SCOL SERVER

Version 4

SCOL ADMINISTRATOR GUIDE

Table of Contents

1. INTRODUCTION

SCOL Server is the software for hosting sites created using SCOL technology, to make them available to visitors with the SCOL Voy@ger plugin.

SCOL Server can be used in two modes: Simple Mode and Advanced Mode.

The Simple Mode corresponds to "manual" hosting and is similar to the operation of previous versions of SCOL Server.

In this mode, SCOL sites created for example using the SCS or E-maginer tools must be copied to the server, and then defined in a database managed by the SCOL Server.

This mode is used when the SCOL sites to be hosted are known in advance.

The Advanced Mode is used for mass hosting of SCS or E-maginer sites.

This mode uses the upload possibilities included in the SCS or E-maginer tools.

It makes a hosting space available to receive SCOL sites created by SCS or E-maginer users with a hosting account on the server.

This mode is thus intended for Internet (ISP) hosting services that do not know in advance the sites to be hosted.

But this mode is not included in the basic SCOL Server installation. Additional software must be provided by Cryonetworks to make it operational.

However, this manual contains all information concerning advanced mode.

This document is divided into 3 main parts:

- general introduction to SCOL technology
- SCOL Server in Simple Mode
- SCOL Server in Advanced Mode

2. SCOL TECHNOLOGY

SCOL uses the high-level SCOL programming language to create client/server type applications on TCP/IP (mainly Internet-oriented) networks.

A plugin is required both on the client and server side to enable communication between machines.

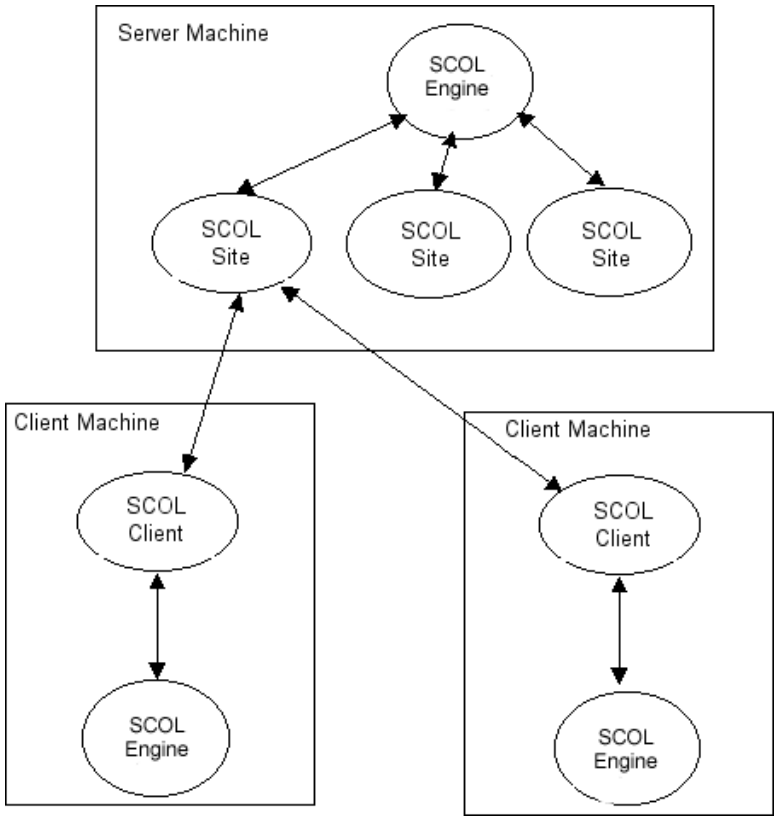
This plugin contains among other things a virtual system capable of compiling SCOL language source into bytecode on the fly.

It can also automatically download files required for execution on the client side of the application from the server: the code is mobile.

A special SCOL program runs on every machine called the SCOL Engine (also called SCOL Voy@ger on client machines).

This program plays several roles:

- it supervises the other SCOL programs running locally on the same machine: these programs are called "SCOL sites" and each corresponds to a process of the machine on which they are running.
- its serves as a name server when a remote machine wants to know the SCOL programs which are active locally, or when it wants to connect to one of the active programs.



A SCOL client that wants to connect to a SCOL site must know two things:

- the IP address of the server machine
- the TCP port number of the SCOL site

The IP address can be known in three ways:

- it is directly known by the client
- it is resolved via the DNS by giving a machine name instead of the IP address
- it is requested from a SCOL directory which references the active SCOL servers

the TCP port can be known in two ways:

- it is directly known by the client
- it is resolved via the remote SCOL Engine which is supervising the SCOL site, by giving a SCOL service name instead of the port number

Example: typing the URL **scol://www.cryonetworks.com:Cryopolis** into a Web browser first resolves the IP address of the machine identified by www.cryonetworks.com via the DNS.

Then a connection is established to the SCOL Engine of this machine on port 1200 (the SCOL Engine default port) with the service name Cryopolis. The latter responds by giving the port number of the service. The client then establishes a direct connection through this port to the Cryopolis SCOL site.

Otherwise, since SCOL version 3.1, a SCOL client can connect to a SCOL site using the HTTP protocol instead of TCP.

This allows a client to bypass a possible proxy/firewall which limits external access (exclusion of protocols and/or ports).

Each SCOL site is thus listening for two TCP ports:

- a port n for a direct TCP connection
- a port n+1 for a HTTP connection .

The SCOL Engine itself is listening to port 1200 under TCP and port 1199 under HTTP.

It is also listening to ports 80 and 8080 under HTTP to play the role of "SCOL proxy": if a client fails to establish a direct link with a SCOL site (neither with TCP on port n, nor with HTTP on port n+1), it attempts to reach the SCOL Engine, via port 80 or 8080 with HTTP, which then routes the client request to the right site.

The SCOL Engine is a software layer written itself in SCOL language.

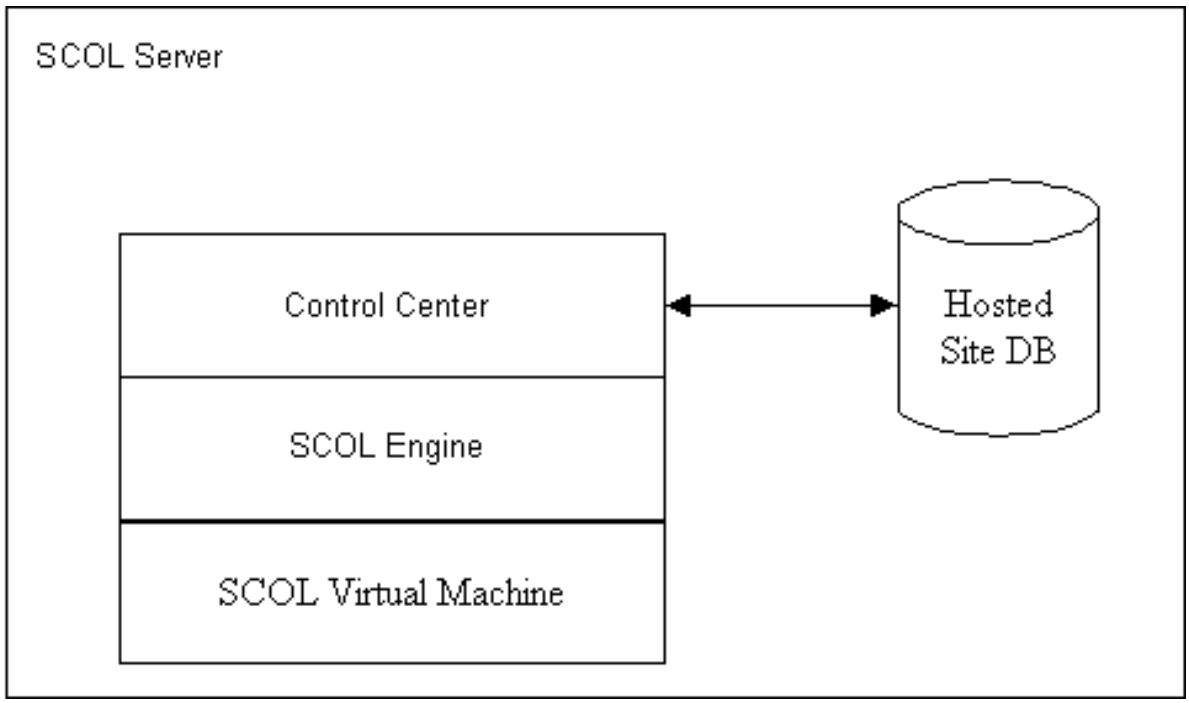
It can be viewed as a layer built on top of the SCOL virtual system.

SCOL Server version 4 can be used to define hosted SCOL sites in a database.

Properties of hosted sites can be modified remotely through an HTML administration interface using, for example, a Microsoft Internet Explorer browser. (See the document "HTML Administration Reference Manual" for a complete description).

This is enabled by another software layer written in SCOL language called the Control Center.

As shown in the following diagram, the Control Center itself is built on top of the functionality of the SCOL Engine:



3. SCOL ENGINE SETTINGS

The SCOL Engine requires two initialization files located in the installation directory: **usm.ini** and **usmress.ini**.

Once SCOL Server is installed, these files are correctly defined and it is not necessary to modify them to begin using the server.

If however the SCOL administrator wants to modify its operation, he can edit these files directly on the server, or preferably use the HTML administration interface (see the document "HTML Administration Reference Manual" for a complete description).

In the HTML interface, choose the tab "**SCOL Configuration**".

Simple Mode offers three groups of parameters: **network**, **history** and **SCOL Engine**.

3.1. Network Configuration

Forced IP

This is the apparent SCOL Server address for external access.

If it is not filled in, the SCOL Server automatically determines the machine's IP address.

Entering a value in this field forces a SCOL program to listen to a given port only on this network interface. Otherwise, it listens to a given port on all the machine's network interfaces.

This field can be used for example to make the SCOL Server cohabit with a WEB server: both need port 80, but each is defined for a different IP address.

Listening Port

TCP port number the SCOL Engine listens to particularly to resolve SCOL service names.

The default port 1200 must be used for SCOL Voy@ger clients to have normal access to hosted SCOL sites.

HTTP ports

List of HTTP ports the SCOL Engine is listening to, for redirecting requests of clients behind a firewall.

The default ports are 80 and 8080.

Not defining these ports does not prevent the SCOL Server from operating normally, but certain clients behind a restrictive firewall will not be able to access hosted SCOL sites.

3.2. History

During its execution, the SCOL Engine and the launched SCOL sites can generate history files useful for maintenance purposes.

In particular, the activation of history logs may be requested by Cryonetworks support if there are problems.

The history files are generated in the **log** subdirectory of the SCOL installation directory with the extension **.log**.

In normal cases, history is not activated because it slows the execution of SCOL sites.

3.3. SCOL Engine

Language

Defines the default language of SCOL sites.

This value can be used by modules of sites using the DMS (Distributed Module System) library, as is the case for SCS or E-maginer sites.

Thus, a visitor to a SCOL site whose SCOL Voy@ger specifies a language not included in the available localizations of the SCOL site will see a SCOL interface in the default language defined in this field.

License

License number allowing operation of the SCOL server.

This value is initialized when SCOL server is installed.

3.4. Expert Mode

In the "**SCOL Configuration**" tab of HTML Administration, the button "**expert mode**" is used to configure other more sensitive SCOL Server settings.

HTTP Administration Port

By default 1201: defines the port to access HTML Administration.

SCOL Partitions

For more secure access to files, SCOL uses the concept of "partitions" defined in the file **usm.ini**.

SCOL Partitions are directory names used by SCOL programs for relative access to files.

Write access is only possible in a single partition: the "cache" partition for clients and the first partition following the cache for servers.

This controls and restricts file access for a SCOL program.

For the SCOL Server, the partitions are:

```
disk ./cache 64000
disk ./partition 0
disk ./common
```

`./cache`	cache partition used by clients (not used by SCOL Server)
`./partition`	SCOL Engine files (read/write)
`./common`	files common to all hosted sites: graphic resources, SCOL source (modules), … (read only) Different types of sites are differentiated: this directory contains subdirectories corresponding to Cryonics, E-maginer and SCS sites

usmress.ini

This file contains the definition of SCOL resources.

These resources can be viewed as global variables accessible by all hosted SCOL sites.

Only the license number, defined in this file, remains inaccessible to a SCOL program.

4. CONTROL CENTER IN SIMPLE MODE

This mode is used to host SCOL sites known in advance and defined statically in the Control Center database.

The physical model of this database is provided in the appendix.

In Simple Mode, only three database tables are used: *admin*, *types* and *sites*.

These tables can be manipulated by the HTML Administration interface (see the document "HTML Administration Reference Manual" for a complete description).

These tables must be configured in the order described in the following sections.

4.1. Administration Settings

SCOL Server is provided with default administration values that are suitable for most cases.

However they can be changed by the HTML interface.

To do this, choose the "**Control Center**" tab and the "**Servers**" category to the left, and click on the line's red editing button.

In Simple Mode, the only useful fields are:

DNS	initialized when SCOL Server is installed, this is the DNS name of the server appearing in the URLs of hosted SCOL sites.
minPort (default 2000) maxPort (default 3000)	defines a range of ports: when a SCOL site is launched by the Control Center, the latter attributes it a free port from this range Note: a specific port number can also be defined at the site level (see the section "Site Definition")
MaxSites (default 100)	maximum number of SCOL sites launchable concurrently by the Control Center
Capacity (default empty)	Defines the default capacity of SCOL sites in terms of connected clients; this value is only useful in the case of a shared SCOL license of type M Note: a specific capacity can also be defined at the site level (see the section "Site Definition")
Close delay (default 120 or 2 min)	Automatic Close Delay of SCOL sites when there are no more visitors (in seconds) Note: a specific timeout can also be defined at the site level (see the section "Site Definition")

4.2. Product Settings

This is where different types of SCOL sites are defined.

The following types are predefined in the table *types* in the SCOL Server package: **cryonics**, **cryonics_flash**, **emaginer**, **emaginer_online** and **scs**.

SCOL sites are classified by type because the Control Center must differentiate them according to the DMS and module versions they use.

From a SCOL technical perspective, when a site is launched by the Control Center, the command `_refine_nth` is applied to it with parameters "**PartNumber**" and "**partition refine**" of its associated type.

This command "refines" a partition defined in the file **usm.ini** with a subdirectory of this partition. The file names used in the SCOL site will thus be read relative to this subdirectory.

When a SCOL site is launched it is thus placed in an execution environment associated with its type.

The predefined types thus provide for three different DMS and module versions, the latter being copied in the subdirectories "**cryonics**", "**emaginer**" or "**SCS**" in the partition "**./common**".

Defining another type is only necessary if you want to host a SCOL site using a DMS or module version different from and incompatible with those used in other hosted SCOL sites.

If you want to define new types, choose the "**Control Center**" tab in HTML Administration, and the category to the left, "**Products**" and click the "**Add**" button.

Product name	identifier of SCOL site type
PartNumber	partition number to be "refined" numbering starts at zero for the first partition used in the file usm.ini (for example, enter the value 1 for the partition **./common** which is the second)
Partition Refine	subdirectory name of the partition to be refined This subdirectory should contain the DMS files and the modules used by the sites belonging to this type
Select Directory category	unused in Control Center Simple Mode
Number of ports	if the Control Center itself attributes a port number to a site it wants to launch, this gives the number of consecutive free ports it must find dynamically in the range defined by the *admin table .*
Load File	gives the name of a file containing SCOL loading instructions for DMS files. This is used to associate the type with a DMS version. Three versions are predefined in the directory **./partition/locked/ctrlcenter/script**: HDMS, DHDMS and EMDHDMS (used for E-maginer)

4.3. Site Definition

In the HTML Administration interface, choose the "**Sites**" tab and click the "**Add New**" button.

A first option chooses between a site managed or not by the Control Center.

This latter case is described at the end of this section.

The following settings concern the usual case of a DMS site managed by the Control Center.

4.3.1. DMS Site Managed by the Control Center

Name	this is the SCOL name for the site as known to visitors
Author	this field is usually empty it is used to complete the SCOL name for the site which will then be of the form *author.name*
Type	assigns the site to a type defined in the *type* table (see the preceding section "Product Settings")
Close Delay	(in seconds) assigns a specific value to the timeout for the site when there are no more visitors This value can be globally defined in the *admin* table for all hosted sites (see section "Administration Settings")
Capacity	Defines the specific capacity of the site in terms of connected clients; this value is only useful in the case of a shared SCOL license of type M This value can be globally defined in the *admin* table for all hosted sites (see section "Administration Settings")
SCS	gives the name of the site's description file the file has extension *.scs* or *.dms* depending on the DMS version used
Auto Start Site	indicates if the site must be started automatically upon launching the SCOL Server This option overrides the functionality of the Control Center which can wait for the first visitor to connect to a SCOL site before launching it. Similarly, the Close Delay field is ignored if this option is chosen.

4.3.2. Site Not Managed by the Control Center

This is a very special case for activating a SCOL program on the server which is not a DMS SCOL site.

In this case, simply enter the name of the SCOL script for launching it, in the corresponding field (file with extension **.scol**).

The usual Control Center functions do not apply to this type of site: startup on connection of first visitor, "refine" of the execution partition on startup, close when no more visitors, etc.

4.4. Example of Putting a SCOL Site Online

We assume in this example that the SCOL Server is correctly installed with the appropriate administration settings.

Take the case of a SCOL site created with a SCS tool on a development PC.

The site is called "**beach**" and uses the standard SCS modules.

Its description file "**beach.dms**" was saved on the development PC in the directory **./partition/worlds/beach**.

Copy of DMS

Copy the directory **./partition/dms** from the PC to the server in the directory **./common/SCS/dms**:

Copy of Site

Copy the directory **./partition/worlds/beach** from the PC to the server in the same directory **./partition/worlds/beach**.

Definition of Site

Through the HTML interface, define the site in the Control Center using the "**Sites**" tab with the following values:

Name	beach
Author	empty
Type	SCS Since the site uses standard SCS modules, it is assigned the predefined product type "**SCS**"
Close Delay	empty The global value defined in the *admin* table will therefore be used
Capacity	empty The global value defined in the *admin* table will therefore be used
SCS	worlds/beach/beach.dms
Auto Start Site	empty The site will not be launched before the first visitor connects

The site is then accessible through the URL **scol://*DNSname*:beach** where *DNSname* is the server DNS name defined in the *admin* table.

5. CONTROL CENTER IN ADVANCED MODE

This mode allows a SCOL server to host a large number of DMS (Distributed Model System) SCOL sites.

The Control Center is connected to a database containing the list of hosted SCOL sites (see the appendix for the database physical model).

A site is not launched on a machine until a visitor requests a connection to it.

When there are no more visitors to a site, it is stopped.

The number of processes running on the machine is thus controlled and, in practice, is considerably less than the total number of hosted SCOL sites.

But even if they are not actually running, the hosted SCOL sites are declared active in a local SCOL directory. This local directory is itself referenced by a global SCOL directory hosted by Cryonetworks (URL **scol://scol.cryonetworks.com:SiteDirectory**). They are therefore accessible at all times by visitors who consult the directory (global or local).

More specifically, in advanced mode, the Control Center performs five functions:

- Management of Hosting Accounts
- upload of a Cryonics, E-maginer or SCS2 site
- local directory listing hosted sites
- launching a hosted site on demand
- purge of sites

These functions are described below for basic Control Center operation.

It is possible to customize this basic operation as shown in the following section.

5.1. Management of Hosting Accounts

A site hosted by the Control Center must belong to a hosting account known to the Control Center.

Hosting accounts must therefore exist before a site is hosted.

The main information about a hosting account is:

- login
- password
- personal directory
- email address
- maximum number of hostable sites
- duration of site hosting

This information is stored in the Control Center database (see appendix, tables *users* and *keycode*).

The Control Center itself does not manage creation of these accounts. This role is performed by an external application.

For example, the Cryonetworks solution uses an HTML application managed by the WebObjects tool.

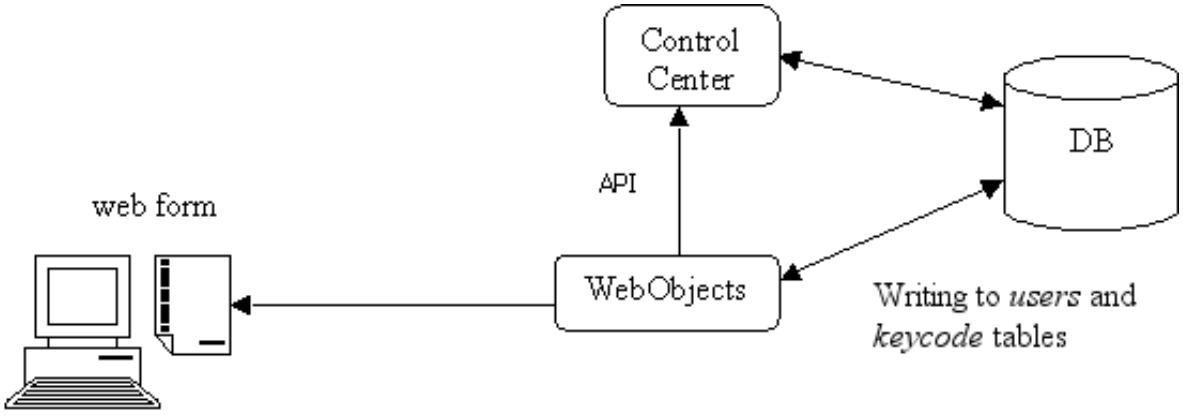
This utility generates Web forms accessible by a Web browser that read and write in the Control Center database.

Any other tool can be used, as long as it obeys the format of the database tables *users* and *keycode*.

In addition, the WebObjects application offered by Cryonetworks allows a registered user to consult their hosted sites.

They can thus see how much hosting time remains, delete a site, change their password, etc.

Again, another utility than that offered by Cryonetworks may have other functions.



The only constraint that must be met by this external application is to write only in the tables *users* and *keycode*. All other database tables are strictly read-only.

Access rights to the database could be set accordingly.

To access other functions (for example removal of a site), the Control Center can be controlled by SCOL requests or HTTP requests (see section CONTROL CENTER APPLICATION PROGRAMMING INTERFACES).

5.2. Uploading a Site

5.2.1. Principle

The Control Center manages sites produced by Cryonics, E-maginer and SCS2 software from Cryonetworks.

These tools have the functionality to automatically upload a site's files to a Control Center server.

A hosting account must of course be created before uploading.

On the user side, the user has only to choose a hosting server (directly enter the address for Cryonics or select in a list for E-maginer and SCS2), provide a login and password and begin the upload.

On the server side, the Control Center listens to the upload ports configurable in the *admin* database table.

After authentification and validation, the Control Center creates the files and information making up the site on disk and in the database.

5.2.2. Restrictions

Upload restrictions exist (particularly for SCS2 sites): no SCOL code can be transferred to the Control Center server.

And certain modules can not be used in SCS2 sites. In particular modules that use a database or that listen to a port (TCP or HTTP).

5.2.3. Partitioning of Server Disk Space

All hosted sites use a common SCOL file hierarchy: graphic library, SCOL programs, etc.

Each site then has a dedicated disk space, which contains data specific to the site author.

A few megabytes (less than 5) are amply sufficient to store user-specific data.

For the Control Center, the partitions are:

```
disk ./cache 64000
disk ./partition 0
disk ./common
```

`./cache`	cache partition used by clients (not used by Control Center)
`./partition`	SCOL Server files (read/write)
`./common`	files common to all hosted sites: graphic resources, SCOL source (modules), etc. (read only) Different types of sites are differentiated: this directory contains subdirectories corresponding to Cryonics, E-maginer and SCS sites

When a site is launched by the Control Center, it uses a new partition which is necessarily a subdirectory of **./partition**.

This partition corresponds to the disk space allocated to the site author (the personal directory of the hosting account defined in database table *users*).

For example, a user John could have a dedicated space in the directory **./partition/users/john**.

When one of their E-maginer sites is launched by the Control Center, it would use the partitions:

`./partition/users/john`	personal files (read/write)
`./common/emaginer`	files common to all hosted E-maginer sites: graphic resources, SCOL source (modules), etc. (read only)

Note for UNIX systems: a user's specific subdirectory can be a symbolic link to another file hierarchy. Thus, in the preceding example, **./partition/users** may be a symbolic link to a NFS mount point.

This partitioning provides two advantages:

- if the site is not customized, the amount of data uploaded to host the site is very small (a few Kb) since all the files making up the site are already contained in the server common partition (**./common**)

- the sites of different users are independent of each other, because the personal partition (subdirectory of **./partition**) is different for each user: one site can not access the files of another

5.3. Local directory

Once uploaded and created in the database, a hosted site is visible in a local directory.

This is a SCOL site that lists all hosted sites of a Control Center.

It is accessible via the URL **scol://*DNS*:*localdir***

where *DNS* is the DNS Name of the Control Center server machine and *localdir* is the name of the local directory.

For example, **scol://scs.cryonetworks.com:cryonetworks**

The local directory is recorded in the Cryonetworks global directory (SiteDirectory).

The global directory is divided into categories.

A category is reserved for each local directory: all sites controlled by the Control Center are referenced in this category.

A login/password authentification mechanism is used when the local directory is recorded in the global directory to prevent inappropriate records within the category.

This information (hosting service category, login, password) are defined in the dmi file of the local directory (**./partition/dirloc/dir.dmi**).

In advanced mode (see the next section), more than one local directory can be assigned to a Control Center.

The allocation of sites to the local directories can correspond to the hosting login of the sites and their type.

5.4. Launching a Site

A hosted site is accessible in two ways: through the directory or through its URL

5.4.1. Access by Directory

Even if they are not necessarily running, all hosted sites are declared active in the local directory (and in the associated category of the global directory).

The site can be accessed by simply selecting it in the directory and clicking the "**GO**" button.

5.4.2. Access by URL

A site's URL can also be used directly by a Web browser or by SCOL Voy@ger.

It has the form:

scol://*DNS*:*localdir*author.name

with:

- *DNS:* DNS Name of the Control Center server machine
- *localdir:* name of the local directory

- *author:* site author
- *name:*site name

For example, **scol://scs.cryonetworks.com:cryonetworks.john.beach**

5.4.3. Startup

The Control Center supervises access requests to sites.

If a request arrives for a site that is not running, it searches for three free consecutive ports among a given range (configurable in the *admin* database table) and attributes them to the site (a TCP port n, an HTTP port n+1, and a port n+2 for the dyneditor online dynamic modification module of a E-maginer site).

It can then launch it and place it in its personal execution environment (subdirectory of **./partition**).

If there are no visitors to the site for 2 minutes, it stops it (the time can be configured in the database).

5.5. Purge of Sites

The hosting duration of Control Center sites can be configured in the database (see table *keycode*).

A few days before expiry (also configurable), a warning mail is sent to the user.

The mail can be localized if language information was provided when the hosting account was created (see table *users*).

The content of the message is contained in the files **./partition/locked/ctrlcenter/lang/mail.*type*.*language*.lang**, where *type* is the site type (Cryonics, E-maginer or SCS2, see table *types*) and *language* is the language of the hosting account (FR, GB, etc.).

After expiry, the site is deleted from the Control Center database, along with the user's personal directory files, unless they are used by other sites.

In addition, a user that does not have other sites is deleted from the database tables *keycode* and *users*.

6. CUSTOMIZATION OF CONTROL CENTER FUNCTIONS

Control Center is an application written in SCOL language. It is made up of files located in the directory **./partition/locked/ctrlcenter**.

Its architecture can be viewed as follows: a central core, non-modifiable in principle, and a set of modifiable functions, contained in the file **./partition/locked/ctrlcenter/ccisp.pkg**.

This file contains functions and variables used by the core whose basic operation was described in the previous section.

These functions and variables can be modified to customize behavior of the Control Center. But the original prototype must be preserved (type of variable, types of function and its parameters).

An important Control Center function that can be customized is the management of hosting accounts.

In basic operation, they are defined in tables *users* and *keycode* of the Control Center database.

Other mechanisms could be implemented here (access to another database, LDAP access, etc.).

An exhaustive list of customizable variables and functions is provided in this section, with each one's prototype.

6.1. Variables

```
typeof CCdbnameU        S
```

Name of database containing the tables *users* and *keycode* of the hosting accounts (more exactly, database ODBC Data Source Name).

```
typeof CCdbloginU       S
```

Access login to database.

```
typeof CCdbpwdU         S
```

Access password to database

```
var CCtabusers          S
```

Name of users table.

```
var CCtabkeycode        S
```

Name of keycodes table.

```
var CCident             S
```

Identifier of current Control Center.

This identifier concerns Control Center multiserver architecture. See section "MULTISERVER CONTROL CENTER".

```
var CCdefaultlang       S
```

Default language used for localization of warning mail sent before expiry and removal of a site.

6.2. Functions

```
fun CCgetSubcatISP(usr,type)        fun [S S] S
```

Subcategory in the local directory for a given hosting account login and type of site.

The site type corresponds to the software that produced the site: Cryonics, E-maginer or SCS2 (see database table *types*).

Subcategories are used to organize the sites in the local directory in hierarchical form (see database table *categories*).

```
fun CCgetRegISP(usr,type)           fun [S S] S
```

Name in the local directory for a given hosting account login and type of site.

All sites belonging to this login or this type will be in this local directory.

Can be use to distribute hosted sites among several local directories associated with the same Control Center.

Each will appear in a different category of the global directory.

```
fun CCgetNextId(cur)          fun [S] S
```

Identifies next Control Center in the case of multiserver architecture.

See section on this subject.

```
fun CCgetCCidentISP(usr)      fun [S] S
```

Control Center identifier for a given hosting account login.

Used to distribute uploaded sites among more than once Control Center in mulstiserver case.

See section on this subject.

```
fun CCinitISP()               fun [] u0
```

Initialization function called during Control Center startup.

Typically, necessary initialization for access to hosting account data.

```
fun CCmkName(locdir,author,name)          fun [S S S] S
```

SCOL name of hosted site according to local directory, author and site name.

This function must be consistent with the following.

```
fun CCsplitName(sitename)     fun [S] [S S S]
```

Three element tuple made up of local directory, author and site name based on SCOL name of hosted site.

This function must be consistent with the preceding.

`fun CCgetCCidentISP(usr)  fun [S] S`

Personal directory for a given hosting account login.

This directory will be the subdirectory of the partition **./partition** corresponding to the specific environment of the sites belonging to the hosting account.

This directory must not start with a '/'.

`fun CCgetUserInfo(userISP,passISP,fcrypt)  fun [S S fun [S] S] [S I]`

Authenticates a user for site uploading.

The function receives the login, encrypted password, and encryption function as input.

If the encrypted password is identical to that of the hosting account to which the encryption function is applied, a tuple is returned made up of the personal directory (see function `CCgetDirISP`) and the maximum size of the personal directory.

Otherwise, nil is returned.

`fun CCgetNbPort(usr,type)  fun [S S] S`

Number of consecutive free ports to be found in a given range when launching a site for a given hosting account login and site type.

`fun CCgetLoadFile(usr,type)  fun [S S] S`

Gives the name of a file containing SCOL loading instructions (`_load`) to launch a site for a given hosting account login and site type.

This differentiates sites according to the DMS version they use.

`fun CCgetCapacity(usr,type)  fun [S S] S`

Gives the capacity (in S) of a site when created in the database (first upload) for a given hosting account login and site type.

Capacity is the maximum number of visitors that can connect to the site.

This is a part of the global M license of the SCOL Server.

If this value is nil, site capacity will be the default defined in the database table *admin*.

`fun CCgetCloseDelay(usr,type)  fun [S S] S`

Gives the time in seconds (in S) after which the site is stopped when there are no more visitors, for a given hosting account login and site type.

If this value is nil, the site close delay is the default defined in the database table *admin*.

A value of –1 means infinite time: the site will not be stopped.

`fun CCcheckSiteUpload(name,author,type,userISP,keycode)`

`fun [S S S S S] I`

Performs various validations when a site is uploaded.

The input parameters are the site name, author and type, the hosting login and the keycode.

Returns a status code with the following interpretation:

- nil -> global error
- 0 -> check OK
- 1 -> max nb of sites attained
- 2 -> invalid user/keycode
- 3 -> expired validity period
- 4 -> site/author already defined

`fun CCupdDatcreISP(keycode,datcre)  fun [S S] I`

Updates creation date for a given keycode if not already done.

Returns a non-zero value in case of problems.

The creation date is the date from which the validity period is calculated.

It corresponds to the first upload of a hosting account.

`fun CCgetKeycodeInfo(keycode)  fun [S] [S r1]`

Information concerning a given keycode.

Composed of the following values:

- creation date (see function `CCupdDatcreISP`)
- validity period in days
- indicator to send mail before expiry date
- days before expiry date to send mail

`fun CCdelKeycode(keycode)  fun [S]u0`

Function called when removing an expired site.

Deletes keycode associated with site if not referenced by any site.

`fun CCdelUser(userISP)  fun [S] u0`

Function called when removing an expired site.

Deletes hosting account associated with site if not referenced by any site.

`fun CCsendExpiredMail(userISP,keycode,type)  fun [S S S] u0`

Sends a warning mail before the expiry and removal of a site for a given hosting login, keycode and site type.

Also positions a mail sent indicator to true, to avoid duplicate mailings.

`fun CCcreSiteISP(type,dmsid,name,desc)  fun [S S S S] I`

Function called when creating a site.

The input parameters are the site type, identifier, name and description.

If the function returns a value other than zero, the site will not be created in the Control Center.

`fun CCcheckExpired(userISP,keycode,type)  fun [S S S] I`

Determines if a site has expired for a given hosting login, keycode and site type.

Returns 1 if the site has expired.

This function is called hourly by the Control Center.

It can thus also be used to send a warning mail (see function `CCsendExpiredMail`)

`fun CCsiteStarted(sitename,port)  fun [S I] I`

Function called when a site launched by the Control Center has actually started up.

Parameters are the SCOL site name and the port attributed to it.

`fun CCgetDNSname(ident)  fun [S] S`

DNS Name which appears in the URL of a hosted site for a given Control Center identifier.

`fun CCgetDMSid()  fun [] I`

Attributes a unique identifier to a site.

7. CONTROL CENTER APPLICATION PROGRAMMING INTERFACES

The rule for external applications that want to use the Control Center is that they may only write to the database tables *users* and *keycode* .

Access to other data and functions of the Control Center must use application programming interfaces (APIs).

Two mechanisms are provided: through SCOL requests or HTTP requests.

7.1. SCOL Requests

Certain Control Center functions can be called from an external SCOL application

using the SCOL "`defcom`" mechanism.

Since the Control Center is simply an extension of the SCOL Engine, only the channel `_masterchannel` is usable, requiring that the external SCOL application be on the same machine as the Control Center.

The following functions can be called (located in the file **./partition/locked/ctrlcenter/ccmaster.pkg**):

```
fun __CCaddFile(name,author,type,userISP,file) fun [S S S S S] I
```

Tells the Control Center that a file was just created in the personal directory of a hosting account.

The input parameters are the site name and author, the site type, the hosting login and file name relative to the partition **./partition** (it must therefore include the personal directory).

The Control Center stores this information in the database table *files*.

It can then delete the files making up a site when it expires, if they are not used by another site.

```
fun __CCdelSite(name,author,type,userISP,keycode)     fun [S S S S S] I
```

Delete a Control Center site.

The input parameters are the site name, author and type, the hosting login and the keycode.

The following operations are performed:

- removal of the site from the database
- termination of site if running
- removal of files making up the site contained in the user's personal hosting directory, unless they are used by another site.
- removal of keycode from the database if not referenced by another site
- removal of hosting account from the database if not referenced by another site

```
fun __CCgetUrlScol(name,author,type,userISP)
```

Returns SCOL URL of site with the given name, author, type and hosting login.

```
fun __CCmodSiteUpload(verif,name,author,email,msg,lang,bitmap,
      type,scs,userISP,keycode)
fun[I S S S S S S S S S] I
```

Creates or modifies a Control Center site.

Input parameters are:

verif	upload verification indicator If positioned at 1, calls the function `CCcheckSiteUpload` defined in section CUSTOMIZATION OF CONTROL CENTER FUNCTIONS without updating the database
name	Site name
author	Site author
email	Email address associated with site (can be different from the hosting account)
msg	Site welcome message as displayed in directory
lang	Site language as displayed in directory
bitmap	Photo of site as displayed in directory
type	Site type: Cryonics, E-maginer or SCS2 The exact values are: 3 Cryonics 4 Cryonics Flash 5 E-maginer 6 SCS2 7 E-maginer Online
scs	Name of site dms file, relative to host personal directory
userISP	Hosting login
keycode	Keycode associated with site

7.2. HTTP Requests

Other Control Center functions are also accessible through GET type HTTP requests.

They are used for example, by the external hosting account management application, WebObjects.

Accepted URLs are of the form:

http://<*DNS*>/<*port*>?cmd=<*command*>&login=<*login*>&pwd=<*password*>&site=<*site*>&author=<*author*>&type=<*type*>

with:

`<DNS>`	Control Center server DNS name
`<port>`	HTTP port number listened to by the Control Center in four digit hexadecimal The port number is configurable in the *admin table.*

`<command>`	Desired Control Center functionality See available commands below
`<login>`	Hosting login
`<password>`	Hosting password
`<site>`	Site name concerned
`<author>`	Site author concerned
`<type>`	Type of site (see database table *types*).

Example:

> **http://scs.cryonetworks.com/04D2?cmd=stop&login=john&pwd=XYZ&site=beach&author=pollux&type=5**

Stops the E-maginer site '**beach**' with author '**pollux**' with hosting account authenticated by login '**john**' and password '**XYZ**'.

List of available commands

(defined in the file **./partition/locked/ctrlcenter/ccmaster.pkg**):

`cmd=delete`

Delete a Control Center site.

The following operations are performed:

- termination of site if running
- removal of files making up the site contained in the user's personal hosting directory, unless they are used by another site.

Unlike the SCOL function `__CCdelSite`, the keycode and the hosting account are not deleted.

This command is used when a user wants to delete a site to replace it with another, when the hosting period has not yet expired.

`cmd=stop`

Termination of site if running

Also makes the site invisible in the directory and inaccessible through its URL.

A "logical" deletion of the site in the Control Center.

Used to upload modifications to the Control Center and to become effective when the site is launched the next time.

`cmd=invisible`

Makes a site invisible in the directory.

The site remains accessible through its URL.

The site is also stopped if it is running.

Gives a user "pseudo-exclusive" access to their site to make online modifications.

cmd=start

Makes the site visible in the directory and accessible through its URL.

Used to put a site back online that they hid through a previous stop or invisible command .

cmd=insert

Creates a site in the database.

In addition to the previously described parameters, the URL of the command must provide:

&email=<*email*>&msg=<*msg*>&lang=<*lang*>&bitmap=<*bitmap*>&scs=<*scs*>&keycode=<*keycode*>

This command simply calls the function `__CCmodSiteUpload`() described previously in section "SCOL Requests".

8. MULTISERVER CONTROL CENTER

In its minimal configuration, the Control Center runs on a single physical machine and manages all hosted sites.

The machine's system resources (CPU, RAM, bandwidth) limit the number of concurrently running sites. For example, if a machine has 512 Mb RAM and one site uses 10 Mb, then a maximum of 50 sites can be launched concurrently.

Of course, the total number of hosted sites is much greater since we can generally assume that only a small number of sites will be accessed at the same time. If the number of sites accessed (and therefore running) increases beyond the capacity of the machine, it is necessary to expand to several machines.

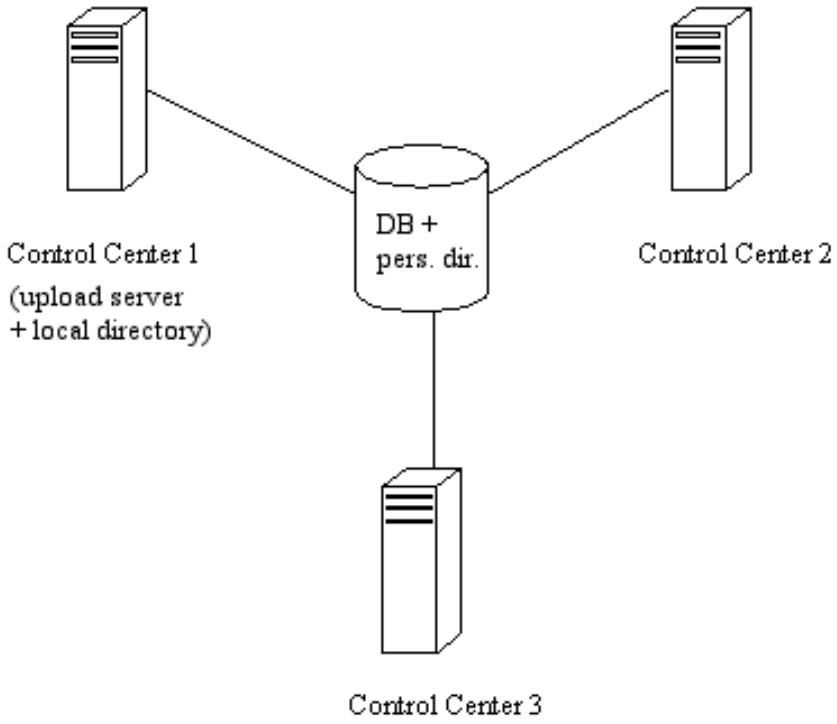
More than one Control Center (one per machine) can cooperate together.

To do this, they must share the same database and the same disk space of hosting account personal directories. Each Control Center has an identifier and only manages the sites that are attributed to it (field *CCident* of the sites table).

This identifier is also used by the Control Center to access its administration settings in the *admin* table.

Certain Control Center functions must be located on only one machine. This is so for the upload server and the local directory.

They are configured as follows:



The database can be shared by ODBC links on each machine.

The personal directories can be in a shared directory under Windows or a NFS mount point under Linux.

Two mechanisms come into play in multiserver configuration: upload balancing and load balancing.

8.1. Upload Balancing

Attributes a site to an available Control Center when it is created (first upload).

When the Control Center playing the role of upload server receives a site creation request, it attributes it the identifier of the Control Center returned by the customizable function `CCgetCCidentISP` (see section CUSTOMIZATION OF CONTROL CENTER FUNCTIONS).

In basic operation, this function gives successive identifiers of available Control Centers retrieved from the database *admin* table.

The sites are then equally distributed among the Control Centers.

8.2. Load Balancing

However, upload balancing is not enough, because the sites of a particular Control Center can be more heavily solicited than others.

If this Control Center reaches the maximum number of activatable sites because of system resources on its machine , it can no longer respond to launching requests, while other Control Centers may be inactive.

This is where load balancing comes in: a site attributed to a Control Center incapable of fulfilling the launch request , will be attributed to another Control Center.

This is performed by the customizable function `CCgetNextId` (see section CUSTOMIZATION OF CONTROL CENTER FUNCTIONS).

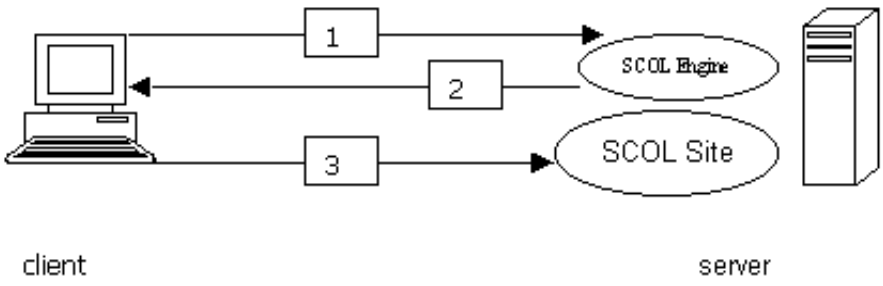
In basic operation, this function returns the next Control Center identifier among those available in the *admin* table, relative to the identifier of the Control Center calling the function (the ordering is simply lexicographical).

We do not know if the next Control Center will be able to fulfill the request to launch the site.

But if necessary, using the same mechanism, it can in turn attribute the site to another Control Center.

This continues until some Control Center can fulfill the request.

This balancing is based on the connection mechanism of a client to a SCOL Server:



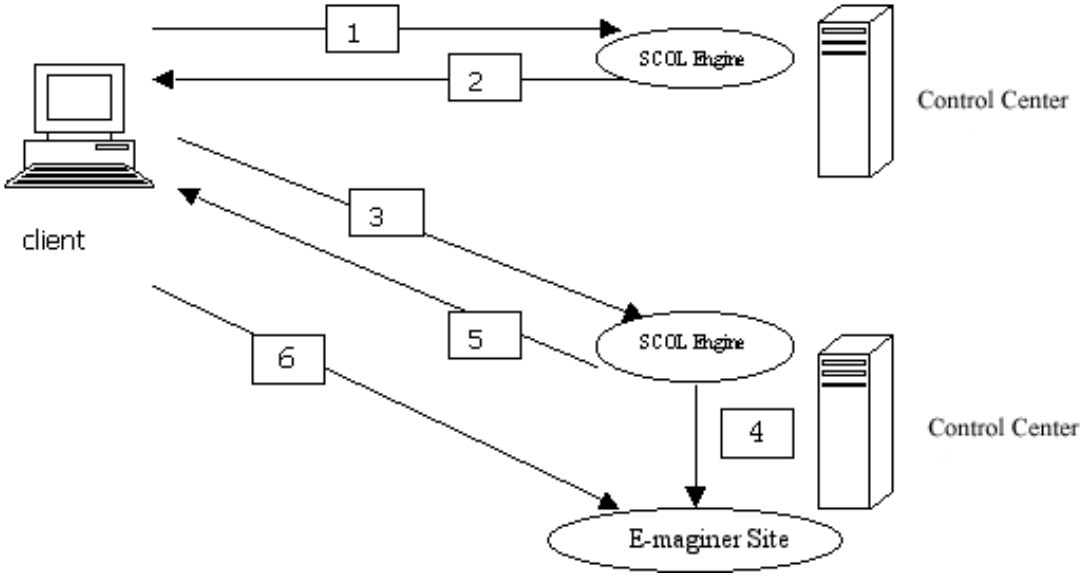
1 A client requests a connection to a SCOL site (Cryopolis for example): **scol://scol.cryopolis.com:Cryopolis**

They first refer to the SCOL Engine of the scol.cryopolis.com machine to request resolution of the Cryopolis service name.

2 The server SCOL Engine answers by giving the complete URL to reach the desired site: **scol://164.109.24.86:4008**

3 The client connects directly to the IP address and port returned by the SCOL Engine.

In the balancing case in question, this mechanism is used to perform a redirection to another Control Center:



1 A client requests a connection to an E-maginer site: **scol://dns1:directory.author.name**

They first refer to the Control Center of the dns1 machine to request resolution of the service name directory.author.name.

2 We assume that Control Center 1 is not capable of launching a new site because it has reached its maximum capacity. It then attributes the site to Control Center 2 and returns to the client the complete URL required to reach it: **scol://dns2:directory.author.name**

3 The client, still following the same logic, refers to Control Center 2 and requests resolution of the service name: directory.author.name.

4 We suppose that Control Center 2 can fulfill the request.

It launches the corresponding SCOL site, dynamically attributing it a port number in a given range.

5 It then returns to the client the complete URL required to reach it: **scol://ip2:2100**

6 The client connects directly to the IP address and port returned by Control Center 2.

8.3. Setup

All that is needed is to add a machine with SCOL server and insert a line in the *admin* table of the database with a new Control Center identifier.

It is not necessary to stop or reconfigure the other Control Centers.

They will immediately be aware of the new one through the *admin* table and will include it in their balancing mechanisms.

9. APPENDIX

9.1. Physical Data Model

admin	
CCident	varchar(50)
minport	numeric
maxport	numeric
nbmaxsites	numeric
HTTPport	char(4)
HTTPpasswd	varchar(50)
UPLOADCNXport	numeric
UPLOADSCSport	numeric
UPLOADEMport	numeric
capacity	numeric
closedelay	numeric
DNSname	varchar(50)

users	
userISP	varchar(50)
passISP	varchar(50)
dirISP	varchar(100)
maxDirSize	numeric
firstName	varchar(50)
lastName	varchar(50)
streetAddress	varchar(255)
city	varchar(50)
state	varchar(50)
zipCod	varchar(20)
country	varchar(3)
gender	char(1)
birthDate	varchar(10)
interest	varchar(255)
email	varchar(100)
question	varchar(255)
answer	varchar(255)

keycode	
keycode	varchar(50)
userISP	varchar(50)
datcre	char(19)
period	numeric
maxSites	numeric
mailsent	char(1)
mailperiod	numeric

files	
site	varchar(200)
fic	varchar(255)

types	
type	numeric
typename	varchar(50)
partnumber	numeric
partrefine	varchar(50)
subcat	numeric
nbport	numeric
loadfile	varchar(255)

sites	
author	varchar(50)
name	varchar(50)
locdir	varchar(50)
category	numeric
email	varchar(50)
password	varchar(50)
proxy	varchar(50)
bitmap	varchar(20)
msg	varchar(255)
urlscol	varchar(127)
urlhttp	varchar(127)
timeout	numeric
nbcnx	numeric
creation	numeric
livinguntil	numeric
lang	varchar(50)
type	numeric
subcat	numeric
subtype	numeric
scs	varchar(255)
capacity	numeric
userISP	varchar(50)
CCident	varchar(50)
datcre	char(19)
datmod	char(19)
reg	char(1)
keycode	varchar(50)
DMSid	numeric
closedelay	numeric
nblaunch	numeric
autostart	char(1)
state	char(1)
port	numeric
nbport	numeric
loadfile	varchar(255)
scolfile	varchar(255)

categories	
id	numeric
locdir	varchar(50)
category	varchar(50)
mother	numeric
url	varchar(50)
urlforced	varchar(50)

ACL	
Login	varchar(50)
Password	varchar(50)
FName	varchar(50)
LName	varchar(50)

9.2. Table *admin*

Contains the Control Center administration settings.

In the multiserver case, there is one line per Control Center.

Column	Obl	Comment
CCident(*)	Yes	Control Center identifier.
minport(*)	Yes	Lowest port number attributed to site launching
maxport(*)	Yes	Highest port number attributed to site launching
nbmaxsites(*)	Yes	Max no. of simultaneously active sites
HTTPport(*)	No	Port number for HTTP API interface in four digit hexadecimal
HTTPpasswd(*)	No	Unused
UPLOADCNXport(*)	No	Port number for upload of Cryonics sites
UPLOADSCSport(*)	No	Port number for upload of SCS2 sites
UPLOADEMport(*)	No	Port number for upload of E-maginer sites
capacity(*)	No	Default max nb. of clients for a site (when the *capacity* field of the *sites* table is not filled in)
closedelay(*)	No	Default time in seconds after which a site is stopped when there are no more visitors (when the field *closedelay* of the table *sites* is not filled in)
DNSname(*)	No	DNS name of the Control Center machine

9.3. Table *users*

Contains hosting account properties

Column	Obl	Comment
userISP(*)	Yes	User's login
passISP(*)	No	User password
dirISP(*)	Yes	Personal directory of the user, relative to the first SCOL partition
maxDirSize(*)	No	Unused
firstName(*)	No	Purely informative optional data
lastName(*)	No	Purely informative optional data
streetAddress(*)	No	Purely informative optional data
city(*)	No	Purely informative optional data
state(*)	No	Purely informative optional data
zipCod(*)	No	Purely informative optional data

(*) "static" column to be filled in by means external to the Control Center

Column	Obl	Comment
country(*)	No	2 letter user country code (FR, GB, etc.); used to localize mail warning of site expiry
gender(*)	No	Purely informative optional data
birthDate(*)	No	Purely informative optional data
interest(*)	No	Purely informative optional data
email(*)	No	Email address for sending warning mail before site expiry
question(*)	No	Can be used to recover forgotten password
answer(*)	No	Can be used to recover forgotten password

9.4. Table *keyCode*

Contains hosting product properties.

The term keyCode is somewhat of a misnomer.

It corresponds to Cryonetwork usage where a product is identified by its keyCode.

Another solution is to associate a product directly to a hosting account.

In this case, the *keyCode* field of the table is the hosting login itself. A hosting account then only manages a single type of product.

Column	Obl	Comment
keyCode(*)	Yes	Product identifier
userISP(*)	Yes	Hosting login associated with product
datcre	No	Creation date of first site of this product uploaded, in format "YYYY-MM-DD hh:mn:ss" The validity period is counted starting from this date
period(*)	No	Validity period of product in days A value of –1 or NULL means infinite
maxSites(*)	No	Maximum number sites authorized for this product A value of –1 or NULL means infinite
mailsent	No	Indicator that mail was sent warning that product has expired O: mail already sent N: mail not yet sent
mailperiod(*)	No	Period in days before product expiration when a warning mail must be sent

(*) "static" column to be filled in by means external to the Control Center

9.5. Table *sites*

List of sites managed by the Control Center(s).

Column	Obl	Comment
author	Yes	Site author
name	Yes	Site name
locdir	Yes	Name of the local directory
category	Yes	Category of hosting provider (always 0)
email	No	Email address associated with site
password	No	Unused
proxy	No	Unused
bitmap	No	SCOL signature of the welcome image file displayed in the directory of sites
msg	No	Welcome image displayed in the directory of sites
urlscol	No	URL for site in the form **scol://machine:locdir.author.name**
urlhttp	No	Unused
timeout	No	Unused
nbcnx	No	Unused
creation	No	Time created (in seconds since 01/01/1970)
livinguntil	No	Validity date in directory (time created + 10 years)
lang	No	Site language
type	No	Site type (3=Cryonics, 4=Cryonics Flash, 5=E-maginer, 6=SCS2, 7=E-maginer Online)
subcat	No	Subcategory in directory tree (0 by default)
subtype	No	Unused (same as type)
scs	No	SCS file name (.scs or .dms)
capacity	No	Max nb. of clients for the site (by default *capacity* of the *admin* table)
userISP	No	Hosting login
CCident	Yes	Identifier of Control Center managing the site
datcre	No	Creation date for the site in format "YYYY-MM-DD hh:mn:ss"
datmod	No	Modification date for the site in format "YYYY-MM-DD hh:mn:ss"
reg	No	Site visibility indicator 0=invisible, inaccessible 1=visible, accessible 2=invisible, accessible

Column	Obl	Comment
keyCode	No	Product identifier
DMSid	No	Unique identifier of SCOL site
closedelay	No	Default time in seconds after which a site is stopped when there are no more visitors (by default *closedelay* of the table *admin*)
nblaunch	No	Counts number of sites running
autostart	No	Indicator of automatic launch when Control Center starts 0=no automatic launch 1=automatic launch
state	No	site status 0=site not launched 1=site launched
port	No	Attributes a static port to the site. If filled in, the Control Center will not dynamically look for a free port in the range defined by the table *admin*.
nbport	No	Number of free consecutive ports the Control Center must look for in the range defined by the table *admin*.
loadfile	No	Gives the name of a file containing SCOL DMS file loading instructions. This is used to associate the site with a DMS version. Three versions are predefined in the directory **./partition/locked/ctrlcenter/script**: HDMS, DHDMS and EMDHDMS (used for E-maginer)
scolfile	No	Gives the name of a SCOL script file to be executed to launch the site. The usual Control Center functions do not apply to this type of site: startup on connection of first visitor, "refine" of the execution partition on launching, close when no more visitors, etc.

9.6. Table *categories*

List of categories of the local directory.

This table must at least contain the category 0 (id=0, locdir=NULL, category='core', mother=-1, url=NULL, urlforced=NULL).

Column	Obl	Comment
id(*)	Yes	Category identifier
locdir(*)	Yes	Name of the local directory

(*) "static" column to be filled in by means external to the Control Center

Column	Obl	Comment
category(*)	No	Category name
mother(*)	Yes	Identifier of mother category (tree structure)
url(*)	No	Unused
urlforced(*)	No	Unused

9.7. Table *types*

Lists properties of site types.

The *type* field has one of the following values: 3=Cryonics, 4=Cryonics Flash, 5=E-maginer, 6=SCS2, 7=E-maginer Online

Column	Obl	Comment
type(*)	Yes	Site type
typename(*)	No	Type name; this name is part of the warning mail file sent before site expiration
partnumber(*)	No	SCOL partition number to be refined to obtain the common partition specific to this type of site (**./common**)
partrefine(*)	No	Name of subdirectory of the common partition specific to this type of site
subcat	No	Subcategory in local directory tree
nbport	No	Number of free consecutive ports the Control Center must look for in the range defined by the table *admin*.
loadfile	No	Gives the name of a file containing SCOL DMS file loading instructions. This is used to associate the type with a DMS version. Three versions are predefined in the directory **./partition/locked/ctrlcenter/script**: HDMS, DHDMS and EMDHDMS (used for E-maginer)

9.8. Table *files*

List of files created in personal directories of hosting accounts.

Control Center uses this table to delete files when it deletes an expired site.

Column	Obl	Comment
site	Yes	Site name as locdir.author.name
fic	Yes	File name including personal directory of the hosting account

9.9. Configuration of Database Connection in the Control Center

The file **./partition/locked/sspadmin/ccdbconfig.ini** contains the following lines, that can be customized:

```
DSN SCOLserver
LOGIN admin
PASSWORD
```

Entry	Description
DSN	Name of the Control Center database defined in ODBC
LOGIN	Access login to database.
PASSWORD	Access password to database

9.10. Configuration of Database Connection in Local Directory

The file **./partition/dirloc/dir.dmi** contains the following lines, that can be customized:

```
database SCOLserver
login admin
password
tables categories\sites
```

Entry	Description
database	Name of the sites database defined in ODBC
login	Access login to sites database
password	Access password to sites database
tables	Name of tables categories and sites (separated by spaces)

9.11. Configuration of Local Directory Authentification in Global Directory

The global directory performs login/password authentification on local directories that register in one of its categories.

The file **./partition/dirloc/dir.dmi** contains the following lines, that can be customized:

```
master scol.cryonetworks.com:3101
mlogin mylogin
mpassword mypassword
```

Entry	Description
master	Address of global directory: must be **scol.cryonetworks.com:3101**
mlogin	Authentification login
mpassword	Password attributed by Cryonetworks

9.12. Configuration of SCOL Name in Local Directory

The local directory has a SCOL name that can be reached by the URL **scol://server:directory_name**

For the sake of consistency, this name must be the same as that provided in the database table *categories* (field *locdir*).

It is defined in the file **./partition/dirloc/dir.scs** in the entry "`name`".