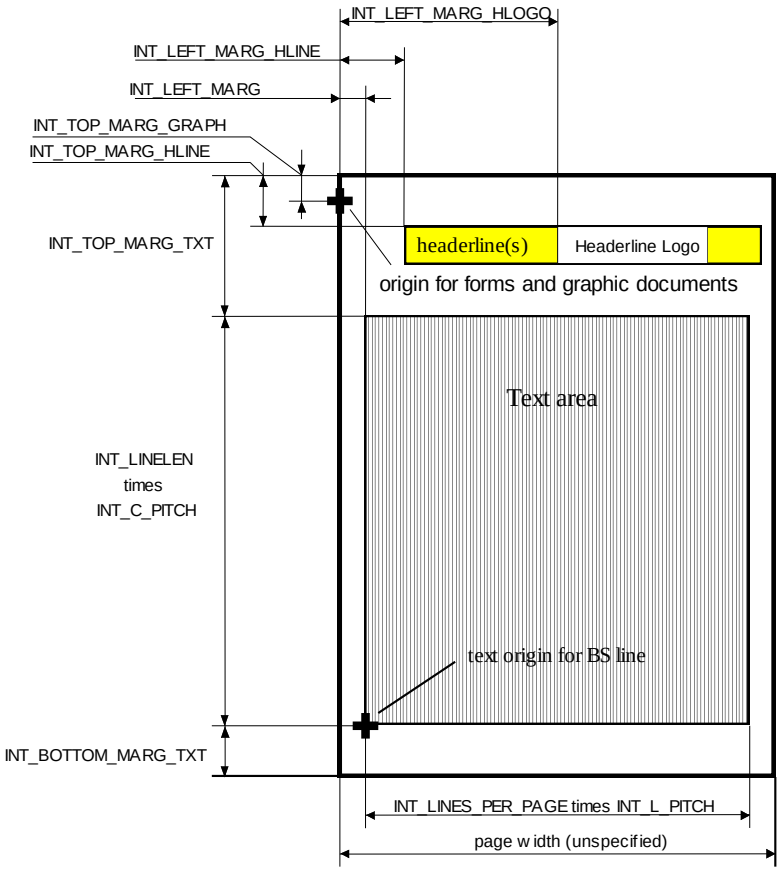
The distance between the top of the page and the top of the first form or graphics block in Twips

See figure at next page.

Portrait Page



Landscape Page


INT_LEFT_MARG_HLOGO
INT_LEFT_MARG_HLINE
INT_LEFT_MARG
INT_TOP_MARG_GRAPH
INT_TOP_MARG_HLINE
INT_TOP_MARG_TXT
headerline(s)
Headerline Logo
origin for forms and graphic documents
Text area
INT_LINELEN
times
INT_C_PITCH
text origin for BS line
INT_BOTTOM_MARG_TXT
INT_LINES_PER_PAGE times INT_L_PITCH
page width (unspecified)

3.6.27 SET_ATT_OBJ

Holds an attached object plus reference to the original source of the object and the creating application. The attached object may be embedded or it may be linked.

Notes on embedding / linking:

Normally all attachments are embedded in the envelope (by value). This can cause two problems:
- the size of the envelope may become impractically large
- if the original attachment has been changed, the changes are not reflected in the envelope

Therefore one needs the option to attach something by reference (link it). In this case always the original is taken. Possible problems:
- the original may not be available (no access to directory; locked . . .)
- changes of the originals affect the contents of the envelope which may no be wanted in some cases.

hild objects: **ttributes** **efault** **value-restr.**

S_SERVICE

filter value for the views on the envelope. Within the cover of a message it defines for which services the attached object applies. Wildcards (asterisk) are supported; the string compare is case insensitive.

xample:
FAX’	xact match
‘FAX*’	matches FAX and FAXISDN
‘*’	matches all (same as child not existing)

NT_MSG_TYPE EXT_DOC | XIMAGE

A filter value for the views on the envelope. Within the cover of a message it defines for which message types this cover applies. The message type is defined by the type of first object in the message content filtered by the view’s capabilities.

S_SERV_ID

The server ID. With this version this is only an info field for the client. (The TS_SERV_ID from SET_APPL_SESSION is taken to find linked objects.)

S_TOS_FOLDER

ee SET_ENTRY_MS.

S_FILE_NAME

The name of the document.

S_COMMENT

Any comment about the attached object.

S_APPL_ID

The ID of the application that created the object.

NT_PG_NUM

The page number from the original document (exists only when the attached object is a page taken from another document).

NT_CONTENT_TYPE ny TCSI object type

The intended type of the linked object. This field may be set by the application for linked objects. It is an info field, so that the client can display the type without having access to the linked object at the server. Since the linked object is stored separately, its current type may differ from the type given in this field. When the linked object is opened the type returned is the current type of the object.
This field is not needed for embedded objects since the c_exist() and c_open() functions always return the true type of the object.

S_CONTENT_ID

Extension for TCOSS SMPT / MIME link

S_CONTENT_TYPE

Extension for TCOSS SMPT / MIME link

S_CONTENT_ENCODING

Extension for TCOSS SMPT / MIME link

_ALT_CONTENT

ist of alternative contents.

N_CONTENT

The attached object in embedded form. May be of any type.

N_CONTENT_LNK

The attached object in linked form. The type of a linked attached object is SET_ENTRY_MS.

When this object is opened the object handler tries to locate the object according to TOS_FOLDER and FILE_NAME. As server name TS_SERVER_ID from SET_APPL_SESSION is taken. If found, the object is opened normally, otherwise ERR_NOT_EXISTING is returned. A linked object follows the same rules as an object opened from a permanent store content (the parent is the permanent store content, the object can be saved as long the server session exists).

Save_at() and c_delete() are not permitted on the linked content. They will give ERR_NOT_PERMITTED. (Linked objects themselves cannot be created here; they have to be created directly in the permanent store.)

Note on contents:

The views on the envelope content will only resolve text blocks, image blocks, and L_ENV_CONTs, of mail and message store entries only the content part is taken, the first level is ignored. Objects of other types are treated as binary blocks. The views will ignore the linked object if an embedded object exists.

To link an object:

Create a SET_ATT_OBJ. Fill at least TS_TOS_FOLDER and TS_FILE_NAME so that they point to the object that is to be linked into the document.
Optionally set the TS_ COMMENT and TS_APPL_ID.
Now you can open the linked object at UN_CONTENT_LNK.

To embed an object:

Create a SET_ATT_OBJ. Fill in any of TS_TOS_FOLDER, TS_FILE_NAME, TS_COMMENT and TS_APPL_ID. These are just info fields for embedded objects.
Now you copy the object you want to embed to UN_CONTENT (with ohh_c_save_at() or ohh_c_close_at()).

3.6.28 SET_SIGNATURE

Is a possible element of the L_ENV_CONT and L_CONTENT_VIEW_TXT objects.

A SET_SIGNATURE holds the userid of the user and the signature itself as an image block.

A signature object may be first inserted to the content list as a place holder defining the position and the userid of the signer. Then, when the user presses the SIGN button the signature image block is inserted from the user entry.

Notes:

- The SET_BLK_IMG object holds the X and Y position, the size and resolution of the signature.
- The meaning of the X and Y positions depends on the preceding object in the parent list.
  The X and Y position specify the offset within the preceding SET_BLK_IMG if it exits (signature is overlaid). If X and Y position do not exist or the preceding object is not an image block the signature is not overlaid.

hild objects: **ttributes** **efault** **alue-restr.**

S_USERID

The userid of the signature holder.

IME_ACTION

he time of signing

NT_PAGE_FORMAT 4H / A4Q / BDH / BDQ

The page orientation of the signature (portrait or landscape).

ET_BLK_IMG

he signature itself.

3.6.29 SET_DIGI_SIGN

Represents a digital signature request. It is intended as a place holder and removed by the document converter on inserting the actual digital signature.

The SET_DIGI_SIGN is an optional child object of the envelope content list L_ENV_CONT, it is not included in a text view (L_CONTENT_VIEW_TXT) of the document.

hild objects: **ttributes** **efault** **alue-restr.**

S_XFIELD

Holds digital signature parameters in an unspecified format (transparent to TCSI).

3.6.30 SET_ACTION

Concepts:

Is a possible element of the L_ACTIONS objects. A SET_ACTION holds a number of filter elements plus the action entry for an in-action.

Child objects INT_DEL_TYPE, INT_ER_RECIPIENT, TS_SERVICE and TS_DOCUMENT_ERR work as filters for the action. The filter controls if the action is executed or not. If one of the filter values does not exist no filtering is done on this value.

When describing an alert a separate group of child objects is used.

hild objects: **ttributes** **efault** **alue-restr.**

NT_DEL_TYPE O_ / CC_ / BCC / AUTHORIZE

he delivery type to which this action applies.

NT_ER_RECIPIENT , 1, 2,

he position of the recipient in the envelope's recipient list (0 = 1st recipient). It allows restricting an action to, e.g., the first recipient.

S_SERVICE

he originator service. Wildcards (asterisk) are supported; the string compare is case insensitive.

Example: 'FAX' exact match
'FAX*' matches FAX and FAXISDN
'*' not matches all (same as child existing)

S_DOCUMENT_ERR

he document error child object allows selecting one of the following filtering options:

TS_DOCUMENT_ERR	selected documents
not existing	all (with or without reception error)
empty string or blank string ("", " ")	without reception error
string with asterisk wildcard ("*")	with reception error
string with specific error code ("XY")	with specific reception error, e.g. "XY"

ET_ENTRY_RS

he action entry.

Child objects of alert: **attributes** **default** **value-restr**

INT_TYPE	-	QUEUELEN	QUEUELEN, QUEUEAGE, QUEUEPAG | SUB_ALERT

he type of alert, queue length, queue age or queued pages. Also holds the optional SUB_ALERT flag which marks an alert as sub-alert.

INT_FOLDER_TYPE	-	IN_FLDR	IN_FLDR, OUT_FLDR

llows to choose between in-box centric and outbox-centric alert definition

INT_ACTIVE	-	ACTIVE	INACTIVE, ACTIVE

he alert's active / inactive setting

INT_MAX_ENTRIES	-	-

The trigger threshold. If the number of queue entries passes INT_MAX_ENTRIES the alert is triggered, and it is not triggered again until the number of queue entries has fallen below the INT_MAX_ENTRIES_U re-activation threshold. In a similar way INT_MAX_ENTRIES and INT_MAX_ENTRIES_U define the 2 thresholds for a warning retraction, in this case INT_MAX_ENTRIES is lower than INT_MAX_ENTRIES_U and the alert is triggered if the number of queue entries falls below INT_MAX_ENTRIES.

INT_MAX_ENTRIES_U	-	-

The re-activation threshold.

INT_TIMEOUT	-	>= ALERT_REPETITION_TIMEOUT_MIN

The alert repetition interval in seconds, minimum supported value is 10 minutes.

INT_PRIORITY, INT_PRIORITY_U	-	-

Act as filter on the queue entry counting.

INT_STATE	-	ST_VIS_IN (active sending queue)	ST_DEFER | ST_DC_QUEUE | ST_VIS_IN | ST_REMOTE | ST_TERMINATE

Allows to filter on message state.

TS_RECIPIENT	-	-

For filtering on the message recipient, wildcards are not supported here.

TS_RECIPIENT_GROUP	-	-

For filtering on the recipient group, wildcards are not supported here.

TS_ORIGINATOR	-	-

For filtering on the message originator, wildcards are not supported here.

TS_ORIGINATOR_GROUP	-	-

For filtering on the originator group, wildcards are not supported here.

TS_FREETEXT	-	-

Text variable, to be inserted into the alert message template replacing the variable "{AlertFreetext}" in the following recipient fields: Department, Full name, Free text and in the TCOSS User ID within a TCOSS address.

SET_ATT_OBJ	-	-

Defines a link (child objects TS_TOS_FOLDER and TS_FILE_NAME) to the actual alert message, which may be stored in a system or private message folder. This message is posted if the alert is triggered.

3.6.30.2 Queue Length Log Agent & NT Performance counters

The queue length log agent and the creation of NT performance counters is controlled by setting a kind of “alert” in the queue’s user profile.

The trigger and re-activation threshold, which must have different values for a real alert, are set to the same value. Alternatively only the threshold INT_MAX_ENTRIES may be used to hold the value and the re-activation threshold INT_MAX_ENTRIES_U is left empty (child does not exist).

A value of zero activates an NT performance counter; any non-zero value activates the queue length log agent plus an NT performance counter. The value itself sets the log interval in seconds; the actual log interval is rounded up to multiples of the alert check interval set in the system configuration.

The folder name TS_TOS_FOLDER defines the field name for the log agent and the NT performance counter name.

The file name TS_FILE_NAME is not used and left empty.

The priority filter is used in the same way as for an alert, restricting the queue length counting to messages of a given priority. This means it is possible to have e.g. counters for the queue lengths of each priority level.

Performance counter object: “TCOSS queues” (or “TCOSSn queues” in an ASP environment)

Performance counter instance: empty

Performance counter name: as defined in folder field of alert

The new TCOSS queue length log agent writes log entries in the user-defined format to the TCOSS short term archive.

The queue lengths of all users who requested this feature are packed into one or more log entries having the following layout:

SET_ENTRY_ARCHIVE	
INT_MSG_TYPE	... new constant LOG_ENTRY
TIME_ACTION	... time of logging
TS_LOG_USER	... user ID = “QUEUELEN”
UN_CONTENT/SET_ATTRIBUTE	
UN_NAME	... “QUEUELEN_LOG”
UN_VALUE/L_ATTRIBUTES	... list of queue length attributes
SET_ATTRIBUTE	
UN_NAME	... field name specified in folder name of alert
UN_VALUE	... integer value: total queue length
SET_ATTRIBUTE	
UN_NAME	... e.g. “FAX high priority”
UN_VALUE	... integer value: total queue length

The queue length data of around 120 users (with user ID length of 8 characters) will fit into a single log entry. If the queue length data can’t be put into a single record because of the 4 KB size restriction of the log entries, two or more log entries are written for each scan interval.

3.6.30.3 Queue Age Monitoring

The “queue age” is defined as the time difference between the first message in the queue (the message which is waiting longest) and the current time. It is always specified in seconds unless stated otherwise.

If queues of different priority are monitored together (by specifying a priority range in the SET_ACTION structure) the queue age of the collection is defined as the maximum queue age of all its members.

The age of the fax queue goes up if

a) There are not enough free lines to handle the outgoing traffic
b) Number locking is active and there are several messages for the same recipient being sent on the limited number of lines defined in LN99 (not enough free lines at the recipient's side)

Queue age monitoring has been implemented as an extension to queue length monitoring. Both use the same SET_ACTION structure extended with a INT_TYPE child object:

SET_ACTION	
SET_ATT_OBJ	
TS_TOS_FOLDER	
TS_FILE_NAME	
INT_MAX_ENTRIES	
INT_MAX_ENTRIES_U	
INT_PRIORITY	
INT_PRIORITY_U	
INT_TYPE	**... 0 = queue length / 1 = queue age**

The INT_TYPE value allows to select queue length or queue age for alerting, logging and performance counters. Default value of INT_TYPE is 0 (= queue length).

The queue age values of all users who requested queue age logging are packed into one or more log entries having the following layout:

SET_ENTRY_ARCHIVE	
TIME_ACTION	... time of logging
TS_LOG_USER	... user ID = "QUEUEAGE"
UN_CONTENT/SET_ATTRIBUTE	
UN_NAME	... "QUEUEAGE_LOG"
UN_VALUE/L_ATTRIBUTES	... list of queue age attributes
SET_ATTRIBUTE	
UN_NAME	... field name specified in folder name of alert
UN_VALUE	... integer value: queue age in seconds
SET_ATTRIBUTE	
UN_NAME	... e.g. "FAX high priority"
UN_VALUE	... integer value: queue age in seconds

3.6.30.4 Queued Pages Monitoring

Similar to queue length monitoring, the number of pages in a queue may also be monitored. The page-based monitoring is selected by setting INT_TYPE = QUEUEPAG (2).

The created log entries are analogous to the queue length log, the log user is "QUEUEPAG" and the structure name will be set to "QUEUEPAG_LOG".

3.6.31 SET_FILTER

The base class for all filters for the folders of permanent stores. The filter of a folder defines its view on the contents of the store.

Generally, all objects of type INT_ , TIME_ or TS_ that are direct child objects of the entry can be used to filter the view.

Filtering:

The view is the logical AND of all the restrictions applied by the filter values given.

For integers and time objects a minimum and a maximum value can be specified. If you want to filter for one discrete value minimum and maximum must be set to the same value.

For strings the filter value is also a string. It may be either:

1. "abcd" filter for exact value
2. "abcd*" string has to start with 'abcd'
3. "*abcd" string must end with 'abcd'
4. "*abcd*" 'abcd' must be found in string at any place

Upper / lower case is ignored for string filtering.

When an object does not exist in the filter it is taken as wildcard.

When an object is in the filter but does not exist in the entry the entry is not selected.

Depending on the used store some fields are used to reduce the potential folder results in an optimized first step so that using these fields may significantly reduce the response times.

Naming convention for child objects in filters:

For string filters the same name as in the entry is used.

For integer or time filters the same name as in the entry defines the lower bound, for the upper bound "_U" is added to the name.

The naming convention provides a full definition of all the filter subclasses for the different kinds of stores.

Store specific execptions from this general rules are described in the corresponding sub chapters.

3.6.31.1 SET_FILTER_MS

see structure diagram SET_ENTRY_MS

The wildcard '?' matching any single character is supported in TS_TOS_FOLDER and TS_FILE_NAME.

Bit-wise filtering

Bit-wise filtering is used for these child objects: INT_DOC_CLASS. See example for INT_RIGHTS_APPL below.

Filtering on TS_DOCUMENT_ERR supports 2 special cases:

TS_DOCUMENT_ERR = "??" … matches all non-blank error codes i.e. any error

TS_DOCUMENT_ERR = "" … matches the blank error code i.e. no error

3.6.31.2 SET_FILTER_MS_MAIL

see structure diagram SET_ENTRY_MS_MAIL

Bit-wise filtering

Bit-wise filtering is used for these child objects: INT_STATE, INT_ANNOTATIONS. See example for INT_RIGHTS_APPL below.

Optimized filter fields:

The following fields

TCSI Field	In-box label	Out-Box label	Notes
INT_STATE	Status	Status	
TS_ORIGINATOR	…From Continued	From	1
TS_ORIGINATOR_GROUP	Orig. Group	Orig. Group	1
TS_RECIPIENT	To	… To Continued	1
TS_RECIPIENT_GROUP	Recip. Group	Recip. Group	1
TS_FILE_NAME	File name	File name	
TS_TC_MSG_ID	Message ID	Message ID	2
TIME_ACTION	Received/Term.	Send Time	

Notes:

1) Optimization is used only if the value is specified without wildcards.
2) **Filter is case sensitive and wildcards are not supported!** This means that no entries are returned if a wildcard is included. Consider also that the message ID filter does not have this restrictions in MS_MAIL_ARC where the default filter behaviour (case insensitve and wildcards are supported) is used.

3.6.31.3 SET_FILTER_MS_MAIL_ARC

see structure diagram SET_ENTRY_MS_MAIL_ARC

Bit-wise filtering

Bit-wise filtering is used for these child objects: INT_STATE, INT_ANNOTATIONS. See example for INT_RIGHTS_APPL below.

Optimized filter fields:

The following fields

TCSI Field	In-box label	Out-Box label	Notes
TS_ORIGINATOR	…From Continued	From	1
TS_RECIPIENT	To	… To Continued	1

TS_RECIPIENT_GROUP	Recip. Group	Recip. Group	1, 2
TIME_ACTION	Received/Term.	Send Time	

Notes:

1) Optimization is used only if the value is specified without wildcards.
2) Optimized since KCS 9.1.1.

3.6.31.4 SET_FILTER_US

see structure diagram SET_ENTRY_US

Bit-wise filtering

Bit-wise filtering is used for these child objects: INT_RIGHTS_APPL, INT_RIGHTS_USERPROF, INT_RIGHTS_YN_2, INT_RIGHTS_YN, INT_RIGHTS_RW

To allow for example a client to easily get a list of users which have the right to post jobs, the filtering on INT_RIGHTS_APPL in the user store folder uses bit wise filtering.

The following table shows how the individual bits in INT_RIGHTS_APPL(_U) have to be set:

filter on bit value	set bit in INT_RIGHTS_APPL	set bit in INT_RIGHTS_APPL_U
0	0	0
1	1	1
don't care	0	1

It is possible to filter a user folder on users which do not have an address of a specific service. Here is an overview of available filter options:

TS_FREE_ADDRESS child of SET_FILTER_US	entries contained in filtered folder list
not existing or empty string ""	all (no filtering)
wildcard "*"	non-empty address
blank string " "	empty address
string plus wildcard "12*"	address starting with "12"
string in wildcards "*12*"	address containing the string "12"
specific string "1234"	address "1234"

3.6.31.5 SET_FILTER_RS

see structure diagram SET_ENTRY_RS

The recipient store may by used with the following values of folder types (INT_FOLDER_TYPE):

Default (=IN_FLDR): Regular recipient folder to display a specific address book. Note that TS_SECTION is mandatory and selects the wanted address book, but wildcards can be used to select multiple or all address books. An empty or missing TS_SECTION value is depreciated with this folder type, because its behavior is TCOSS specific. See bug 705643 for details.

ADDRESS_MAP_FLDR	Address mapping of users.
CHANGES	Get changes for directory syncronization.

Filtering on Type of Entry

The address book holds objects of three different types:

Addresses of users: INT_TYPE = USER (1)
Addresses without user profile: INT_TYPE = RECIPIENT (30)
Distribution lists: INT_TYPE = DISTRIBUTION_LIST (60, this constant is new)

The INT_TYPE (lower limit) and INT_TYPE_U (upper limit) child elements of the SET_FILTER_RS object may be used to specify a range or a particular type value.

Examples:

A SET_FIILTER_RS with INT_TYPE = USER (1) and INT_TYPE_U = RECIPIENT (30) selects all addresses, with or without user profile.

A SET_FIILTER_RS with INT_TYPE = DISTRIBUTION_LIST (60) and also INT_TYPE_U = DISTRIBUTION_LIST (60) selects only distribution lists.

Filtering on Parent

The SET_DL_PARENT object allows to select all recipients (or nested lists) contained in a particular distribution list. The TS_SECTION and TS_RECP_ID child objects of SET_DL_PARENT are used to specify the parent distribution list. Wildcards are not allowed and the parent distribution list should exist in the address book, otherwise the folder display is empty.

Example:

```
SET_FILTER_RS
  SET_DL_PARENT
    TS_SECTION = “BMW”
    TS_RECP_ID = “LIST27”
  TS_SECTION = “BMW”
```

The above filter selects all child elements of “LIST27” which is stored in the private address book of user “BMW”. If all elements of this list are also stored in the same private address book filtering a second time on TS_SECTION = “BMW” improves performance.

Filtering on TS_SECTION at the first level of the recipient filter is also required to pass the server’s rights check if the folder display is asked by a user who does not have unrestricted rights in the address store.

Note that the ohh_c_open() function in the recipient store content with a distribution list as selector entry also returns the complete list of its child elements, and with better performance. The advantage of the filtered folder display is that one can request partial lists and use additional sorting and filtering options.

Filtering on List Element

The SET_ENTRY_RS object in SET_FILTER_RS allows to select all distribution lists containing a particular recipient (or nested list). The TS_SECTION and TS_RECP_ID child objects of SET_ENTRY_RS are used to specify the contained list element. Wildcards are not allowed and the recipient (or nested list) should exist in the address book, otherwise the folder display is empty.

Example:

```
SET_FILTER_RS
  SET_ENTRY_RS
    TS_SECTION = “BMW”
    TS_RECP_ID = “Customer7”
  TS_SECTION = “BMW”
```

The above filter selects all distribution lists which contain a reference to the recipient “Customer7” which is stored in the private address book of user “BMW”. If all distribution lists containing this recipient are also stored in the same private address book filtering a second time on TS_SECTION = “BMW” improves performance.

Filtering on TS_SECTION at the first level of the recipient filter is also required to pass the server’s rights check if the folder display is asked by a user who does not have unrestricted rights in the address store.

Since TCOSS ()

Filtering Distribution Lists on Number of Elements

The INT_N_OF_ENTRIES (lower limit) and INT_N_OF_ENTRIES_U (upper limit) child elements of the SET_FILTER_RS object may be used to filter distribution lists on number of entries.

Examples:

A SET_FILTER_RS with INT_N_OF_ENTRIES = 0 and also INT_N_OF_ENTRIES_U = 0 selects all empty distribution lists.

A SET_FILTER_RS with INT_N_OF_ENTRIES = 1 selects all non-empty distribution lists.

Filtering on Usage Count

The INT_USAGE_COUNT (lower limit) and INT_USAGE_COUNT_U (upper limit) child elements of the SET_FILTER_RS object may be used to filter recipients (or nested distribution lists) on the number of references pointing to it.

Examples:

A SET_FILTER_RS with INT_USAGE_COUNT = 0 and also INT_USAGE_COUNT_U = 0 selects all recipients and distribution lists which are not contained in any distribution list.

A SET_FILTER_RS with INT_USAGE_COUNT = 1 selects all recipients and distribution lists which are contained in at least one distribution list.

Filtering on Vanity Name

The TS_FULLNAME_VANITY child object of SET_FILTER_RS allows to filter on the TS_FULLNAME field specifying the name’s vanity representation. This is only implemented for addresses of type user.

The full name of each recipient of type user in TCOSS will be split into words (e.g.: “Larry Nunn” into “Larry” and “Nunn”. Or “James Tiberius Kirk” into “James”, ”Tiberius” and “Kirk”). Each of the words of a name will be transformed into its vanity representation (eg.: “Larry” -> “52779”, “Tiberius” -> “84237487”) and an index is created automatically .

This gives TCOSS the possibility to search for “James Kirk” and also for “Kirk James Tiberius”. The filter string specified in TS_FULLNAME_VANITY is always handled as if it had a wildcard at the end, for example “547” (Kir) will return “Kirk” and “Kirmet” but it will not return “Kim”.

The vanity representation is based on the ITU standard E.161 Option A (05/95) which specifies the following keypad mapping:

	2 A B C	3 D E F
4 G H I	5 J K L	6 M N O
7 P Q R S	8 T U V	9 W X Y Z
*	0	#

The standard does not define blanks. Therefore TCOSS allows “0” and “1” as blanks, but it is not necessary to put a blank between 2 names (eg.: 527796866 is also recognized as Larry Nunn).

Digits in names are supported as long as the name starts with a letter. This means that it is possible to call “Kirk4711” (54754711) or “Larry0815” (527790815).

3.6.31.6 SET_FILTER_SS

see structure diagram SET_ENTRY_SS

3.6.31.7 SET_FILTER_REGS

see structure diagram SET_ENTRY_REGS

3.6.31.8 SET_FILTER_ARCHIVE

Concepts:

Holds all the information needed for searching the archive. There are basically two types of search restrictions:
a) date / time of message sending / reception
b) strings in the message text or in one or more fields of the message header and the mail entry

Child objects:	**attributes**	**default**	**value-restr.**
TIME_ACTION	-	-	-
Restricts the search to messages sent / received at or after this date / time.			
TIME_ACTION_U	-	-	-
Restricts the search to messages sent / received at or before this date / time.			
L_TEXTFILTER	R/O	-	-
The list of attributes to match the text, header or entry fields.			

3.6.31.9 SET_ATTRIBUTE

Concepts:

Defines a search restriction for archived messages. Wildcards and a list of string values allow to specify different forms or spellings of search words. Words may be searched in one, several or all message fields.

Child objects:	**attributes**	**default**	**value-restr.**
UN_VALUE / L_STRINGS	-	-	-
The search strings. The order in the list is not relevant; the strings are searched applying a logical ‘OR’. Wildcards in the form of an asterisk at the end of the string are allowed.			
UN_NAME / L_STRINGS	-	-	-
A list of strings that specifies the field(s) where the words given in UN_VALUE are searched applying a logical ‘OR’. If this child object does not exist all fields are searched. Possible name strings are:			

"SET_BLK_TXT"	text content of message (includes OCR text blocks)
"TS_REF"	reference field
"TS_RECIPIENT"	TCOSS user ID or queue of recipient
"TS_ORIGINATOR"	TCOSS user ID or queue of originator
"TS_RECIPIENT_GROUP"	recipient group
"TS_ORIGINATOR_GROUP"	originator group
"TS_DOC_NR"	document number field
"TS_ENV_NAME_POSTED"	original message name
"TS_NORMALIZED_ADDR"	normalized address
"TS_NORMALIZED_ORIG"	normalized originator
"TS_ORIGINATOR_INFO"	originator info
"ORIGINATOR.TS_RECP_ID"	recipient ID of originator
"ORIGINATOR.TS_COMPANY"	company of originator

"ORIGINATOR.TS_DEPTM"	department of originator		
"ORIGINATOR.TS_FULLNAME"	full name of originator		
"RECIPIENT.TS_RECP_ID"	recipient ID		
"RECIPIENT.TS_COMPANY"	company of recipient		
"RECIPIENT.TS_DEPTM"	department of recipient		
"RECIPIENT.TS_FULLNAME"	full name of recipient		
INT_TYPE	-	-	ATTR_POSITIVE / ATTR_NEGATIVE

ATTR_POSITIVE	(default value): normal search, messages which contain any of the value strings in any of the name fields are included in the search result
ATTR_NEGATIVE	negative search, messages which contain any of the value strings in any of the name fields are excluded from the search result

3.6.32 SET_CONTENT_VIEWS

Holds the text and image views on the content of the envelope.

Empty pages are not shown in any view.

Child objects:	**attributes**	**default, value-restr.**
L_CONTENT_VIEW_TXT	-	-

A list that holds text blocks, image blocks, forms and page objects. Attached objects are put to the list flatly, within the cover part of the content the variables are resolved. Alternative contents to a binary block are evaluated and the first displayable entry of the alternative content list is included in the view. OBJ_BODY_PART is also included in the text view to mark the beginning of the message content.

L_CONTENT_VIEW_IMG	(not implemented)	-

The list that holds the page images and page objects. The image view is based on the text view. All text parts are resolved to image form.

INT_NPAG	R/O	-

After the text view is opened INT_NPAG holds the number of pages for this view.

TS_SERVICE	-	-

Defines for which service the view is created when L_CONTENT_VIEW_TEXT is opened.

INT_SERVICE_CAPS	-	TEXT_DOC | XIMAGE

The capabilities of the service for which the view is created.

3.6.32.1 Text Preview with rendering

As an alternative to the image preview, which is not implemented, the text preview (L_CONTENT_VIEW_TXT), which resolves cover sheets and produces a flat list of text and image blocks, has been enhanced to do rendering of Unicode text blocks.

The rendering is activated by setting the new child object INT_RENDER_CODEPAGE to a value of 65001 (CP_UTF8) in the SET_ENTRY_MS parent before opening the text preview. In the resulting preview all text blocks with a code page other than TCOSS 0 or 1 are converted to image blocks.

The Windows Font which is used to render Unicode characters is selected using registry key “HKEY_LOCAL_MACHINE\SOFTWARE\TOPCALL\CodePages\65001”, the same as used by TCOSS, see chapter “Unicode Rendering” above.

Also make sure to set the INT_CODEPAGE value in the application session correctly, it is used to convert cover variables into the appropriate text code page and to insert cover variables as Unicode into RTF cover sheets.

3.6.33 SET_CLIENT_SETUP

Concepts:

Holds the setup for the different types of clients a user may have. When a user does not need a certain type of client the corresponding setup does not exist.

The content of the child objects is local to the clients of a certain type. Thus, the definition of the child objects is not part of the client - server interface. TCSI only defines the names of the child objects. The types (real classes) have to be defined in a separate header file.

Child objects:	**attributes**

SET_FAX_CL_SETUP -

Holds the setup for the FAX client. The FAX client setup controls the behavior of the system when the user logs in per phone or FAX machine.

SET_WS_CL_SETUP -

Holds the setup for the PC client.

SET_PRO_CL_SETUP -

Holds the setup for the FAX Desk Pro PC client.

SET_FAX_CH_SETUP -

Holds the setup for the FAX channel. The FAX channel is an MDA (message delivery agent) that logs in automatically.

SET_MAP -

To support all future types of clients. The map object is a flexible hierarchical storage of all common types of configuration values needed by a client application.

3.6.34 SET_BLK_TXT

Concepts:

A text block holds a block of text lines plus text formatting information.

ASCII representation:

```
value : ( [ object ] { , [ object ] } , ( [ "line 1" ] {, "line 2"} ) ) a
```

a) See class TEXTSTRING for description of string content.

Child objects:	**attributes**	**default, value-restr.**
INT_TYPE	-	NORMAL / OCR_FRAME / APICMD / TESTKEY

The type of the text block. In case of OCR the first line defines the OCR frames + contents, the next lines are printed over the OCR frames.
APICMD: text blocks holding info from printer driver (serial printing)
TESTKEY: block holds test key as input by authorizer

INT_FONT_ID	see SET_PAGE	
INT_LEFT_MARG	see SET_PAGE	
INT_L_PITCH	see SET_PAGE	
INT_LINELEN	see SET_PAGE	
INT_C_PITCH	see SET_PAGE	
INT_BS_LF_OFFSET	see SET_PAGE	
INT_BASED_ON	R/O	PREV_SET_/ CONFIGURED

The base for block attributes that do not exist. (Info field for View.)

PREV_SET: take values from last text block in list
CONFIGURED: fall back to configured values

Only included for compatibility - may not be set by client.

INT_CODEPAGE	-	0 / 1 / 932 / ...

The code page of the text block. If INT_CODEPAGE is not specified, the code page of the text may be either TCOSS 0 or TCOSS 1 (undefined between 0 and 1).

UN_CONTENT	-	-

The text content.

3.6.35 SET_BLK_IMG

An image block holds a block of image lines plus resolution information.

ASCII representation:

```
value : ( [ object ] { , [ object ] } , ( [ "TCI line 1" ] {, "TCI line 2"} ) )
```
[a]

a) See description of TCI format for description of string content.

Child objects:	**attributes**	**default, value-restr.**
INT_TYPE	-	NORMAL / FAX_RECEPTION

The type of the image block. Blocks from FAX reception contain a header line and may be treated differently.

INT_X_RES	-	204

The x resolution in dots per inch.

INT_Y_RES	-	98

The y resolution in dots per inch.

INT_X_POS	-	0

The x position in Twips.

INT_Y_POS	-	0

The y position in Twips.

INT_N_OF_LINES	R/O	-

The actual number of pixel lines in the image block.

INT_ORG_IDX	R/O

Index to L_ENV_CONT object, exists only in child objects of L_CONTENT_VIEW_TXT.

The viewer/editor changes image blocks in the L_CONTENT_VIEW_TXT. These changes should be reflected into the L_ENV_CONT. A mapping is required from image blocks in the L_CONTENT_VIEW_TXT to image blocks in the L_ENV_CONT.

All objects in the L_CONTENT_VIEW_TXT which are editable contain an INT_ORG_IDX which contains the index of the original object in the L_ENV_CONT. Editable objects are SET_BLK_IMG and SET_BLK_IMG_FORM objects which are in the first level of the L_ENV_CONT or in the UN_CONTENT of a SET_ATT_OBJ which is in the first level of the L_ENV_CONT.

NOTE: An editable object in the UN_CONTENT of a SET_ATT_OBJ contains the index of the SET_ATT_OBJ in INT_ORG_IDX.

TS_FREETEXT	-	-

Any info, such as reception quality information.

INT_PIXEL_WIDTH	-	-

Width of the image block in pixels. The image block always defines the full width (1728 pixel), INT_PIXEL_WIDTH specifies the valid width beginning from left.

UN_CONTENT	-	-

The content as TCI, Huffman or G4 block.

ohh_b_get_irl (h_set_img_blk, pbuffer, pnum_of_rls, index)

Note: not implemented in current release (empty function) - use TCIMGIO.DLL instead.

Gets the IRL[8] line (index) from the image block.
Returns the number of runlengths that are in the irl buffer.

errors: ERR_END_OF_BLOCK the end of block is reached

ohh_b_put_irl (h_set_img_blk, pbuffer, num_of_rls)

Note: not implemented in current release (empty function) - use TCIMGIO.DLL instead.

Puts one IRL line to the image block. The number of run lengths must be specified.

errors: none

ohh_b_get_huf (h_set_img_blk, pbuffer, psize, index)

Note: not implemented in current release (empty function) - use TCIMGIO.DLL instead.

Gets the huffman code line (index) from the image block.
The size of the huffman code in byte is returned.

errors: ERR_END_OF_BLOCK the end of block is reached

ohh_b_put_huf (h_set_img_blk, pbuffer, size)

Note: not implemented in current release (empty function) - use TCIMGIO.DLL instead.

Puts one huffman code line to the image block.

errors: none

ohh_b_get_bitm (h_set_img_blk, pbuffer, index)

Note: not implemented in current release (empty function) - use TCIMGIO.DLL instead.

Gets the bitmapped line (index) from the image block.
The size of the bitmapped line is fixed to 216 bytes (1728 pixels).

errors: ERR_END_OF_BLOCK the end of block is reached

ohh_b_put_bitm (h_set_img_blk, pbuffer)

Note: not implemented in current release (empty function) - use TCIMGIO.DLL instead.

Puts one bitmapped line to the image block.
The size of the bitmapped line is fixed to 216 bytes (1728 pixels).

errors: none

[8] Integer run length format. One (short) integer per run length.

3.6.35.2 SET_BLK_IMG_FORM

Is a possible element of the L_ENV_CONT object. A form defines the overlay for the following pages in the documents view. The form does not have a page format or orientation. If the form's image is larger than the page it is used on, it is cut off.

The SET_IMG_BLK_FORM is an image block with additional validity information.

Child objects:	attributes	default, value-restr.
INT_VALID_PGNO	-	0 ... n

Gives the relative page number, counted from the form's position, where the form becomes valid.

INT_VALID_PGCNT	-	1 ... n, 0

1...n Gives the number of pages the form is valid.

0 Valid until definition of next form

+ all child objects from SET_IMG_BLK.

3.7 LIST

A list holds a number of objects.

general errors:

ERR_OBJ_NOT_EXIST	no list element at the given index

ohh_c_any function ( )

For all container functions the id parameter is replaced by the parameter index.
The index must be between 0 and (number_of_elements-1) otherwise ERR_OBJ_NOT_EXIST is returned.

errors: none

ohh_l_length (h_l, pnumber)

Returns the number of elements in the list.

errors: none

ohh_l_insert (h_l, index, h_obj)

Inserts the specified object in the list, at the position **index**. The other list items are moved up one position. The handles of the other list items (if any are open) remain valid.

With index = 0 the object is inserted at the beginning of the list,
with index = AFTER_END the object is appended to the list.

errors: none

***ohh_c_delete* (h_l, index)**

Deletes the list element at the position **index** from the list.
If open copies of the target exist their original is set to changed.

The other list items are moved down one position. The handles of the other list items (if any are open) remain valid.

index = 0 deletes the first object of the list,
index = AT_END deletes the last object of the list.

errors: none

ohh_l_move (h_l, index_old, index_new)

Moves the object at **index_old** to **index_new**. The other list items in between are moved up or down one position. All handles to list items (if any other are open) remain valid.

With index_new = 0 the object is moved to the beginning of the list,
with index_new = AT_END the object is moved to the last position.

errors: none

ohh_l_move_hdl (h_l, h_obj, index_new)

Moves a list object to **index_new**. The other list items in between are moved up or down one position. All handles to list items (if any other are open) remain valid. The handle to the list object (h_obj) must have been obtained by opening it from the same list.

With index_new = 0 the object is moved to the beginning of the list,
with index_new = AT_END the object is moved to the last position.

errors: ERR_NOT_PERMITTED object to be moved is not a child of this list

3.7.2 Memory Allocation Parameters for Lists

The default values for handle memory allocation are set to the maximum supported values. Registry changes are only required to restrict the maximum amount of memory allocation.

The following table shows the limits and default values:

	External Handles	Internal Handles	Object Memory
Registry Value (REG_DWORD)	“HandleSpace1”	“HandleSpace2”	“ObjSegments”
minimum value	0x8	0xA	0x2
maximum value	0x1200	0x4B00	0x1000
default value	0x1200	0x4B00	0x1000
maximum number of handles	131 072	983 040	
maximum amount allocated	4 718 592 Bytes	19 660 800 Bytes	256 MB

The limit for lists is 65 535 child elements.

Memory allocation for system tables holding internal and external handles is done on demand. At start up time only 224 Kbytes of memory is committed for 4096 external and 4096 internal handles. Additional memory is allocated as the number of handles in use is growing, up to the configurable maximum amount.

3.7.2.1 L_SERVERS

The list of server entries.

List elements:	attributes	default
SET_SERVER_ENTRY	-	-

3.7.2.2 L_DISKS

The list of disks that are connected to a hardware node.

List elements:	attributes	default
SET_DISK_STATUS	-	-

3.7.2.3 L_SUBNODES

The list of sub-nodes that are connected to a hardware node.

List elements:	attributes	default

SET_NODE	-	-

3.7.2.4 L_FOLDER_DISPLAY

Holds folder entries. The type of the entries depends on the type of the store.

The list elements are copies of the stores entries. The list shows all folder entries that match the filter specification (see: SET_FOLDER).

List elements:	attributes	default
SET_ENTRY_type_of_perm_store:	-	-

3.7.2.5 L_ACTIONS

Holds the list of actions for events such as in mail, out mail, notification received and so on.

List elements:	attributes	default
SET_ENTRY_RS	-	-
SET_ACTION	-	-

3.7.2.6 L_FULL_ADDR

For general purpose use.

List elements:	attributes	default
SET_FULL_ADDRESS	-	-

3.7.2.7 L_ENV_CONT

	attributes	default

Holds the contents of an envelope.
List elements:

SET_HEADER	-	-

Holds all the sending related info. To be accepted by the mail system an ENTRY_MS_MAIL must have a header at any position in the list.

SET_PAGE	-	-

A page break with an optional format definition.

SET_BLK_TXT	-	-

A text block.

SET_BLK_IMG	-	-

An image block.

SET_BLK_IMG_FORM	-	-

A form definition for any of the following pages.

SET_ATT_OBJ	-	-

Any attached object. The object may be linked or embedded.

Note: Attached objects that are of type SET_BLK_IMG or SET_BLK_TXT are resolved as if they were direct members of the L_ENV_CONT by all views. Other types of attached objects are treated as binary blocks by the views.

OBJ_BODY_PART	-	-

A mark in the list that defines the beginning of the body part of the envelope. Up to this mark the list holds the cover part. This object must be the first in the list for envelopes with no cover part.

SET_SIGNATURE	-	-

A signature object.

3.7.2.8 L_CONTENT_VIEW_TXT

A list that holds text blocks, image blocks, forms and page objects. In text blocks in the cover part the text variables are resolved.
The view is always on the contents of the envelope at the time this object was opened. When the envelope's contents are changed, the view has to be closed and opened again to reflect the changes.

List elements:	attributes	default
SET_PAGE	-	-

A page break with an optional format definition.

SET_BLK_TXT	-	-

A text block.

SET_BLK_IMG	-	-

An image block.

OBJ_BODY_PART	-	-

Marks the beginning of the body part.

The text view is generated by resolving the envelope content in the following steps:

1. INT_ER_RECIPIENT and INT_ER_ALT_ADDR_LEFT from the message store or mail entry are taken to determine the selected recipient and its currently selected active alternative address. If INT_ER_RECIPIENT does not exist, the first recipient from the envelope's header is assumed to be selected. If INT_ER_ALT_ADDR_LEFT does not exist or has value 0, the first active address is selected.
2. The capabilities associated with the selected address are presumed according to the address type: TEXT_DOC capability for types SET_TX_ADDRESS and SET_TTX_ADDRESS, TEXT_DOC | XIMAGE capability for all other types.
3. the first object in the content part of the message matching the capabilities is located, its type defines the **message-type** (XIMAGE if image, TEXT_DOC for all other cases)
4. the TS_SERVICE from the selected address and the evaluated message-type are matched against the filter values for all attached objects in the cover part of the message. Only the first matching attached object is included in the view, the others are skipped regardless of their filter values. This skipping rule works only at one level (the level where the first filtered attached object is found) and applies only to filtered attached objects (TS_SERVICE or INT_MSG_TYPE exists).
5. When attached objects are included, their content with all child objects is resolved and all text blocks, image blocks, forms and page objects are put flatly to the envelope's content list. (Note: the view ignores the linked object if an embedded object exists within an SET_ATT_OBJ.)
   When the list holds other lists of the same type (directly or as attached objects) these are resolved flatly up to nesting level 5.
6. In text blocks in the cover part, in TS_HEADERL_N and TS_HEADERL_B (in the entry) the text variables are resolved.
   If the form xxx of the variable is used, the resolved length of the variable depends on the value of the variable. If the form $xxx_____$ is used, the length is fixed including the $ signs. Values that are too long are automatically truncated.
7. Pages are made up as described in the separate chapter "repagination algorithm".
   On landscape pages image blocks are ignored for page makeup, they still exist in the generated text view.
   Layout values are used in the following order:
   - the values from the text block
   - the values existing in the current page
   - the values set for previous pages
   - the standard configuration for the base format (A4H, A4Q, ...)
   (The local default values for text block and page objects are not used within the text view).

Repagination Algorithm

Manual Page Breaks

SET_PAGE objects within the content are considered page-breaks only if the format (INT_PAGE_FORMAT) is not AUTO, e.g. A4H, A4Q, … (hard breaks).

Note: The page format LAST_USED is treated as page-break even if the previous format was AUTO!

SET_PAGE objects with format AUTO (soft break) are not inserted into the resulting content list. Furthermore, TCSI ignores all other objects (settings for margins, line length, pitch …) in soft page breaks (AUTO).

SET_PAGE objects inserted because of repagination have the format LAST_USED.

Automatic Page Breaks in Image Blocks

SET_BLK_IMG objects may be divided into multiple image blocks if the image block extends beyond the current page length.

Several rules apply to determine if and when an image block may be divided and a page break is inserted.

An image block **never** gets divided if it is the first content (text or image) on the page. An image block following a hard page break (non-AUTO format) and **not** preceded by any other text or image block is not divided by a new page break.

For all other image blocks a page break is inserted if the image block extends beyond the bottom margin

- and an empty image line is detected
  - break at the position of the next non-empty image line
  - or at the end of the block if there is a text or image block following
- or the block extends 3mm (12 normal mode pixels) before the end of the page
  - break at the position of the next non-empty line following
  - or at the end of the block if there is a text or image block following

The above rules suppress empty image lines at the beginning of a page after an automatic page break. Empty pages at the end of a message, caused by empty lines at the end of an image block, are suppressed as well.

Restriction: Image blocks in SET_SIGNATURE objects are **never** divided into multiple blocks. I.e. handling of signatures remains unchanged. Page breaks can only be inserted after a block and do not divide a block.

In case signatures are inserted into a message at the bottom of a page, the viewer of TCfW may display the message differently from the actual FAX output.

Content View Displaying Binary Attachments

The content text view (L_CONTENT_VIEW_TXT) of a message may also display binary attachments. Whether and how binary attachments show up in the content view is controlled by the new child object TS_COMMENT in the message store entry SET_ENTRY_MS.

If TS_COMMENT does not exist in SET_ENTRY_MS, or holds an empty string, the content view will not include binary attachments (compatible to previous releases).

If a non-empty TS_COMMENT is set in SET_ENTRY_MS prior to opening the content view SET_CONTENT_VIEWS, binary attachments in the message will represented by text blocks in the content view. These text blocks contain a single line which is generated by using the TS_COMMENT string from the message store entry as format string to print the file name and comment fields (TS_LONG_FILE_NAME and TS_COMMENT) of the binary attachment.

Example:

Setting TS_COMMENT in the message store entry to

“Attachment Name <%s> Comment <%s>”

will include an attachment with file name “const.doc” and comment “construction overview”

as text block holding the line

“Attachment Name <const.doc> Comment <construction overview>”

in the view.

Note: Displaying alternative content has priority over the replacement text described above. Only in case that there is no displayable alternative content the replacement text is shown for a binary attachment.

Customized Content View

The content text view (L_CONTENT_VIEW_TXT) of a message may be customized for different font sizes and page layouts. Whether and how the view is customized is controlled by the new child object SET_PAGE in the message store entry SET_ENTRY_MS.

If SET_PAGE does not exist in SET_ENTRY_MS, the content view will be done according to the page layout settings in the message (compatible to previous releases).

If a SET_PAGE is set in SET_ENTRY_MS prior to opening the content view SET_CONTENT_VIEWS, the view will be done according to the settings in this SET_PAGE, ignoring all settings in the message itself (which could be included in SET_PAGE and SET_BLK_TXT objects).

Currently the following child objects of SET_PAGE influence the content view:

INT_LINELEN	(default: 95 characters)
INT_BS_LF_OFFS	(default: 5 characters)
INT_LINES_PER_PAGE	(default: 59 lines)
INT_L_PITCH	(default: 236 Twips)
INT_TOP_MARG_TXT	(default: 1148 Twips)
INT_TOP_MARG_GRAPH	(default: 0 Twips)
INT_BOTTOM_MARG_TXT	(default: 1767 Twips)

If any of the above values is not contained in the SET_PAGE of the message store entry a program default value is used (same default as for regular content view).

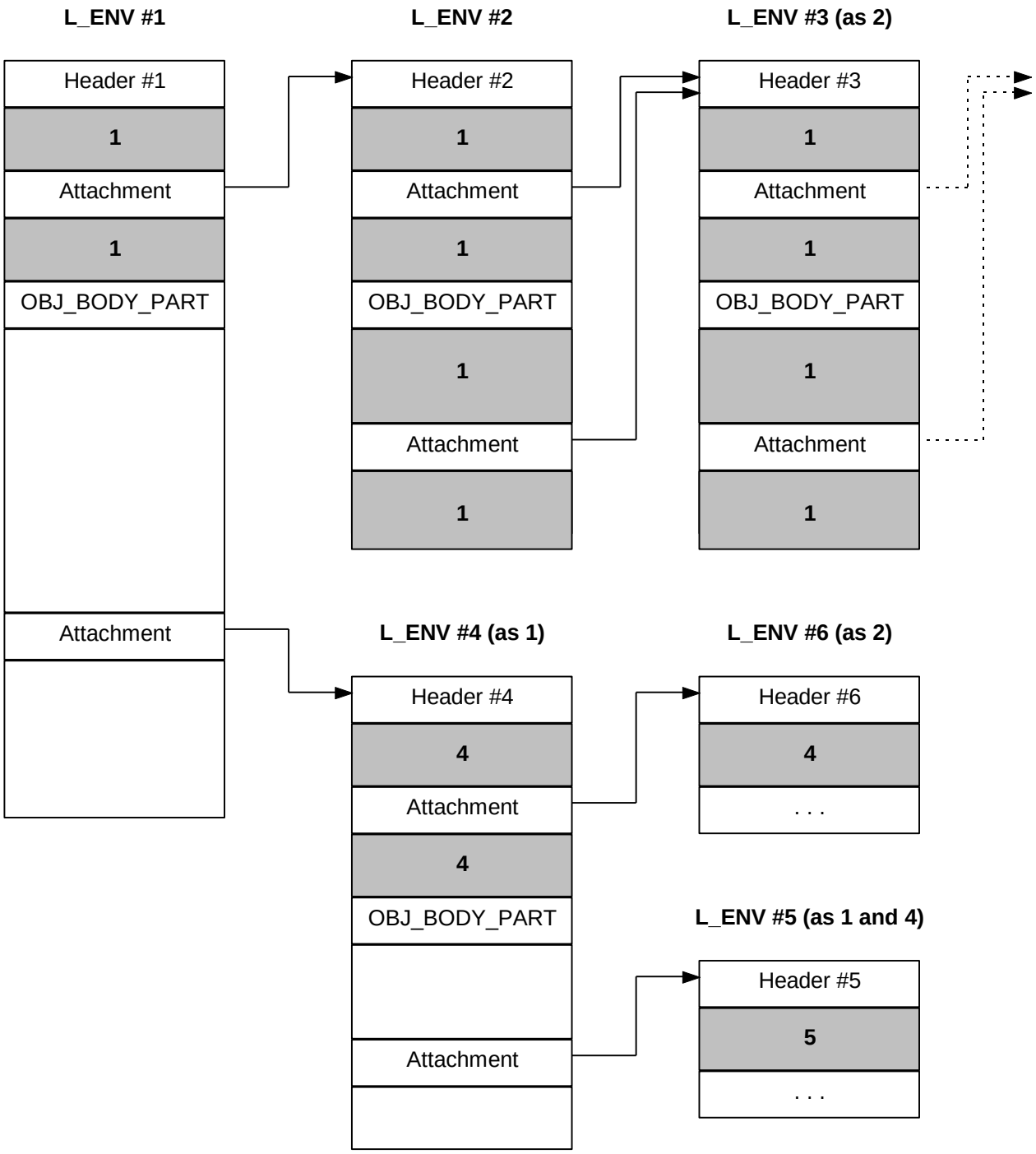
Note: The feature which allows to switch off line make up in the text view (by setting INT_LINELEN in SET_ENTRY_MS to LINELEN_UNLIMITED) has priority over the customized content view feature. If both features are used at the same time a customized view according to the SET_PAGE settings is produced but without doing any line make up.

Resolution of Cover Variables in Attachments:

Example:

The main envelope and attachments 4 and 5 are not part of a higher-level cover, so they use their own header for resolution.

Attachments 2,3 and 6 are part of a higher-level cover and use its header.



Variables are only resolved within the grey areas. The number in each grey area denotes the header that is used for resolution.

Variables from entry:

The following variables refer to the envelope's message store or mail entry.

TS_TOS_FOLDER	-		
TS_FILE_NAME	$FNam$	x)	File name
TS_TC_MSG_ID	$MsgId$	x)	TCOSS message Id
TS_REF	$ERef$		Subject field of the message
INT_NPAG	EPg		Number of pages of the message
INT_DOC_CLASS	-		
INT_FILE_SIZE	$Size$	x)	the file size of the message
TIME_CREATED	$Ctimef$	x)	File creation time (f = format ..0,1, 2, see below)
	$CTimef$	x)	File creation time (f = format ..0,1, 2, see below)
	$CDate$	x)	File creation date (f = format ..0,1, 2, see below)
TIME_REC_END	$RTimef$	x)	End of reception time (f = format ..0,1, 2, see below)
	$RDatef$	x)	End of reception date (f = format ..0,1, 2, see below)
UN_CONTENT			see header variables
TS_DOCUMENT_ERR	$RCode$	x)	Reception error code
	$RError1$	x)	English reception error text (from file tc01.err)
	$RError2$	x)	German reception error text (from file tc02.err)
INT_DEL_TYPE	-		
TS_RECIPIENT	-		
TS_RECIPIENT_INFO	$RcInf$	x)	Recipient info
TS_REC_INFO_ORIG	$RcInfOr$	x)	Originally specified recipient info
TS_ORIGINATOR	-		
TS_ORIGINATOR_INFO	$OrInf$	x)	Originator info
TS_ENV_NAME_POSTED	$ENam$		original message name
TIME_INTENDED	$ITimef$	x)	Intended sending time (f = format ..0,1, 2, see below)
	$IDatef$	x)	Intended sending date (f = format ..0,1, 2, see below)
TIME_ ACTION	$Date*f* $		Date and time of actual sending or of last retry
	$Time*f*$		(f = format ..0,1, 2, see below)
	$TZone1$		Offset from UTC in hh:mm, e.g. "+01:00"
	$TZone2$		Time zone name, e.g. "CET"
TS_DOC_NR	$Docnr$		serial sending number
INT_PRIORITY	-		
TS_LAST_MDA_ACTION	$Laction$	x)	last action of delivery agent (sending error code)
	$SError1$	x)	English sending error text (from file tc01.err)
	$SError2$	x)	German sending error text (from file tc02.err)
TS_LAST_MDA_NOTE	$Lnote$	x)	last note of delivery agent
INT_OPEN_RETRIES	-		
INT_SUSP_DUPL	P or $Pstring$		Inserts "possible duplicate message" or "string" if possible duplicate transmission occurred
TS_COST_CENTER	$Cctr$	x)	the cost center
TS_COST	$Cost$	x)	the cost of sending
TS_DURATION	$SDur$	x)	Sending duration
TS_CHANNEL	$SChan$	x)	Sending channel
TS_SERV_ID	$SMedia$	x)	Sending Media Server
TS_LOCALIZED_ADDR	$LocAdd$	x)	Localized Address
TS_NORMALIZED_ADDR	-		
TS_NODELIST	$Nodes$	x)	The list of nodes the message was routed
INT_CIF_NR, INT_CIF_ID	$Correl$	x)	The correlation information, 18 characters =

mail entry number (8) + unique ID (10)

INT_EVENT_TYPES -

INT_OPTIONS -

x): Only available via TCSI. (not supported by TAM.)

format: 0=extended format, see next page; 1=YY-MM-DD and hh:mm:ss; 2= DD-OCT-YYYY; hh:mm:ss

Extended cover variable formats for date and time in TCSI32.DLL

Extended cover variable formats to specify date and time are supported by the 32-bit version:

$Date0nfffffffffffffffffffffff$ or $Date0nfffffffffffffffffff_____$
$Time0nfffffffffffffffffffffff$ or $Time0nfffffffffffffffffff_____$

n … Windows locale identifier (LCID)
fffffffffffffffffff … format string

Examples: „$Date09dddd, yyyy-MMMM-dd$“ is resolved to „Tuesday, 1998-July-21“
„$Time09hh:mm:ss tt$“ is resolved to „09:58:20 AM“

language	locale identifier	text block code page	NT ANSI code page
Basque	45	0	1252
Catalan	3	0	1252
Czech	5	1	1250
Danish	6	0	1252
Dutch	19	0	1252
English	9	0	1252
Finnish	11	0	1252
French	12	0	1252
German	7	0	1252
Hungarian	14	1	1250
Icelandic	15	0	1252
Indonesian	33	0	1252
Italian	16	0	1252
Japanese	17	932	932
Norwegian	20	0	1252
Polish	21	1	1250
Portuguese	22	0	1252
Romanian	24	1	1250
Slovak	27	1	1250
Slovenian	36	1	1250
Spanish	10	0	1252
Swedish	29	0	1252

This feature is only supported for the combinations of Windows and text block code pages listed above. A Japanese edition of Windows is required for locale ID 17, all others work on both English and Japanese versions. Using locale

identifiers not listed above or with other combinations of text block and Windows ANSI code page will produce either an empty string or an incorrectly converted string.

Format controls for date:
The letters must be in uppercase or lowercase as shown in the table (for example, "MM" not "mm").

d	Day of month as digits with no leading zero for single-digit days.
dd	Day of month as digits with leading zero for single-digit days.
ddd	Day of week as a three-letter abbreviation in the selected language
dddd	Day of week as its full name in the selected language
M	Month as digits with no leading zero for single-digit months.
MM	Month as digits with leading zero for single-digit months.
MMM	Month as a three-letter abbreviation in the selected language
MMMM	Month as its full name in the selected language
y	Year as last two digits, but with no leading zero for years less than 10.
yy	Year as last two digits, but with leading zero for years less than 10.
yyyy	Year represented by full four digits.
gg	Period/era string. This element is ignored if the date to be formatted does not have an associated era or period string

Format controls for time:
The letters must be in uppercase or lowercase as shown in the table (for example, "ss" not "SS").

h	Hours with no leading zero for single-digit hours; 12-hour clock
hh	Hours with leading zero for single-digit hours; 12-hour clock
H	Hours with no leading zero for single-digit hours; 24-hour clock
HH	Hours with leading zero for single-digit hours; 24-hour clock
m	Minutes with no leading zero for single-digit minutes
mm	Minutes with leading zero for single-digit minutes
s	Seconds with no leading zero for single-digit seconds
ss	Seconds with leading zero for single-digit seconds
t	One character time marker string, such as A or P (depends on selected language)
tt	Multi-character time marker string, such as AM or PM (depends on selected language)

The 16-bit version will evaluate the extended cover variables like $Date1$ and $Time1$, not using the locale identifier and the format string.

Header Variables:

Recipient:

The following variables refer to the recipient list in the header of the envelope.

When used outside of the list-brackets (see below) the variables refer to the active recipient.
(The active recipient is defined by INT_ER_RECIPIENT in the mail entry. If this object does not exist the first recipient in the L_RECIPIENTS in the header is taken as the active one.)

Object	Variable	Description
TS_RECP_ID	$SNam$	Short name
TS_COMPANY	$Comp$	Company
TS_DEPTM	$Dept$	Department
TS_FULLNAME	$Name$	Receiver name
TS_SALUTE	Sal	Salutation
TS_FREETEXT	Txt	Free text
TS_CORREL_n	$Corr1$ ..$Corr6$	Correlation fields
L_FULL_ADDR	$Add*n*xxxx$	Address part-n of Service xxxx (n=0 all parts in one string)
		The address list is searched for a matching service in TS_SERVICE. Part # refers to the position of the child of the address as in the description of the address objects
		(1=topmost child, higher numbers going down the list). Only the first alternate of each service is used
		E.g.: $Add0FAX$
		$Add1TELEX$ $Add2TELEX$
		$Add1LETTER$ .. $Add6Letter
	$Addn$, n = ‘0’ ..’9’	Address part n of actually used alternative address
		n = 1 … first child object, n = 2 .. 2nd child object, etc
		n = 0 … all child objects in one comma separated list
	Add	Service of actually used alternative address

Originator:

The following variables refer to the originator (SET_ENTRY_RS_ORIGINATOR) in the header of the envelope.

Object	Variable	Description
TS_RECP_ID	$USNam$	Short name of the user sending
TS_COMPANY	$UComp$	Company
TS_DEPTM	$UDept$	Department
TS_FULLNAME	$UName$	Name of the user
TS_SALUTE	$USal$	Salutation
TS_FREETEXT	$UTxt$	Free text
TS_CORREL_n	$UCorr1$ ..$UCorr6$	User correlation fields

L_FULL_ADDR	$UAdd*n*xxxx$	Address part-n of Service xxxx (n=0 all parts in one string) The address list is searched for a matching service in TS_SERVICE (match is case sensitive). Part # refers to the position of the child of the address as in the description of the address objects (1=topmost child, higher numbers going down the list). Only the first alternate of each service is used E.g.: $UAdd0FAX$ $UAdd1TELEX$ $UAdd2TELEX$ $UAdd1LETTER$ .. $UAdd6LETTER

Creation of To and cc lists (distribution list):

$To< > or $to< > produce a list of all “to” receivers

$Cc< > or $cc< > produce a list of all “cc” receivers

Note: All variables as shown above can be used within the brackets. Recipient variables within brackets refer to recipient n in the list. n is counted up for each line of the list. The $To and $Cc line must fit on one line.

Additional cover variables to be used within lists:

$L1anystring$ ... converted to “anystring” in the first entry of a recipient list, “” in all other entries
$L2anystring$ ... converted to “” in the first entry of a recipient list, “anystring” in all other entries
The purpose of these two variables is to add a descriptive string like “To: “ to the first recipient list entry and a different string, e.g. spaces to keep the indent, to the following list entries.

There is no difference between $to< > and $To< > as long as there are no distribution lists involved.

If the keywords are specified with a capital first letter ($To< > or $Cc< >), distribution lists are shown as a single line with the $Name$, $Comp$ etc. fields taken from the distribution list.

If the keywords are lower case ($to< > or $cc< >), all actual recipients of a distribution list are listed with the $Name$, $Comp$ etc. fields taken from each individual recipient.

Example:

Text block in cover part:

```
------------------------------------------------------------
Attn: $Name_____________________$Date: $Date1_$Time: $Time1$
Number of pages including cover sheet: $EPg$
------------------------------------------------------------
To:          $to<$Name$, $Comp$, $Dept$>

cc:          $cc<$Name$, $Comp$, $Dept$>

From:  $UComp$, $UDept$, $UName$

Subject:     $ERef$
```

Resulting view:

```
------------------------------------------------------------
Attn: Hans Meyer              Date: 930530  Time: 15:45:24
Number of pages including cover sheet: 4
------------------------------------------------------------
To:          John Smith, ACME Ltd., Export

cc:          Hans Meyer, Export International PLC., Contracts
             Peter Best, Best Company & Co., Sales

From:  Best Company & Co., Marketing, Jonny Fast

Subject:     The Deal of the Year

Dear John,

I have great news for you. ..................
```

Rich text Cover Sheets

An RTF Coversheet may include all coversheet variables that can be included in a standard coversheet. The only exception is that the syntax of $to< > and $cc < > will change a little bit. A message with a RTF cover sheet may previewed with the normal TCfW preview function.

Identification of rich text cover sheets

If the TS_FILE_NAME string of a SET_ATT_OBJ is of the form ”*.rtf” a BLK_BINARY residing in this SET_ATT_OBJ is assumed as consisting of Richtext. In this way a RTF coversheet is represented in TCSI as a SET_ATT_OBJ which lies before the OBJ_BODY_PART in a L_ENV_CONT and has its TS_FILE_NAME set to a string of the form “*.rtf”. Its Richtext is located in the SET_ATT_OBJ’s BLK_BINARY.

Rich text cover sheets as overlays

Any RTF attachment is handled as image overlay if the TS_COMMENT field of SET_ATT_OBJ holds overlay parameters in the format “++FX1 x-pos, y-pos, startpage, pages”.

Changed syntax of “To” and “cc” lists in RTF cover sheets

In a line where $To (or $to) is used, $To should be the first sequence of characters in this line. In this way $To marks the beginning of a line. The closing square bracket > may be on a different line following the line containing the $To<. It actually has to be on the next line if one wants to have a separate line for each recipient.

The $To<...> command in RTF covers is interpreted in the following way: $To< and > are removed. The remaining rich text between these markers (possibly containing line feeds) is copied as many times as there are recipients, or removed altogether if the list is empty. All cover sheets variables are resolved, including the newly defined $L1...$ and $L2...$ variables.

In contrast to RTF covers recipient lists of standard cover sheets are limited to one line, the closing bracket has to be on the same line, and the text before the $To< variable is set to spaces in all list entries except the first.

The processing of recipient lists in RTF cover sheets is not line oriented, so the “Line feed” sequence has to be included by putting the closing bracket on the next line if necessary. The text before the $To< variable is not set to spaces for extra entries like in the standard cover, so one uses the $L1..$ and $L2..$ variables instead.

Example of recipient list in an RTF cover sheet:

```
$To<$L1To:$$L2   $  $Name$, $Comp$, $Dept$
>$cc<$L1cc:$$L2   $  $Name$, $Comp$, $Dept$
>
```

Resulting view:

To: John Smith, ACME Ltd., Export
cc: Hans Meyer, Export International PLC., Contracts
Peter Best, Best Company & Co, Sales

Restrictions to use of cover sheet variables in RTF covers

The variables in RTF cover sheets must not be interrupted by RTF command sequences. This could happen if the style changes within the variable as in the following example which is partially bold:

“$name$” would not be a valid variable and therefore not resolved because in the RTF stream it would come out like “$n\b ame$”.

Cover sheet variables may be given in any style or font, which is then applied to the replacement string, but the variable has to be uniform.

Page Format

It is only possible to use Rich Text documents in portrait format in your coversheet. There is no support for landscape format.

If you want to use a Rich Text document in portrait format your document should have the width of A4 format (that is 210 mm). If the width of your document is smaller than 210 mm it is automatically widened to 210 mm when it is attached to your coversheet. On the other hand your document does not have to have the height of A4 format (= 297 mm). If the height of the Rich Text document is smaller it is attached to your coversheet with no changes to its height.

Don't use Rich Text documents which are wider than 210 mm. Loss of text or word- line- or page breaks in the preview/fax which differ from your original RTF document will occur.

3.7.2.9 L_CONTENT_VIEW_IMG

Not implemented yet.

Holds an image view of an envelope. The list elements are just page breaks and image blocks. Each image block holds a page of the document as seen on an image device (FAX).

The view is always on the contents of the envelope at the time this object was opened. When the envelopes contents are changed, the view has to be closed and opened again to reflect the changes.

Resolution of image

List elements:	attributes	default
SET_PAGE	-	-

A page break with an optional format definition.

SET_BLK_IMG	-	-

An image block.

OBJ_BODY_PART	-	-

Marks the beginning of the body part.

3.7.2.10 L_IDX_PGNR

Child of SET_CONTENT_VIEW.

After opening SET_CONTENT_VIEW_TXT this object contains a list of integers with one entry for each element of the object L_ENV_CONT containing the page number.

List elements:	attributes	default
INT_PG_NUM	R/O	-

The page number of the corresponding object in L_ENV_CONT.

3.7.2.11 L_TEXTFILTER

Child of SET_FILTER_ARCHIVE.

Holds a list of attributes for searching the TC/Archive. A logical 'AND' is applied to the individual attributes in the list, i.e. only messages which match all specified attributes are included in the search result. A different order of attributes in the

list does not change the search result, but may influence the search performance. For best performance put the more “restrictive” attributes at the beginning of the list.

An empty text filter list puts no restriction on the archive search, all messages which meet the other filter values in SET_FILTER_ARCHIVE (upper and lower bound for send / receive time) appear in the search result.

List elements:	attributes	default
SET_ATTRIBUTE	-	-

A search attribute (see separate description).

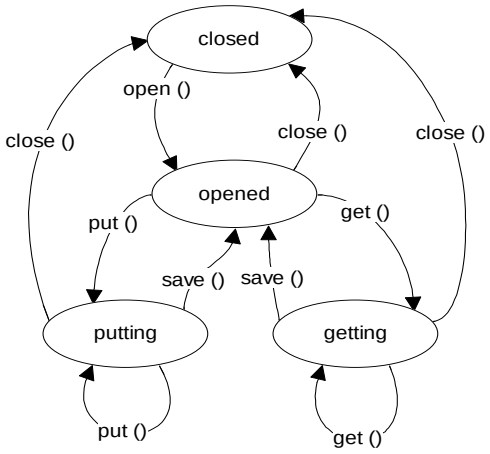
3.8 BLOCK

A block holds a block of data. The data can be read or written by a sequence of get or put operations.

Possible sequence of put / get operations:

The diagram is an additional definition to the general object states given for class OBJECT.

The first put after open or save starts at the beginning. The same is true for get operations without index.



Note:

When a class inherits not only from BLOCK but also, for example, from SET the child objects may be accessed independently of the block put / get operations.

Get / put delivers / accepts the content of the block in a sequence of (buffer sized) pieces.

Restriction: any get / set operation must transfer the complete object in a closed sequence.

The binary put / get functions may be used on objects of all types that are not anchored. The format of the binary stream is as defined in the object handler low (OHL) interface
Note: For binary streaming the header (type, name and length) of the topmost object is not part of the stream.
Restriction: Object access to child objects may not be intermixed with the ASCII put / get functions. (Child objects can not be opened while the object is streamed.)

Take great care when using stream access to put objects since **no** format check (on type range and value) is performed. Objects that are injected into the system in this way may cause havoc at remote places and times!

ohh_b_bin_put (h_blk_binary, pbuffer, size)

Writes size bytes from buffer to object.

errors: none

ohh_b_bin_get (h_blk_binary, pbuffer, psize, bufsize)

Reads maximum bufsize bytes from object to buffer. This function may return any number of bytes up to bufsize.

errors: INF_END_REACHED the end of the object has been reached

3.8.2 Random Binary Access to Block Objects

These functions provide random read and write access to all types of blocks (BLK_TXT, BLK_TCI and BLK_BINARY).

TCSI_RET ohh_b_get(HOBJ h_blk, LPCHAR pbuffer, LPINT4 psize, INT4 bufsize, INT4 position)

The ohh_b_get function copies *bufsize* bytes from offset *position* of object *h_blk* into the buffer pointed to by *pbuffer*. The number of bytes copied to the buffer is returned in the integer pointed to by *psize.* (It may be less than *bufsize* if the end of the block is reached). An error code (ERR_END_OF_BLOCK) will be returned if *position* >= blocksize.

TCSI_RET ohh_b_put(HOBJ h_blk, LPCHAR pbuffer, INT4 size, INT4 position)

The ohh_b_put function writes *size* bytes from the buffer pointed to by *pbuffer* into the block object *h_blk* at offset *position*. If the specified *position* is AFTER_END or if it is beyond the current end of the block, the data will be appended at the end of the block, no error or warning is returned for this case.

Performance hint: Don't build a large block by writing a lot of small pieces. The memory is allocated and re-allocated as the block keeps growing, so it is more efficient to write it in large chunks.

The POP3 server implementation needs a way to determine the size of a block object (BLK_BINARY, BLK_TCI, BLK_TXT) without opening the object. For this purpose the ohh_b_size function is implemented:

ohh_b_size (HOBJ hParent, TCSI_ID nID, LPINT4 lpSize)

This function returns the amount of data bytes stored in the specified block. The return value is either OK or a TCSI error code (e.g. ERR_WRONG_TYPE if child object is no block).

Parameters:

hParent	…	handle to parent object
nID	…	id of child object
lpSize	…	pointer to INT4 variable receiving the result

3.8.3 ASCII Stream Access to Block Objects

ASCII representation:

For objects of each class an ASCII representation exists:

```
object : type [/ name ] = value
```

type: the type as given in TCSI.H

name: the name of the object within its container (as defined in TCSI.H)

Exists only if: - the object is child of a SET

- the name of the object (within the SET) is different from its type

value: the value of the object. See all base classes for definition.

White space and carriage returns may be inserted at any position.

The following two functions allow to get / put the ASCII stream:

The ASCII put / get functions may be used on objects of all types that are not anchored.

For ASCII streaming the topmost object (the object on which ascii_get() is performed) is part of the stream. On ascii_put() the topmost object is ignored.

Restriction: Object access to child objects may not be intermixed with the ASCII put / get functions. (Child objects can not be opened while the object is streamed.)

ohh_b_ascii_get (h_obj, pszbuffer)

Gets a line (zero terminated string) from the objects ASCII representation.
Maximum size is fixed to 256 characters (including terminating zero). The lines of text are delivered sequentially from the beginning after opening of the object.

errors: INF_END_REACHED the end of the object has been reached

ohh_b_ascii_get_ex (h_obj, pszbuffer, bufsize)

Like ohh_b_ascii_get, allows to specify the buffer size. Minimum buffer size is 256, maximum 2048 bytes.

ohh_b_ascii_put (h_obj, psz_string)

Puts one text line to the object. The line must follow the rules for the ASCII stream form of objects.
Maximum size is fixed to 256 characters (including terminating zero).

The original strings from object contents are delivered in TCOS code, within quotation marks (" ") without terminating zero. Original strings are never made up.

When an error is returned the object is automatically forgotten.

errors: INF_END_REACHED the current line held the end of the object

ERR_WRONG_FORMAT an error was detected in the ASCII stream

3.8.4 Optimized Stream Read Access to Block Objects

Normal binary access to block objects loads the complete block into memory when it is opened. The idea of the stream access is that large block objects are read directly from the file on the TCOSS server and not buffered in memory by TCSI. Small block objects are still loaded completely.

Stream access works for binary, text and image blocks (objects of type BLK_BINARY, BLK_TXT or BLK_TCI) contained in permanent store objects of a TCOSS server under the following conditions:

1) The optimization parameters **Xlevel**, **Xblock**, **Xtotal** and **Xmode** are set correctly

2) The data is read using the **ohh_c_stream_get**() function

Every call of ohh_c_stream_get() will establish a server connection, activate the server session, read the data directly from the TCOSS file system and close the connection. In case of very small block objects, which have been loaded into memory during the initial open of the message, the data is read from local memory. Buffering of the data and reading ahead has to be done by the application if a continuous flow of data is required.

Ad 1) Optimization parameters to be set in the “TCSI” registry section of the application:

Xlevel (REG_DWORD	not set or set to 0, default value is 0 = all levels
Xblock (REG_DWORD)	block size in KB, only blocks exceeding this size are optimized
	default value is 2 (32-bit version), 0 = unlimited size
Xtotal (REG_DWORD)	total stream size in KB, must be exceeded before optimization starts
	default value is 50 (32-bit version), 0 = unlimited size
Xmode (REG_DWORD)	set to 1 = block optimization on, default value is 1

With the 32-bit version, stream access works well with the default values of all parameters, Xblock and Xtotal may be adapted for further optimization. The optimized stream access does not work with the default settings of the 16-Bit DLL (Xblock=0 and Xtotal=0, both values have to be changed in the INI-File to e.g. Xblock=2 and Xtotal=50).

If the optimization parameters are set incorrectly, the access with ohh_c_stream_get() still works, but it is not optimized and the complete block is loaded into memory in all cases.

Ad 2) Function ohh_c_stream_get

The function ohh_c_stream_get () provides optimized stream access:

ohh_c_stream_get(h_container, id, pbuffer, psize, bufsize, position)

The function copies *bufsize* bytes from offset *position* of child object *id* into the buffer pointed to by *pbuffer*. The number of bytes copied to the buffer is returned in the integer pointed to by *psize.* (It may be less than *bufsize* if the end of the block is reached). An error code (ERR_END_OF_BLOCK) will be returned if *position* >= blocksize.

It may be used to access child objects of types BLK_BINARY, BLK_TXT or BLK_TCI only.

The ohh_b_size function may be used to get the size of the block object.

Note: The optimized stream access feature requires TCOSS rel. 7.29.00. With older TCOSS releases the ohh_c_stream_get function works if the block has been loaded into memory already (no stream optimization, e.g. when using the recommended setting of Xlevel, Xblock etc. for release 7.29.00). Error 620 (ERR_UNKNOWN_FUNCTION) is returned if the block happens to be stored as a file pointer (e.g. when using Xlevel =3 and the block is found at level 4).

3.8.5 BLK_BINARY

A block of binary data.

3.8.6 BLK_TCI

A block of TCI data.

3.8.7 BLK_HUF

A block of huffman data. (not used)

3.8.8 BLK_G4

A block of group 4 coded data. (not used)

3.8.9 BLK_TXT

Concepts:

A text block holds a number of text lines.

ohh_b_txt_get (h_blk_txt, pbuffer, psize, bufsize, index, position)

Reads a text line (index) as zero-terminated string from the text block.
Via psize the actual length (not including the terminating zero) of the line is returned.
When the line is longer than the buffer size the line can be read in segments, each starting at an other *position*.
The first character of the line is at position 0.

The first line in the block has index 0.
The first character of the line is the TCOS format control character [9].

When the rest of the line starting with the specified position does not fit in the supplied buffer, the last buffer position is set to zero (buffer holds a zero terminated string) and WRN_TRUNCATED is returned. To read the next segment you add (bufsize-1) to position.

errors:		
	WRN_TRUNCATED	the line has been truncated
	ERR_END_OF_BLOCK	the index is out of block

ohh_b_txt_put (h_blk_txt, psz_string)

[9] see TCOS manual 5.20 or higher for reference

Appends a zero-terminated string as text line to the text block. The maximum line length is limited only by the calling function.

The first character of the line must be the TCOS format control character.

errors: none

```
TCSI_RET ohh_b_txt_put_ex (HOBJ hobj,  /* in: handle to text block    */
                     CONST CHAR *pstr, /* in: string in text codepage */
                           INT4 len)   /* in: string length          */
```

The new ohh_b_txt_put_ex() function allows to append a text line to a text block in several parts. The first character of the first part of the line must be the TCOSS format control character. The last part of the line is zero-terminated and the specified length includes the terminating zero, the other parts are not zero-terminated.

Unicode Text Functions

The functions ohh_b_txt_getW() and ohh_b_txt_putW() will access text within text blocks using wide character (16-bit) Unicode strings in UTF-16 encoding. The text block's code page (either 0 or TCOSS system code page) has to be set before writing any text to the block, the default code page of a text block is TCOSS 0.

```
TCSI_RET ohh_b_txt_getW(
         HOBJ hobj,        /* in: handle to text block   */
         INT4 index,       /* in: line index 0,1,2,..    */
         INT4 position,    /* in: internal position in line, 0 = whole line */
         WORD2 *pstr,      /* out: UTF-16 string          */
         INT4 cchstrmax,   /* in: max. Unicode string length incl. term. 00
                              in 2-byte units   */
                   LPINT4 pcchstr,  /* out if != NULL: length of string excl. term. 00
                                       in 2-byte units */
         LPINT4 pposition,/* out if != NULL: internal position of next part
                                       of line          */
         INT *pformat)     /* out if != NULL: format control character */
```

The ohh_b_txt_getW function allows to read lines which are longer than the buffer size in several parts. To read the first part specify position = 0, to read the next part set position to the value returned in *pposition by the previous call. Return value WRN_TRUNCATED indicates that reading of the line in parts is not yet complete. The TCOSS format control character is not part of the returned wide character string, it is returned in a separate parameter "pformat" which is set only when the first part of the line is read.

```
TCSI_RET ohh_b_txt_putW (HOBJ hobj,   /* in: handle to text block   */
                    CONST WORD2 *pstr) /* in: UTF-16 string, 00-term. */
```

The ohh_b_txt_putW function appends a zero-terminated string as text line to the text block. It does expose the TCOSS format control character on the text line interface.

The ohh_b_txt_putW_ex function allows to write a text line in several parts and to choose the TCOSS format control character.

```
TCSI_RET ohh_b_txt_putW_ex (HOBJ hobj,   /* in: handle to text block      */
                            INT format,  /* in: format control character  */
                      CONST WORD2 *pstr, /* in: UTF-16 character array    */
                            INT4 cchstr, /* in: number of codes in pstr   */
                            BOOL is_last)/* in: last or only part of line */
```

Specify the format control character in the first or only part of a line, set format = 0 in a continuation part of a line. The UTF-16 character data in "pstr" is not zero-terminated, the character count "cchstr" gives the number of 16-bit codes. Specify the "is_last" flag as TRUE in the first or only part of a line, set it to FALSE if at least one more part follows.

3.8.9.2 BLK_TXT_SEQ

A text block that can only be accessed sequentially.

***ohh_b_txt_get* (h_set_blk_img, pszbuffer, index)**

Note: may be removed in future versions

index is ignored. The lines of text are delivered sequentially from the beginning after opening of the object.
After OK_MORE is returned the next calling of the function will give the next section of the line regardless of index.

3.9 General functions

3.9.1 Direct Server Access

The SET_XML_SERVER child of the SET_SERVER_SESSION object provides direct streaming access to the connected TCOSS or TC/Archive server's OHL (object handler low) interface. The following TCSI functions may be applied to the SET_XML_SERVER object:

- ohh_c_open(h_server_session, SET_XML_SERVER, &h_server, &type)
  Connects to the server and activates the session on the server. The network connection to the server stays open.

- ohh_b_sax_setcodepage(h_server, codepage)
  Sets the default TCOSS codepage.

- ohh_o_sax_put(h_server, event, databuffer, datalength)
  Is used to construct commands to the server, see detailed description below.

- ohh_o_save(h_server, FALSE)
  Flushes the output buffers and switches from command sending to response reception.

- ohh_o_sax_get(h_server, &event, saxbuf, &saxlen)
  Is used to read the server's response, see detailed description below.

- ohh_o_close(h_server, FALSE) or ohh_o_forget(h_server)
  Is used to close or abort the command – response sequence. The network connection to the server is closed. Close and forget have the same functionality.

3.9.2 SAX Interface Functions

The Simple API for XML (SAX) 2.0 offers a streaming model that can be used to read or write XML documents. The following two groups of TCSI functions allow to implement a SAX interface in a simple C++ wrapper class.

Functions to build up or read objects in memory (to be applied to objects of any type):
ohh_b_sax_setcodepage
ohh_b_sax_getW
ohh_b_sax_putW

Functions to send a command or read a response from the server (applied to SET_XML_SERVER):
ohh_b_sax_setcodepage
ohh_o_sax_get
ohh_o_sax_put

3.9.2.1 Set Default Code Page for SAX Get / Put Functions

INT **ohh_b_sax_setcodepage**	(HOBJ hobj,	/* in: handle to TCSI object	*/
	INT4 codepage)	/* in: TCOSS code page	*/

This function is usually called right before the first ohh_b_sax_get or put function. It sets the TCOSS code page which is used for the conversion of all TCSI string objects and as a default for text blocks without code page information.

If the TCOSS code page is not set explicitly it defaults to 0.

3.9.2.2 SAX Get Function

INT **ohh_b_sax_getW**	(HOBJ hobj,	/* in: handle to TCSI object	*/
	INT *p_event	/* out: SAX event	*/
	wchar_t *pstr	/* out: string	*/
	INT4 *cchstr)	/* out: length of string	*/

Allows to pull SAX events from a TCSI object in a way analogous to the ohh_b_ascii_get function. The root object is part of the event stream.

The following events are returned in parameter *p_event accompanied by the appropriate string data:

OHH_SAX_STARTELEMENT ... start tag

OHH_SAX_CHARACTERS ... element content

OHH_SAX_ENDELEMENT ... end tag

A return value of INF_END_REACHED (without any event or data) indicates that the event stream is complete. Any return value other than OK or INF_END_REACHED indicates an error. In this case all internally opened handles of child objects are closed, the handle passed to the function stays valid.

The data buffer provided by the caller must have a minimum length of OHH_SAX_STRLEN (16-bit codes), i.e. a minimum size of 2*OHH_SAX_STRLEN bytes.

All string data returned is in 2-byte Unicode. XML markup delimiters like ampersand or angle brackets have to be escaped outside e.g. by using predefined character entities.

The binary get trace (flag 0x10 in DebugLevel) may be activated to get an XML-style trace of the returned data before the Unicode conversion. Trace snippet:

```
14:46:39.656 (49/5f) GET:<set_entry_ms>
14:46:39.671 (49/5f) GET: <ts_file_name>Correlation</ts_file_name>
14:46:39.671 (49/5f) GET: <un_content.l_env_cont>
14:46:39.671 (49/5f) GET:  <set_header>
14:46:39.687 (49/5f) GET:   <ts_ref>subject of message</ts_ref>
14:46:39.687 (49/5f) GET:   <int_priority_to>50</int_priority_to>
14:46:39.687 (49/5f) GET:   <int_priority_cc>50</int_priority_cc>
14:46:39.703 (49/5f) GET:   <int_termination>243</int_termination>
```

3.9.2.3 SAX Put Function

INT **ohh_b_sax_putW**	(HOBJ hobj,	/* in: handle to TCSI object	*/
	INT event	/* in: SAX event	*/
	wchar_t *pstr	/* in: string	*/
	INT4 *cchstr)	/* in: length of string	*/

Allows to construct a TCSI object from SAX events in a way analogous to the ohh_b_ascii_put function.

The root object's start tag is part of the expected event stream, but not evaluated except for format extensions like ".tctext" or ".utf-8". If no format extension is used an empty string suffices as root element start tag, of course the correct start tag is also accepted. The root object's end tag is optional, unless the root object is a block e.g. a text,

image or binary block. If the root object's end tag is included the return value of this last call is INF_END_REACHED in the ok case.

The following events may be entered in parameter event accompanied by the appropriate string data:

OHH_SAX_STARTELEMENT	... start tag
OHH_SAX_CHARACTERS	... element content
OHH_SAX_ENDELEMENT	... end tag (the actual string is not evaluated)

The element content may be entered with a single or with several OHH_SAX_CHARACTERS events.

String data is expected in 2-byte Unicode with all XML entities already resolved. Element content outside simple elements (e.g. white space in container objects) is skipped with an OK return value.

Any return value other than OK or INF_END_REACHED indicates an error (WRN_TRUNCATED is also seen as error). In this case all internally opened handles have been closed, discarding all partially built child objects. The handle passed to the function stays valid and should be closed outside by calling ohh_o_forget(). It is not possible to continue building the TCSI object after an error has been reported.

The reporting of format errors in the element content may be delayed to the end tag event because the content or part of it may be buffered and processed later with the end tag event.

To intentionally discontinue constructing a TCSI object one should call ohh_b_sax_putW() with an invalid event, e.g. 0, to produce an error return value, and then call ohh_o_forget() on the handle.

The binary put trace (flag 0x04 in DebugLevel) may be activated to get a trace of the object being built up in memory.

3.9.2.4 SAX Put Function for Direct Server Access

INT **ohh_o_sax_put**	(HOBJ hobj,	/* in: handle to SET_XML_SERVER	*/
	INT event	/* in: SAX event	*/
	const char *pstr,	/* in: UTF-8 string	*/
	INT4 strlen)	/* in: length of string	*/

Allows to construct a TCSI command stream from SAX events. Note that the binary element content event is a proprietary extension of the SAX interface.

The following events may be entered in parameter event accompanied by the appropriate string data:

OHH_SAX_STARTELEMENT	... start tag
OHH_SAX_CHARACTERS	... element content
OHH_SAX_BINARY	... binary element content (optionally for BLK_BINARY content)
OHH_SAX_ENDELEMENT	... end tag (the actual string is not evaluated)

The element content may be entered with a single or with several OHH_SAX_CHARACTERS events. The content of binary blocks may be entered with OHH_SAX_CHARACTERS events giving the data in Base64 encoding or alternatively with OHH_SAX_BINARY events giving binary data. It is not allowed to mix OHH_SAX_CHARACTERS and OHH_SAX_BINARY events within the same binary block, use one option or the other to build up the complete block.

String data is expected in UTF-8 encoding with all XML entities already resolved. Element content outside simple elements (e.g. white space in container objects) is skipped with an OK return value.

Any return value other than OK indicates an error. The reporting of format errors in the element content may be delayed to the end tag event because the content or part of it may be buffered and processed later with the end tag event.

The ohh_o_sax_put() function does a relaxed validation of the generated command stream:

- All TCSI objects and the separately defined OHL function codes are allowed as top level elements.
- Only the child objects defined in TCSI are allowed within a particular parent.
- String length limits are not checked.

3.9.2.5 SAX Get Function for Direct Server Access

```
INT ohh_o_sax_get   (HOBJ hobj,              /* in: handle to SET_XML_SERVER   */
                     INT *p_event_flags,     /* out: SAX event + flags         */
                     char *pstr,             /* out: UTF-8 string or binary data */
                     INT4 *pstrlen)          /* out: length of string or binary data */
```

Allows to pull SAX events from a TCSI response stream. Note that the binary element content event is a proprietary extension of the SAX interface.

A combination of SAX event and flags is returned in parameter *p_event_flags. The get the event value without flags calculate (*p_event_flags & OHH_SAX_EVENTMASK).

The following events are returned accompanied by the appropriate string data:

OHH_SAX_STARTELEMENT	... start tag
OHH_SAX_CHARACTERS	... element content
OHH_SAX_BINARY	... binary element content (always for BLK_BINARY content)
OHH_SAX_ENDELEMENT	... end tag

The following flags may be returned in *p_event_flags (together with a start element event):

OHH_SAX_CHILD_OF_SET	… start tag is child of TCSI SET (as opposed to a list)
OHH_SAX_LAST_CHILD	… start tag is last child of TCSI SET (last in stream)

A return value of INF_END_REACHED (without any event or data) indicates that the event stream is complete. Any return value other than OK or INF_END_REACHED indicates an error.

The data buffer provided by the caller must have a minimum length of OHH_SAX_UTF8LEN (3*2048) bytes. The buffer length value of 3*2048 bytes is derived from the maximum string length in TCSI of 2048 bytes and the fact that one byte may be expanded to three bytes in the TCOSS codepage to UTF-8 conversion.

All string data returned is in UTF-8 encoding. XML markup delimiters like ampersand or angle brackets have to be escaped outside e.g. by using predefined character entities.

The content of binary blocks is returned with OHH_SAX_BINARY events accompanied by the binary data without any conversion. If the data is to be passed on to a standard XML parser which does not support a binary content event, the binary data has to be Base64 encoded and passed on with a “characters” event.

The ohh_o_sax_get() function does a relaxed validation of the received response stream:

- All TCSI objects and the “type_ok” and “type_err” objects are allowed as top level elements.
- Only the child objects defined in TCSI are allowed within a particular parent, unknown child objects are skipped without reporting an error (only a trace line is written).

- String length limits are not checked.

3.9.2.6 Restrictions of SAX Functions for Direct Server Access

- The ohh_o_sax_put function uses a 2-byte counter for the generation of list ids. This means that lists are restricted to a maximum number of 65535 child objects. This limit of 65535 child exists in all TCSI releases and also applies to regular TCSI objects.
- Block objects (text, image and binary blocks) are buffered in memory before being sent to the server, because the TCSI stream format does not allow block objects of unknown length.

3.9.3 Interface Version

ohh_version (psz_requested_version, psz_actual_version) // depreciated

ohh_version_ex(psz_requested_version, psz_actual_version, buffer_size)

Called by the application to determine compatibility. The application supplies the TCSI version and release (e.g. '1.00') it expects. This function checks for compatibility and returns either OK or ERR_VERSION.

The version of TCSI is may be returned in psz_actual_version. (e.g. '1.00.03').

errors: ERR_VERSION Returned if the called TCSI module is not compatible to the requested 'Version.Release' anymore.

Notes on psz_actual_version:

- It is considered that existing applications are using a depreciated code as shown below:

```
char    tcsi_ver[TS_TCSI_VERSION_LEN];  // TS_TCSI_VERSION_LEN is >= 10
if (ohh_version("2.00.00", tcsi_ver)==OK) …
```

- Function ohh_version limits the used length of psz_actual_version to 10 bytes in order to avoid a possible buffer overrun when TCSI32.dll with a long version number is used by clients that where compiled with tcsi.h before KCS 10.1.1 where TS_TCSI_VERSION_LEN was 10. This means that a long version are truncated.
  E.g. TCSI version number "10.1.1.0.0.1234" is truncated to "10.1.1.0.".
- New code should use ohh_version_ex with a minimum buffer size of (TS_TCSI_VERSION_LEN+1) bytes as shown in the example below:

```
char    tcsi_ver[TS_TCSI_VERSION_LEN+1];
if (ohh_version_ex("2.00.00", tcsi_ver, sizeof(tcsi_ver))==OK) …
```

- Since KCS 10.1.1 it is supported to use a NULL pointer for psz_actual_version if the application does not need the TCSI version number.

3.9.4 Code Conversion Functions

ohh_s_ansi2tcos (psz_string, tcos_codepage)

Note: This function has been replaced by ohh_s_codeconv(), it is still kept in TCSI for compatibility but will be removed in a future version.
s_ansi2tcos (psz_string, 0) may be replaced by ohh_s_codeconv(1252, 0, psz_string)
s_ansi2tcos (psz_string, 1) may be replaced by ohh_s_codeconv(1250, 1, psz_string)

Converts the supplied zero-terminated ANSI string (Windows code page 1252 / 1250) to TCOSS code of the specified codepage.

errors: none

ohh_s_tcos2ansi (psz_string, tcos_codepage)

Note: This function has been replaced by ohh_s_codeconv(), it is still kept in TCSI for compatibility but will be removed in a future version.
s_tcos2ansi(psz_string, 0) may be replaced by ohh_s_codeconv(0, 1252, psz_string)
s_tcos2ansi(psz_string, 1) may be replaced by ohh_s_codeconv(1, 1250, psz_string)

Converts the supplied zero-terminated string of TCOSS code to an ANSI string(code page 1252 / 1250).

errors: none

ohh_s_codeconv(int from_codepage, int to_codepage, char * psz_string)

Converts the supplied zero-terminated string. Conversion is done on a character by character basis. The length of the string is limited only by the calling function.

The following combinations of from_codepage and to_codepage are supported (0 = TCOSS 0, 1 = TCOSS 1):

from_codepage	to_codepage
437	0
850	0
1252	0
0	437
0	850
0	1252
852	1
1250	1
1	852
1	1250

Additionally the case of from_codepage and to_codepage being equal is handled correctly (no conversion done, return value OK).

return values: OK ... conversion supported
ERR_WRONG_VALUE ... conversion not supported (string unchanged)

3.9.4.2 Customized Code Conversion

The code conversion function
int **ohh_s_codeconv**(int from_codepage, int to_codepage, char * psz_string)
may be customized. New code conversions may be added or existing conversions modified by providing the conversion table as ASCII file in the shared directory.

Conversion Table Directory:

The TCSI32.DLL takes the conversion tables from the shared directory as set in the registry key "HKEY_LOCAL_MACHINE\SOFTWARE\TOPCALL\SharedDirectory", the default value is "C:\TOPCALL\SHARED". For the 16-bit version "C:\TOPCALL\SHARED" is used.

Conversion Table File Name:

For the conversion from code page A to code page B the conversion table's file name is "AtoB.cnv", A and B being 1 ..4-digit numbers.

Examples:

"850to0.cnv"	replaces the built-in code page 850 to TCOSS 0 conversion
"0to850.cnv"	replaces the built-in TCOSS 0 to code page 850 conversion
"1253to0.cnv"	new conversion from Windows Greek to TCOSS 0
"0to1253.cnv"	new conversion from TCOSS 0 to Windows Greek
"999to0.cnv"	new conversion from custom defined code page 999 to TCOSS 0
"0to999.cnv"	new conversion from TCOSS 0 to custom defined code page 999

All standard conversions are available as files (437to0.cnv etc.) to simplify customization if only a few characters need to be changed.

Conversion Table Format:

A conversion table contains 256 hexadecimal values in 2-Byte ASCII notation, letters 'A' .. 'F' may be in upper or lower case. Comments may be added only at the end of the table.

Example:

```
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e d7
c7 fc e9 e2 e4 e0 e5 e7 ea eb e8 ef ee ec c4 c5
c9 e6 c6 f4 f6 f2 fb f9 ff d6 dc a2 a3 a5 50 83
e1 ed f3 fa f1 d1 aa ba bf 5f ac bd bc a1 ab bb
5f 80 b4 70 6f af af b7 a8 a8 b0 b8 b7 22 b8 af
70 a7 d0 69 48 d7 55 4c 4c b3 8c 6b de 54 6e 6e
6b a4 f0 f0 68 69 49 49 49 f8 9c be fe 74 6e 41
42 df 47 44 45 5a b5 45 54 49 4f 4b 4c d8 4d 4e
58 b1 4f 50 52 53 f7 54 b0 59 46 48 50 b2 6c 20
(Comment: This is the built-in TCOSS 0 to code page 1252 conversion table)
```

The character value in the source code page is used as an index into the conversion table to retrieve the code point in the destination code page. The appropriate conversion table is loaded at the time of the first call to the conversion function and then kept in memory. A maximum of 24 new conversion tables (in addition to the 8 built-in tables) is supported.

Conversion Table Trace:

The loading of custom defined or built-in code tables may be traced by setting a new switch in "DebugLevel": 0x8000 … conversion table trace

3.9.4.3 Conversion Between Unicode and TCOSS Code Pages

INT4 **ohh_s_conv_to_uni**	(INT4 from_cp,	/* in: from TCOSS code page	*/
	CHAR *input_buffer,	/* in: TCOSS string	*/
	INT4 input_length,	/* in: TCOSS string length	*/
	wchar_t *output_buffer,	/* out: Unicode string	*/
	INT4 cchoutputmax)	/* in: max. Unicode string length	

Converts from a TCOSS code page to Unicode. Returns the number of 2-byte characters written to the output buffer. If the output buffer is too small input data will be truncated, the conversion function does not write beyond the buffer size specified in 2-byte units.

No terminating zero byte is appended to the output data. If the input data includes a zero byte this will be transferred as 2-Byte zero to the output data.

INT4 **ohh_s_conv_fr_uni**	(INT4 to_cp,	/* in: to TCOSS code page	*/
	wchar_t *input_buffer,	/* in: Unicode string	*/
	INT4 cchinput,	/* in: Unicode string length	*/
	CHAR *output_buffer,	/* out: TCOSS string	*/
	INT4 output_buffersize)	/* in: max. TCOSS string length	

Converts from Unicode to a TCOSS code page. Returns the number of bytes written to the output buffer. If the output buffer is too small input data will be truncated, the conversion function does not write beyond the specified buffer size.

No terminating zero byte is appended to the output data. If the input data includes a 2-byte zero this will be transferred as zero byte to the output data.

The code page specified with any of the conversion functions may be 0, 1 (TCOSS code pages) or a Windows code page identifier like CP_UTF8. For code pages other than 0 or 1 the Windows API functions MultiByteToWideChar() or WideCharToMultiByte() are called.

Note: With both conversion functions TCOSS string length is specified in bytes, Unicode string length is specified in 2-byte units.

The conversion between TCOSS 0 / 1 and Unicode may be customized by putting files named “TC0toUni.cnv” or “TC1toUni.cnv” into the shared directory (default “C:\Topcall\Shared”). Conversion in both directions is controlled by a single file which is inverted internally. The built-in conversion tables are provided as files to allow fast adaptation.

The Unicode conversion tables contain 256 hexadecimal numbers in 4-digit ASCII notation, giving the corresponding Unicode values in TCOSS code page order. Letters ‘A’ ..’F’ may be in upper or lower case. Comments are allowed after all 256 Unicode values.

If the TCOSS 0 or 1 to Unicode code conversion table is customized in a way that several TCOSS code points are mapped to the same Unicode point, the inverted table for conversion from Unicode uses:

a) a 1:1 conversion if it is defined also, otherwise

b) a mapping to the lowest TCOSS code

3.9.5 Type Number to String Conversion:

TCSI_RET ohh_type2string (TCSI_TYPE nType, char *szBuffer, int nMaxLen);

Return values: OK

WRN_TRUNCATED

ERR_WRONG_TYPE

Maps the numerical value specified by *nType* to a string and returns it in *szBuffer*. The maximum string length including terminating zero is specified with *nMaxLen*. If *nMaxLen* is less than the length of the resulting string, the string is truncated and WRN_TRUNCATED is returned. If the specified type value can not be mapped because it is no valid type, the function returns ERR_WRONG_TYPE and leaves *szBuffer* unchanged.

3.9.6 String to Type Number Conversion

TCSI_RET ohh_string2type (char *szString, TCSI_TYPE *pType);

Return values: OK

ERR_WRONG_TYPE

Maps the string specified by *szString* to a numerical type value returned by *pType*. String mapping is not case sensitive. If an invalid type string is specified, the function returns ERR_WRONG_TYPE and leaves *pType* unchanged.

3.9.7 Object id to String Conversion

TCSI_RET ohh_c_id2string (HOBJ hParent, TCSI_ID nID, char *szBuffer, int nMaxLen);

Return values: OK

WRN_TRUNCATED

ERR_WRONG_NAME

ERR_INVALID_HANDLE

Maps the numerical object identifier *nID* to a string returned in *szBuffer*. The maximum string length including terminating zero is specified with *nMaxLen*. If *nMaxLen* is less than the length of the resulting string, the string is truncated and WRN_TRUNCATED is returned.

The function additionally checks for validity of the identifier within the parent object specified by h*Parent*.

For SET-objects, the child id is valid if the exists or is would be a valid child object even if it does not exist yet.

For LIST-objects, the child id is the index into the list and not a predefined name as with SET-objects. Hence, the id has to be numerical and is valid only if the child object exists.

An invalid id causes the error ERR_WRONG_NAME to be returned by the function.

Note: The function always returns a string in *szBuffer*! If the id can not sufficiently be mapped an empty string is returned.

For SET-objects, the function returns an error but performs an extended search if the specified id is not valid within the parent object. The extended search walks through all possible object ids and tries to find a matching one. If none is found the function returns an empty string, else it returns the id found.

If the parent handle specified by *hParent* is invalid the function returns ERR_INVALID_HANDLE and performs an extended search as described.

The extended search allows mapping ids without the need to open a parent object, or easily extending the search beyond child objects.

The extended search is not performed if the parent object is a LIST-object. Ids in LISTs are bound to child instances and not to predefined child definitions.

The extended search is costly in terms of CPU time. In addition, the function might not be able to uniquely resolve the id to a string. A child id without the corresponding parent object can be ambiguous.

Thus, the parent object should be specified whenever possible.

3.9.8 String to Object id Conversion

TCSI_RET ohh_c_string2id (HOBJ hParent, char *szString, TCSI_ID *pID);

Return values: OK

ERR_INVALID_HANDLE

ERR_WRONG_NAME

Maps the string specified by *szString* to a numerical id value returned by *pID*. String mapping is not case sensitive. If an invalid id string is specified, the function returns ERR_WRONG_NAME and may perform extended search as with *ohh_c_id2string()*.

Extended search is also performed if the specified parent handle is invalid.

If the string can not sufficiently be mapped a value of –1 is returned.

3.9.9 Support of long subject

Kofax Communication Server supports up to 1024 (instead of 128) characters in the subject field of the message header (SET_HEADER/TS_REF). The subject field at other locations still have a limit of 128 characters, such as SET_MAIL_ENTRY_MS_MAIL/TS_REF in the inbox and outbox folders.

To avoid compatibility issues, the maximum subject length (1024 characters) is not set by default. It must be enabled using the following registry value:

Key: HKEY_LOCAL_MACHINE\Software[\Wow6432Node]\TOPCALL\Common\TCSI
(Client applications may also use HKEY_CURRENT_USER\Software\TOPCALL\Common\TCSI with a fallback to HKEY_LOCAL_MACHINE\Software\...)

Value: MaxSubjectLength

Type: REG_DWORD

Value	Description
0	Maximum subject length is restricted by implementation (currently 1024 characters). To use the maximum subject field length, set this value.
128	Subject is truncated after 128 characters. This is the default value which is compatible with all the versions prior to KCS 10.2.
129 ... 1024	Truncates the subject length after xxx characters. Where, xxx can be any value from 129 to 1024. If you want to increase the subject length but the maximum limit (1024 characters) is not supported by all applications, use this value.

3.10 Required TCOSS Release

The full functionality of this interface is only provided with TCOSS release 7.96.00 or higher on the TCOSS server. The following list gives an overview which features will not be supported with older TCOSS releases.

TCOSS release level	Features implemented in this release
TCSI 2.83.00 (KCS 10.1.1)	Support Kofax version numbers.
TCSI 2.81.10 (KCS 10.1)	Support child TS_PASSWORD_NEW of SET_ENTRY_US for password changes.
TCOSS 7.96.21 (KCS 10.1)	New status flag ST_POSSIBLE_RETRY so that retries with alternative numbers can be be shown is retry.
TCOSS 7.96.00 (KCS 10.1) TCOSS 7.95.40 (KCS 10.0.1+FR 4803)	New right flags for message folder of group user and write cover-sheets.
7.95	bit-wise filtering on INT_RIGHTS_USERPROF, INT_RIGHTS_YN_2, INT_RIGHTS_YN, INT_RIGHTS_RW, INT_ANNOTATIONS reception error filtering with “??” and “” update TS_MEMO and INT_ANNOTATIONS in mail archive

	TS_ROLE in SET_SERVER_SESSION (role feature)
7.64	TS_REC_INFO_ORIG original recipient info in mail entry, cover variable $RcInfOr$, cover variables in notification: $FNam$, $MsgId$, $RTime$, $RDate$, $ITime$, $IDate$, $OrInf$
7.56	Time zone support
7.54	Account lockout, password history, minimum password length
7.53	Location-based routing, address routing check function, customer ID in server session
7.46	New licensing model
7.43	Unique message ID generation with TS_TC_MSG_ID in SET_SYSTEM
7.41	Queue length log agent, all send attempts in short term archive
7.39	Alerts
7.37	TS_CHANNEL and TS_SERV_ID in mail entries, reception end time in message store entries, channel status extensions for remote channels (ASP system)
7.35	flag AUTO_REJECT in user profile, flag HANDLE_INVALID_REC for posting of message
7.34	posting while receiving (streaming to TCOSS)
7.33	password expiry
7.30	latest delivery support
7.29	optimized stream read access to block objects, document class XVOICE, “count entries” function in mail folder
7.27	security enhancements, INT_RIGHTS_YN_2 etc. in user store entry, TFC license
7.26	model 210 licenses
7.25	INT_CODEPAGE in text block, code page 932 (Japanese) support
7.24	posting-for-correction with INT_STATE = ST_REJECTED, INT_RIGHTS_RW, INT_RIGHTS_YN in user store entry, CHANGES folder for user profile synchronization
7.23	extended document converter, flag GATEWAY in INT_TERMINATION, flag AUTO_TERMINATION in INT_OPTIONS
7.22	address mapping folder
7.08	license store, extensions in registration store
7.07	basic document converter support, address book synchronization
7.01	distribution lists
7.00	actions restricted to only the first recipient
6.07	extended service prefix length
6.06	gateway interface

3.11 Required TC/ARCHIVE Release

The full functionality of this interface regarding archive server objects is only provided with TC/ARCHIVE release 2.01 or higher on the archive server. The following list gives an overview which features will not be supported with older TC/ARCHIVE releases.

TC/ARCHIVE release	Features implemented in this release
2.01	flags for enhanced administrator functions in INT_ACTIONS
2.00	SET_VOLUME with TS_SECTION, child objects for index volumes, flags for index volumes in INT_STATE_ARCHIVE, flags for automatic CD writing in INT_ACTIONS
1.05	search on recipient or originator group
1.04	INT_ERROR with value SEARCH_TOO_COMPLEX in SET_ENTRY_ARCHIVE of archive search folder
1.02	volume path for online volumes, flag STA_FILE_ERROR in INT_STATE_ARCHIVE
1.00	first TC/ARCHIVE release

3.12 Trace Options

Trace Options are set in the registry for the 32-Bit version or in the TOPCALL1.INI file for the 16-Bit version. The following description is for the 32-Bit version only:

Registry sub key "TCSI\DebugLevel" (under process main key, e.g. "..\TOPCALL\TCFW")

The following bits are defined (hexadecimal notation):

0x0001 Error Trace (default, always added if any of the other bits is set)

0x0002 Call Trace (traces all calls to TCSI functions)

0x0004 Binary Put Trace

0x0008 Receive Object Trace (traces all objects received from TCOSS)

0x0010 Binary Get Trace

0x0020 SendObject Trace (traces all objects sent to TCOSS)

0x0040 Heapcheck Trace

0x0080 Handle Trace

0x0100 TCTI Call trace

0x0200 File Handle trace

0x0400 L4 Dump

0x0800 object check

0x1000 TOS calls

0x2000 TCTI attach - detach trace

0x4000 Session Trace

0x8000 code conversion tables trace

Most useful are the error trace, the call trace and the send and receive traces. All other traces are TCSI internal.

3.13 Error Codes

The following error codes are defined in TCSI (some are for internal use and do not appear on the interface):

error code	description
10	text string was too long, has been truncated
100	ok
107	the end of the object (or file) has been reached
200	send orders deleted partially
201	texts deleted partially
202	send orders re-routed partially
300	attempt to overwrite an existing object
301	the specified object does not exist
302	maximum number of entries / envelopes reached
303	maximum number of send orders reached
304	object is temporarily locked
305	send order(s) not deleted
306	file(s) not deleted
308	maximum store capacity reached
309	temporarily no handle available
310	memory full

311	abbreviated number does not exist
312	abbreviated numbers file does not exist
314	drive does not exist
315	disk error
316	abbreviated numbers file locked
317	wrong command
318	no message
319	no record
320	loop detected
321	send order(s) not re-routed
322	insufficient user rights
400	bad syntax of ..command
401	bad parameter in ..command
402	missing parameter in ..command
403	bad reference
404	bad correction mode
405	bad number
406	bad channel
407	bad date
408	bad time
409	bad low value of line number
410	bad high value of line number
411	bad number series
412	bad actual value of number series
414	bad value for last value of number series
415	bad value for file number
416	bad author
417	bad mode parameter
418	bad break parameter
419	bad error code
420	bad document- and page-number
421	bad type parameter
422	bad end code parameter
423	bad stop parameter
424	bad connection time parameter
425	value exceeds the object's range
426	bad termination parameter
427	bad nodes parameter
428	bad originator
429	bad user id
430	bad notification parameter
431	bad cost centre
432	bad possible duplication parameter
433	bad send type
434	bad priority
435	bad message type parameter
436	bad record (= record too long)
500	different object version (object changed)
501	no object exists for the specified handle (invalid handle)
502	truncate with wrong length
600	call of function is not permitted
601	not enough memory left to perform operation
602	save attempt to an object without parent
603	type does not allow this kind of operation
604	invalid time value

605	invalid sub object specifier
606	invalid number series
607	invalid channel specifier
608	connection to server could not be established
609	invalid user ID
610	wrong password
611	not possible to maintain connection
612	server temporarily busy
613	passive session was cancelled
614	selector object is invalid
615	corrupted entry
616	a receiver entry could not be accepted
617	invalid originator (missing or no active address)
618	status change not possible
619	the end of block is reached
620	wrong function for that object type
621	registration limit reached
622	illegal ASCII object format
623	invalid syntax for path
624	incompatible program versions
625	the password has expired
626	folder name too long for local folder
627	the new password is invalid (e.g. equal to old one, starts with “#”)
628	wrong password, next invalid attempt will lock account
629	your account has been locked
630	disconnected due to transport problems
631	no call confirmation received within connect timeout
632	the new password is too short
633	text is too long to fit into field
634	the new password must be more complex
700	session was cancelled
701	stream was cancelled by control frame
702	disconnect for any reason
703	wrong type of stream
750	maximum data size exceeded
751	invalid transport attachment handle
752	temporarily no access point available
754	sending queue full
755	line disconnected
756	illegal frame received
757	disconnect indication received
758	disconnect due to line problems
759	call indication received
760	call confirmation received
761	data indication received
762	control data received
770	error during connect
771	error during disconnect
772	server could not be started
801	maximum possible level of object nesting reached
802	internal object buffer overrun
803	internal error inserting into a sorted index
804	internal error deleting from a sorted index
805	maximum number of sessions reached
806	internal problem with file number in CIF process

807	internal error reading envelope
808	internal error writing envelope
851	server received entry without content
852	wrong function code sent to server
853	lock of memory block failed
854	container has no child slot left
855	corrupted stream received
888	general fatal error code
889	can not initialize stream (tcti, file, ..)
900	TCSI32 initialization failed
901	create mutex for login lock failed

Additional error codes returned by an Archive Server:

704	maximum number of documents on archive volume reached
705	buffer for mail entry too small
706	info record split
707	string too long for word tree
708	error inserting key into word tree
709	info occurrence list record full
710	info already set in occurrence list
711	wrong index in volume table
712	error creating win32 event
713	info incompatible file version
714	error in file access
715	transaction handler blocks maximum number reached
716	info search thread reset
717	no search thread found
718	search result buffer overflow
719	wrong document number
720	CD volume compare failed, files not equal
721	maximum number of volumes reached
722	search stopped for CD writing
723	error with tcrOpen function
724	volume space full
725	archive data base file corrupt
726	error allocating memory
727	archive disk full
728	index volume can't be created
729	corrupt word tree building index volume
730	corrupt occurrence list building index volume
731	internal error opening index volume
732	Error in communication with TCJUKE process
733	TCJUKE process did not complete function within timeout
734	internal error in CD writer thread
735	encrypted copy of volume not found
736	no blank CD or no CD writer accessible
737	CD could not be written without errors
738	the jukebox manager service is not responding
739	jukebox or magazine inactive
740	not enough space on CD
741	generic jukebox access or CD writing error
742	jukebox / CD writer license error

4 Subject Index