Administration of AVEVA Marine Environment

AVEVA Marine Administrator’s Guide

Contents **Page**

AVEVA Marine Environment

1 Administration of the Environment

1.1 Program Environment

1.1.1 System Directories

The system is always installed in a home directory, the 'root' directory. This contains a number of different files arranged in a subdirectory structure.

Subdirectories

`Config:`	Used for storage of the current project configurations & settings.
`Customise:`	Project independent example files.
`Documentation:`	Home of the on-line HTMLHelp documentation files.
`Etc:`	Message files, syntax files and Symbol files.
`Projects:`	Contains the sample projects which may be installed together with the system.
`Template project:`	Contains the template project that is used when creating new projects.
`Vitesse:`	Vitesse source code files.
`Python:`	Python source code files.

1.1.2 System Files

The Bin, Config, and Documentation directories contain the system installation and should not be manually edited.

1.1.3 Program Execution - the Job Launcher

A session refers to an execution of a single application, i.e. any executable. Examples of sessions are Hull Extract Scheme, Hull Parts List and Ppanparts.

The environment refers to all the environment variables listed in the environment table. The environment table specific for AVEVA Marine is separated from Windows system and user variables.

Sessions together with their corresponding input- and output parameters are called Jobs. A job file refers to an instance of a specific job. A session that is executed has a corresponding job file that contains information about the application, the input/output parameters, date of execution etc. Job files are considered as temporary; i.e. they hold no actual data, but only information about the parameters of that specific job. They are stored in a subdirectory called jobs below the directory given by the Windows environment variable %PDMSWK%.

In practice, job files are invisible to the end user and are used e.g. by the Log Viewer to view input/output parameters and to follow up the execution status.

- **Applications**

All applications (except for utility programs) are defined in an xml file referred to as the application definition file. The application definition file is located in the SB_SYSTEM directory and is called `MarJobs.xml`. For each application there is an <application> xml node with the following parameters:

- Application name
- Short application name (used e.g. to give name to log files)
- Name of the executable file
- Input parameters (description & type)
- Output parameters (description, type & file name extension)

Example:

```
 <application>
<name>Hull Plane Part Gen.</name>
       <short_name>ppanparts</short_name>
       <executable>sf416d.exe</executable>
       <inputs>
          <input>Input Selection</input>
       </inputs>
       <outputs>
          <output extension="log">Check of input selection</output>
          <output extension="lst">List file</output>
       </outputs>
</application>
```

The names of the output files are chosen by the job launcher at run time, but the file name extensions are defined here.

Normally, the job launcher searches for executables in the %PDMSEXE% directory. This is true if the file name in the `<executable>` node is given without path, e.g.:

```
<executable>sp600.exe</executable>
```

To start an application from another location than %PDMSEXE%, then the full path has to be given in the `<executable>` node, e.g.:

```
<executable>c:\winnt\system32\notepad.exe</executable>
```

- **List of Executables**

The following is a list of executable programs and common object modules. Please note that every installation does not necessarily have access to all the modules. Further, some installations may have some additional modules. The actual contents depends on licenses, differences in software packs, customer developed programs etc.

Executables.xls

- **Start Method**

  To give each session a private copy of the environment and to supply the application with all its input data, each session is started by the Job Launcher. This is achieved by having a special starter program (`marjobstarter.exe`) responsible for starting sessions.

  This command line will create and execute a new instance of the batch program **Hull Parts List**.

```
marjobstarter.exe -input c:\input.dat -application "Hull
Parts List".
```

  The job launcher waits in the background for the application to terminate and sends notification messages to the log viewer about the execution status.

- **Running Console Applications**

  The Job Launcher can also start console applications (e.g. `sa004.exe`). There are a number of ways how to run console applications:

  1. Read input from user and display output in the console window. This is the *normal* way.
  2. Read input (stdin) from a supplied file. This alternative can be useful when running a console application in batch without any user interaction. In this case, it's probably desired to also redirect the console ouput (stdout) to a file.

  To cover both of these cases, two new command line arguments have been added to `tbstartjob.exe`:

  - `-stdin_file <fname>`
    This argument tells the application to read standard input (stdin) from the specified file (<fname>).
  - `-console_stdout`
    This argument tells the application to write standard output (stdout) on the console. This should only be used for console applications (e.g. `sa004.exe`). If this argument is omitted, the default behaviour is to redirect stdout to a file (which is showed in the Log Viewer).

  To make applications visible to the job launcher, the applications have to be defined in `MarJobs.xml`. Below is an example of a MarJobs.xml file that defines `sa004.exe`.

```
<?xml version="1.0" standalone="yes"?>
<applications>
…
  <system name="User">
    <application>
      <name>sa004</name>
      <short_name>sa004</short_name>
      <executable>sa004.exe</executable>
    </application>
</system>
…
```

```
</applications>
```

Example:

Run `sa004.exe`, show output on the console and let user do the interaction:

```
marjobstarter.exe -console_stdout -application sa004
```

Run `sa004.exe`, show output on the console and supply an input file:

```
marjobstarter.exe -console_stdout -stdin_file c:\input.txt -application
sa004
```

Run `sa004.exe`, redirect stdout to a file and supply an input file:

```
marjobstarter.exe -stdin_file c:\input.txt -application sa004
```

1.2 Projects

The collection of data used during the work with a ship design is called a project. The data constituting a project is stored in a number of data banks and files. These files are arranged according to a certain directory structure.

In addition to the section *Setting Up a New Project* in the *Admin User Guide*, there is a Windows environment variable called `<proj>MAR` which points to the location of additional Marine project data. For example, in a project called ABC, this variable would be ABCMAR.

The `<proj>MAR` directory structure consists of subdirectories containing user created data files (input to programs, output from programs, etc.) and other data files (data banks, default files, etc.) specific for a Marine project. The standard environment consists of the following set of subdirectories:

Subdirectory	**Example of Contents**
`<Proj>\MAR\dat`	Miscellaneous input data files.
`<Proj>\MAR\def`	Defaults and standards (e.g. default files).
`<Proj>\MAR\lst`	Output and log files.
`<Proj>\MAR\nc`	Output to NC machines.
`<Proj>\MAR\sch`	Structure generation schemes, curved panel schemes.
`<Proj>\MAR\navarch`	Surface files

Default Files

Default files contain specific and detailed settings for individual applications. A default file is a text file that is structured using keywords and values. Default files are accessible by applications using an environment variable that points out the directory path and file name of the default file. One example of an environment variables referring to a default file is:

`SBD_DEF1`	Drafting application default file.

IP Files

IP files are analogous to default files used within the Hull applications.

Settings Objects

Some application specific setups are stored as objects in data banks.

1.3 Runtime System

1.3.1 Environment Variables

A separate Excel sheet contains a complete list of environment variables. (All variables are not necessarily valid for all type of customers or all versions of VANTAGE Marine.)

Link to the Excel sheet: All_logical_names.xls

1.3.2 Surface Server

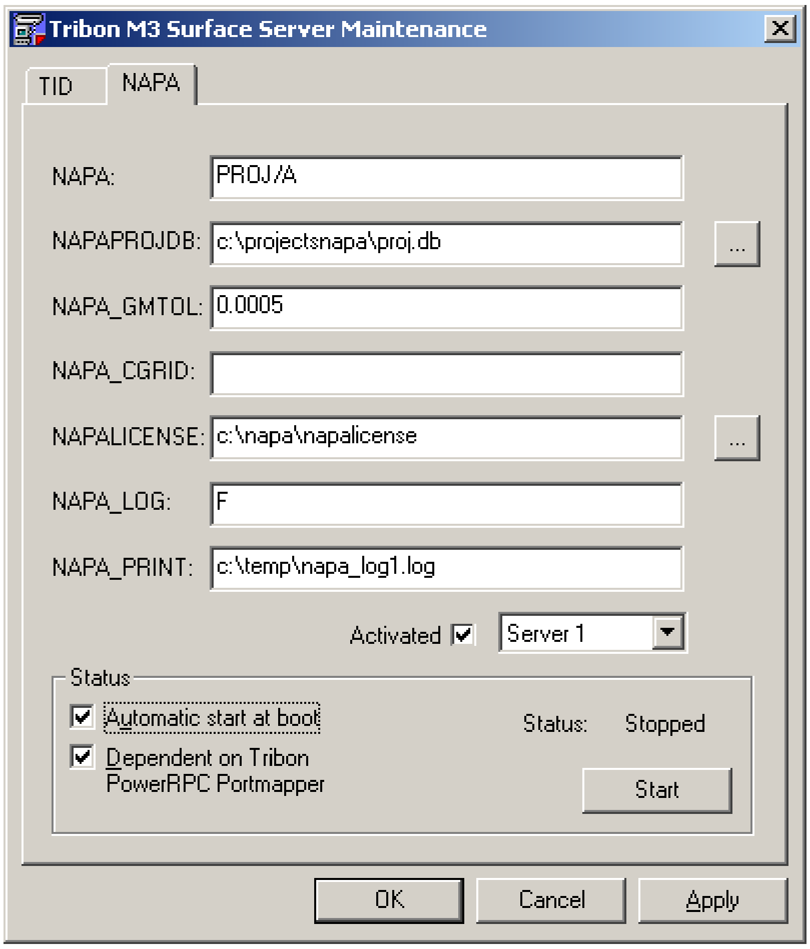
The access to an external hull surface system (NAPA) is done via a surface server, which is an application running on a server machine, and to which applications connect to retrieve surface information. The server must execute on the network node where the surface database is residing, and the applications communicates with the server via ONC RPC in a similar way as the database servers. Access to a surface database is always handled through the Surface Server meaning that also in a stand-alone single client Windows installation, a Surface Server must be installed on the machine. In this case the same machine will act both as the server and the client.

Normally, the surface is released to Dabacon when running Initial Design and thus no surface server will be required. However, for backwards compatibility reasons, it's still possible to use an initial design surface server. The chosen surface server(s) will be installed as a Windows service. At installation time, the service is configured to be started manually, since it will need to be properly setup before use. This configuration is done with a tool called Surface Server Maintenance (`marsurfservmaint.exe`).

Note: Please note that this maintenance program has to be executed on the actual machine where the Surface Server service is installed.

- #### NAPA Surface Server

In case the NAPA Surface Server was selected during installation, a second page will appear in the program, allowing configuration of that server.


Tribon M3 Surface Server Maintenance

TID | NAPA

NAPA: PROJ/A
NAPAPROJDB: c:\projectsnapa\proj.db
NAPA_GMTOL: 0.0005
NAPA_CGRID:
NAPALICENSE: c:\napa\napalicense
NAPA_LOG: F
NAPA_PRINT: c:\temp\napa_log1.log

Activated ☑ Server 1

Status
☑ Automatic start at boot
☑ Dependent on Tribon PowerRPC Portmapper
Status: Stopped

Figure 1:1. The Setup page for the NAPA Surface Server

The NAPA surface server service can handle up to 10 NAPA surface servers with individual settings. This functionality is provided to allow access to several NAPA surface projects on one server host. As the NAPA surface server service is started, only those NAPA Surface servers defined as Activated will be started.

NAPA, NAPAPROJDB, NAPA_GMTOL, NAPA_CGRID, NAPALICENSE, NAPA_LOG, NAPA _PRINT

These are environment variables used by NAPA. Please consult the NAPA documentation for the explanation of these variables.

Note: In order to be able to run the NAPA surface server you have to obtain a license from NAPA. This is not delivered together with the system. Not more than one activated server should be connected to the same NAPA project and the log-file defined by the NAPA variable NAPA_PRINT should not be shared between different servers.

Automatic start at boot

As mentioned earlier, the service is configured for manual start at install time. When the service has been properly configured, it is more convenient to have it started automatically at boot time.

Dependent on PowerRPC Portmapper

Normally this box should always be checked. The only reason to uncheck the box is if a portmapper is already running on the machine, in which case the one delivered with the system will not install properly. If the box is not checked, it is strongly recommended to configure the surface service to be started manually instead of automatic. This is because the system in this case cannot make sure that the portmapper is running before trying to start the surface service.

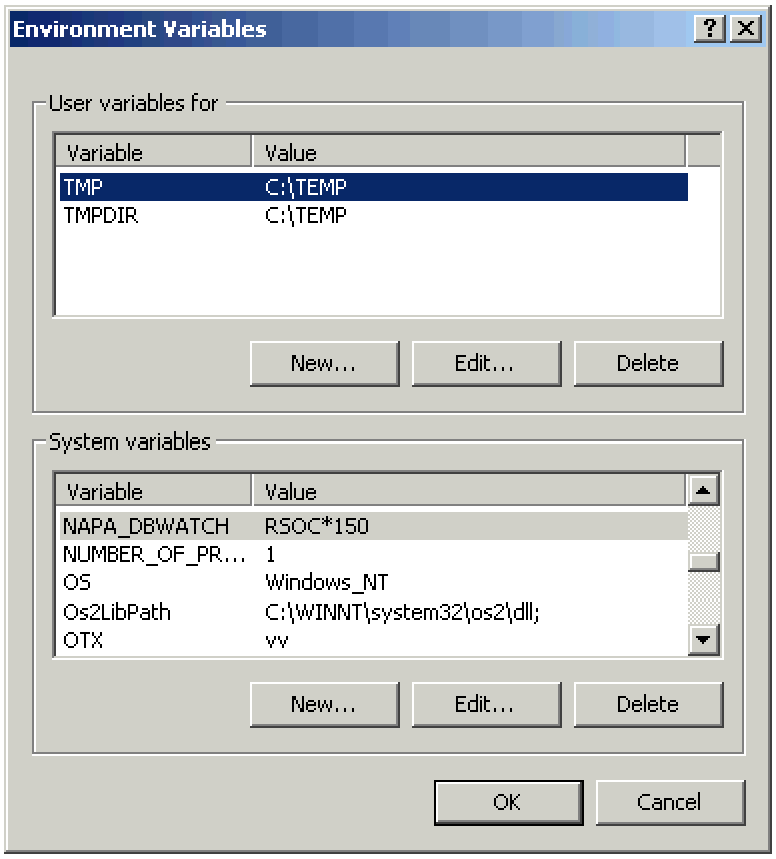
Start/Stop

If the settings are satisfactory, and the service is just stopped for some reason, the start button can be used. If any settings have been changed though, either apply or **OK** should be used instead. In this case, the current settings will be stored in the Windows registry, the service will be stopped and then restarted to reflect the new settings.

If the service is running, it is possible to stop it with this button.

- **Additional Environment Variables**

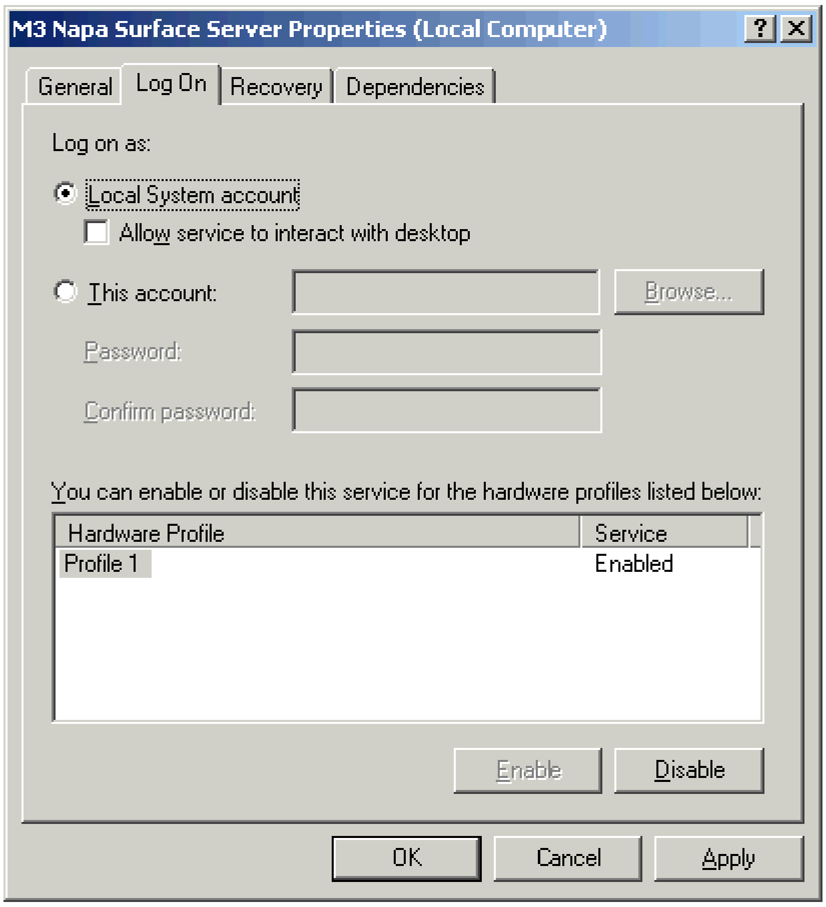
Additional environment variables (not handled in the control applet user interface) affecting the NAPA functionality e.g. like NAPA_DBWATCH (see *Option NAPA_DBWATCH*), must be set in the System Variables section of the Windows environment variables.



Variable	Value
TMP	C:\TEMP
TMPDIR	C:\TEMP

Variable	Value
NAPA_DBWATCH	RSOC*150
NUMBER_OF_PR...	1
OS	Windows_NT
Os2LibPath	C:\WINNT\system32\os2\dll;
OTX	vv

If the M3 Napa Surface Server service (Start -> Settings -> Control Panel ->Administrative Tools -> Services -> Properties of the M3 Napa Surface Server service) is running under Local System Account, the computer must be restarted in order to make changes of such variables to take effect.


M3 Napa Surface Server Properties (Local Computer)

General | Log On | Recovery | Dependencies

Log on as:

(●) Local System account

[ ] Allow service to interact with desktop

( ) This account: [ ] Browse...

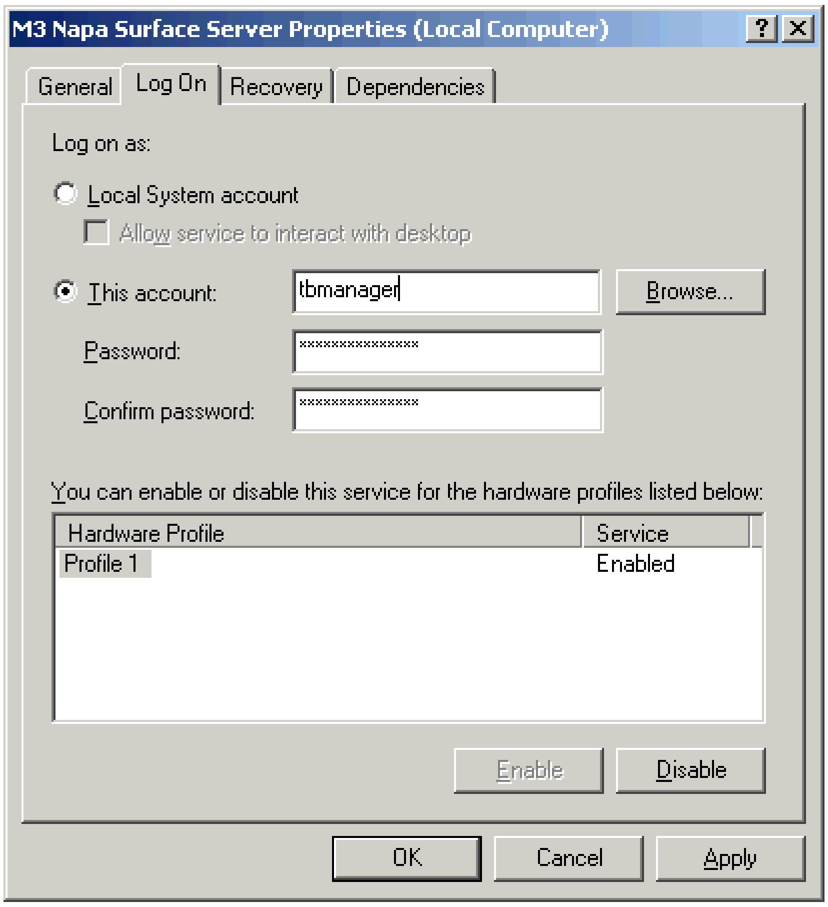
Password: [ ]

Confirm password: [ ]

You can enable or disable this service for the hardware profiles listed below:

Hardware Profile	Service
Profile 1	Enabled

If the service is running under any other account the M3 Napa Surface server service, only the service itself needs to be restarted in order to make changes of such variables to take effect.


M3 Napa Surface Server Properties (Local Computer)

Log on as:

- Local System account
  - [ ] Allow service to interact with desktop
- [✓] This account: tbmanager
  - Password: ****************
  - Confirm password: ****************

You can enable or disable this service for the hardware profiles listed below:

Hardware Profile	Service
Profile 1	Enabled

Option NAPA_DBWATCH

There is a new option called NAPA_DBWATCH to detect changes of geometry that has been made by another NAPA run while the surface server is running. This feature is controlled by the environmental variable NAPA_DBWATCH. All alternatives of the NAPA function GM.DBWATCH are available. At specified intervals, the dates of the currently used objects (of the specified types) are compared with the dates in the NAPA database and if a newer date is found, the object is updated.

The default behavior, i.e. if the variable is unset, is consistent with setting the variable to NAPA_DBWATCH=RSOC*300 which checks rooms (=R), surfaces (=S), surface objects (=O) and curves (=C) and where the time interval between the checks is 300 seconds.

If NAPA_DBWATCH=OFF, the server works as before and a restart of the server is needed to make the changes visible to the interface process.

- **Connect to Surface Server**

  There are some environment variables that rule how applications connect to the surface server of choice. These should be set in the `d065<project>.sbd` file.

  - SBB_SURFACE_SYSTEM should be set to bmt_1 in case the TID surface system is used, and to napa_1 if the NAPA surface system is used
  - SBB_SURFACE_SERVER_HOST should be set to the name of the server machine where the surface server is running.

- SBB_SURFACE_SERVER_NUMBER should be set to the appropriate NAPA surface server number (1 - 10). If the variable is not set, a value of 1 is assumed.

  This variable is not relevant when using the TID surface server.

Example of the entries in a `d065<project>.sbd` file:

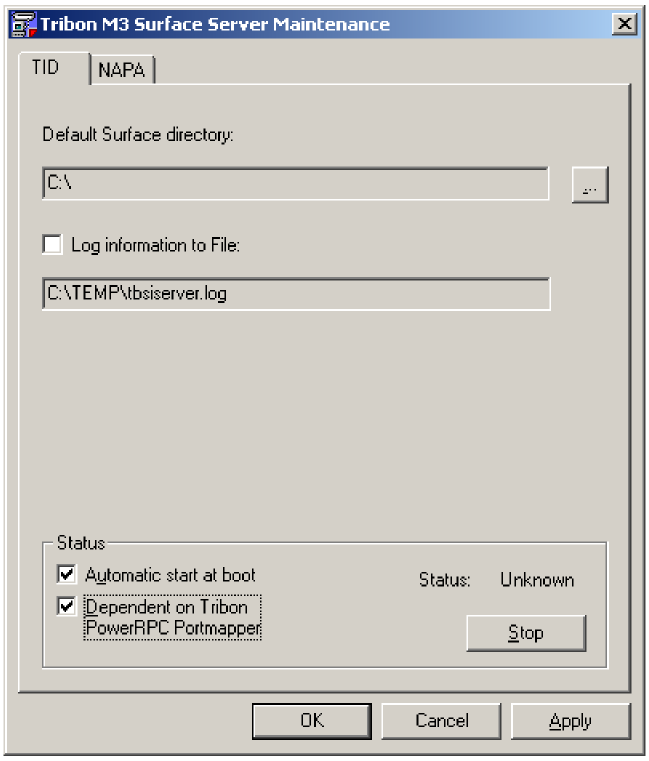
Example:

```
SBB_SURFACE_SYSTEM bmt_1
SBB_SURFACE_SERVER_HOST the_surface_server
```

- **Initial Design Surface Server**

The configuration of a Initial Design server is done in the first page of the setup program.

Default Surface Directory	Optional parameter. In case the environment variable SB_NAVARCH is set in the project, the surface server will look for the surface database in the directory indicated by that variable. In case the variable is not set, the Default Surface Directory is the place where the server will look for the surface databases. **Note:** It is only possible for the server to handle surfaces from one directory at the time. This means that if surfaces are to be kept in different locations, and are to be used at the same time, more than one surface server machine has to be used. This is because a server machine can only host one surface server at a time.
Log information to file	If checked, the server will log messages to a file named tbgiserver.log in the directory given by the Windows environment variable TEMP.
Automatic start at boot	As mentioned earlier, the service is configured for manual start at install time. When the service has been properly configured, it is more convenient to have it started automatically at boot time. (If the demo project ds4 is selected at installation the Surface server will be started and configured. However, for automatic start at boot. The customer must change the configuration manually)


Tribon M3 Surface Server Maintenance

TID | NAPA

Default Surface directory: C:\

☐ Log information to File: C:\TEMP\tbsiserver.log

Status
☑ Automatic start at boot
☑ Dependent on Tribon PowerRPC Portmapper
Status: Unknown

Figure 1:2. The setup page for the Initial Design Surface Server (`marsurfservmaint.exe`*)*

Dependent on PowerRPC Portmapper

Normally this box should always be checked. The only reason to uncheck the box is if a portmapper is already running on the machine, in which case the one delivered will not install properly. If the box is not checked, it is strongly recommended to configure the surface service to be started manually instead of automatic. This is because the system in this case cannot make sure that the portmapper is running before trying to start the surface service.

Start/Stop

If the settings are satisfactory, and the service is just stopped for some reason, the start button can be used. If any settings have been changed though, either apply or **OK** should be used instead. In this case, the current settings will be stored in the Windows registry, the service will be stopped and then restarted to reflect the new settings.

If the service is running, it is possible to stop it with this button.

- **Connect to Surface Server**

There are some environment variables that rule how applications connect to the surface server of choice. These should be set in the `d065<project>.sbd` file.

- SBB_SURFACE_SYSTEM should be set to bmt_1
- SBB_SURFACE_SERVER_HOST should be set to the name of the server machine where the surface server is running.

Example of the entries in a `d065<project>.sbd` file:

Example:

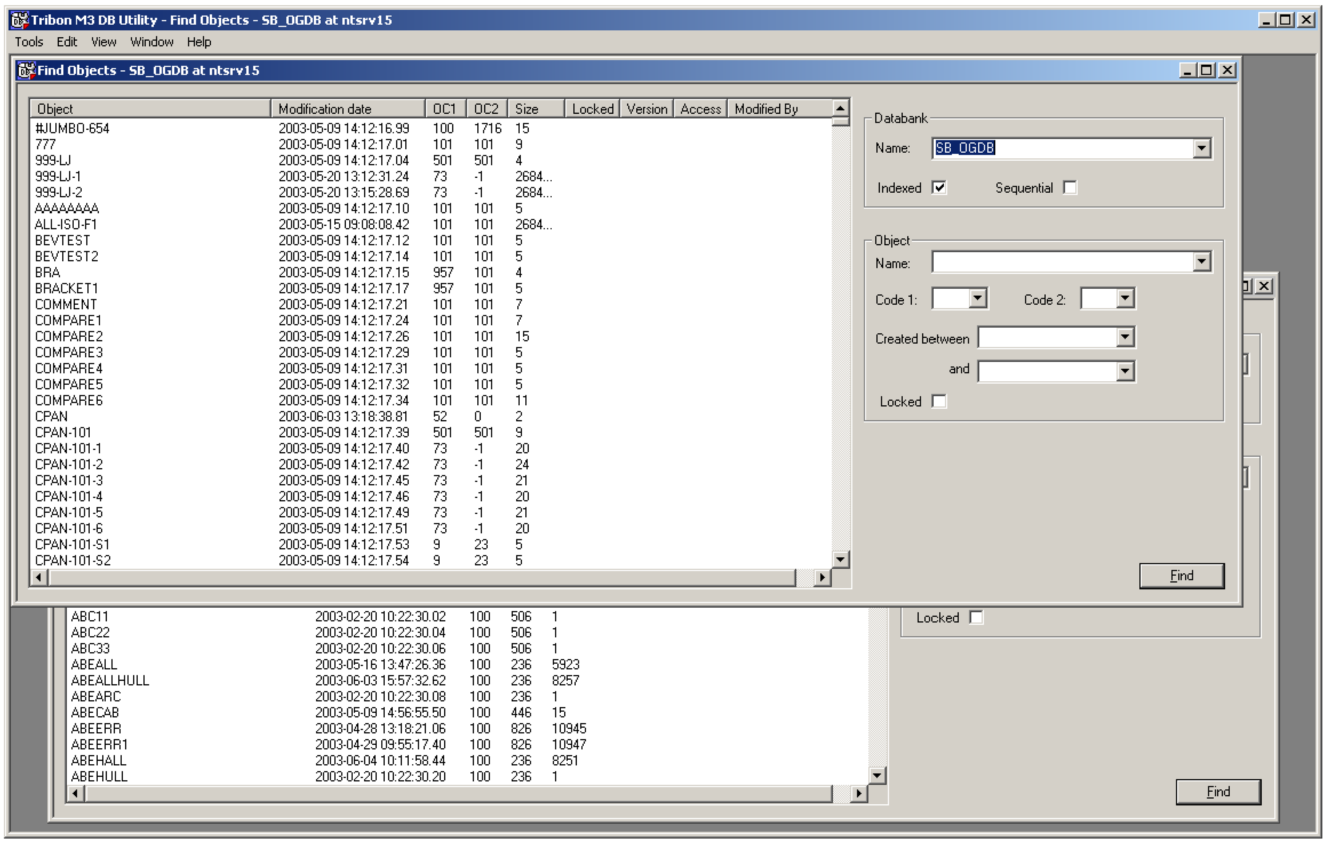
```
SBB_SURFACE_SYSTEM bmt_1
SBB_SURFACE_SERVER_HOST the_surface_server
```

1.3.3 Data Bank Utility

The program is a utility program for objects on data banks.

• Find Objects

This part of the utility program has a multiple-document interface that maintains multiple child windows within a parent window. To each child window it is possible to connect any data bank and thereby perform operations for objects in that data bank Through cut - copy and paste operations it is possible to perform operations between different child windows i.e. between data banks.


Tribon M3 DB Utility - Find Objects - SB_OGDB at ntsrv15
Tools
Edit
View
Window
Help
Find Objects - SB_OGDB at ntsrv15
Find
Find

Object	Modification date	OC1	OC2	Size	Locked	Version	Access	Modified By
#JUMBO-654	2003-05-09 14:12:16.99	100	1716	15				
777	2003-05-09 14:12:17.01	101	101	9				
999-LJ	2003-05-09 14:12:17.04	501	501	4				
999-LJ-1	2003-05-20 13:12:31.24	73	-1	2684...				
999-LJ-2	2003-05-20 13:15:28.69	73	-1	2684...				
AAAAAAAA	2003-05-09 14:12:17.10	101	101	5				
ALL-ISO-F1	2003-05-15 09:08:08.42	101	101	2684...				
BEVTEST	2003-05-09 14:12:17.12	101	101	5				
BEVTEST2	2003-05-09 14:12:17.14	101	101	5				
BRA	2003-05-09 14:12:17.15	957	101	4				
BRACKET1	2003-05-09 14:12:17.17	957	101	5				
COMMENT	2003-05-09 14:12:17.21	101	101	7				
COMPARE1	2003-05-09 14:12:17.24	101	101	7				
COMPARE2	2003-05-09 14:12:17.26	101	101	15				
COMPARE3	2003-05-09 14:12:17.29	101	101	5				
COMPARE4	2003-05-09 14:12:17.31	101	101	5				
COMPARE5	2003-05-09 14:12:17.32	101	101	5				
COMPARE6	2003-05-09 14:12:17.34	101	101	11				
CPAN	2003-06-03 13:18:38.81	52	0	2				
CPAN-101	2003-05-09 14:12:17.39	501	501	9				
CPAN-101-1	2003-05-09 14:12:17.40	73	-1	20				
CPAN-101-2	2003-05-09 14:12:17.42	73	-1	24				
CPAN-101-3	2003-05-09 14:12:17.45	73	-1	21				
CPAN-101-4	2003-05-09 14:12:17.46	73	-1	20				
CPAN-101-5	2003-05-09 14:12:17.49	73	-1	21				
CPAN-101-6	2003-05-09 14:12:17.51	73	-1	20				
CPAN-101-S1	2003-05-09 14:12:17.53	9	23	5				
CPAN-101-S2	2003-05-09 14:12:17.54	9	23	5				

Databank
Name: SB_OGDB
Indexed [✓] Sequential [ ]

Object
Name:
Code 1:
Code 2:
Created between
and
Locked [ ]

ABC11	2003-02-20 10:22:30.02	100	506	1
ABC22	2003-02-20 10:22:30.04	100	506	1
ABC33	2003-02-20 10:22:30.06	100	506	1
ABEALL	2003-05-16 13:47:26.36	100	236	5923
ABEALLHULL	2003-06-03 15:57:32.62	100	236	8257
ABEARC	2003-02-20 10:22:30.08	100	236	1
ABECAB	2003-05-09 14:56:55.50	100	446	15
ABEERR	2003-04-28 13:18:21.06	100	826	10945
ABEERR1	2003-04-29 09:55:17.40	100	826	10947
ABEHALL	2003-06-04 10:11:58.44	100	236	8251
ABEHULL	2003-02-20 10:22:30.20	100	236	1

Locked [ ]

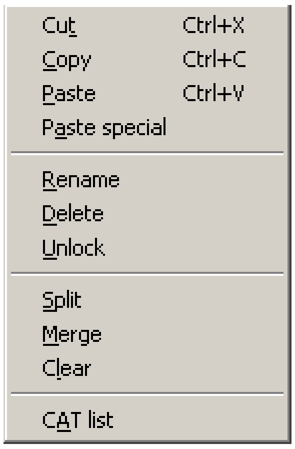
Figure 1:3. Databank Utility.

The columns Object, Modification date, OC1, OC2, Size and Locked are used for all types of objects and databanks while the columns Version, Access and Modified By are valid only when the Data Management solution is used.

Pressing the **Find** button a search operations starts as according to a search criteria specified through:

· **Data bank name**	Name of the data bank to be treated. Projects the name can be a full file-path to the indexed or sequential data bank as well as a Environment Variable appropriately set. Such a data bank is accessed either from a local file system or through a database server. The title text of a child window contains the name of the host acting as database server for the given data bank.
· **Object name**	Name given as specific object name or a name with wild cards. Accepted wild cards are asterisk '*' for any number of characters and percent sign '%' to replace exactly one character. Nothing given is the same as '*'.
· **Object code 1**	Objects with the specified object code 1 are treated.
· **Object code 2**	Objects with the specified object code 2 are treated.
· **Created between date**	Objects created after this date are treated. Current date is kept in scroll list as an example of the syntax
· **and date**	Objects created before this date are treated. Current date is kept in scroll list as an example of the syntax.
· **Locked**	Only locked objects are treated.

The following operations can be performed from the **Edit** menu or the **Pop up** menu:



· **Copy - Paste - Paste special**	Objects are copied between data banks. Paste special will keep the original storing date of the object(s).
· **Cut - Paste - Paste special**	Objects are moved between data banks. Paste special will keep the original storing date of the object(s).

Through the Options function in the Tools menu it is possible to decide whether the creation date should be kept or if current date should be set for the handled objects.

- **Delete** Objects are deleted from a data bank.
- **Rename** Objects in a data bank are renamed. It is possible to rename a range of objects through group name specification. A group name is given by zero or several characters followed by an asterisk '*'.
- **Unlock** Unlocking locked objects in a data bank. The function must be used with the greatest care and only when the locks known to be caused by an error situation.
- **Split** Create a sequential data bank of each object from an indexed data bank and place them in directory specified by user.

  Note: Please note that since the resulting file names are created from the object names, the split function can not be used for all indexed databanks. Excluded are:

`SBP_SKETCH_DB`	files named `SP:*` cannot be used
`SBF_DB_FSTD`	files named `SID::*` cannot be used

- **Merge** Retrieve objects from sequential data banks located in a directory specified by user and store the objects in an index data bank
- **Clear** Will clear the display of listed objects.
- **CAT list** AVEVA Marine information is stored in objects of different types. A general object type, at least for hull information, is the CAT ("Curve And Table") object. The internal structure and contents of such an object depends on the type of model object that it describes. The type of model object is identified by an object code (object code 1) that is registered centrally by the system. The internal structure of a CAT object is built by Contours, Attributes and Segments while the data is stored as Integers, Reals and Text strings. By this function it is possible in a dialog to expose the contents of a CAT object.

1.4 Command-line Utilities

Although the Control Panel applets are the preferred tools to use, selected functionality is also available through command line utility programs.

1.4.1 Data Bank Utilities

A family of tools have been developed for backup, recovery and other kinds of maintenance of indexed data banks. These tools are described below. Some of the tools can also be used for sequential data banks (e.g. copying and listing of objects).

These tools are to be considered as Administrator tools only and should not be used by the normal user.

Note: It is recommended to avoid data bank accessed by other applications during data bank backup/reorganization.

- **Programs**

The data bank utility programs for indexed data banks are described below.

SA004 - Data Bank Utility

The program is a utility program for data banks (indexed or sequential). Input to the program is given interactively upon request from the program.

The following operations can be performed:

Copy	Objects are copied between data banks.
The copied objects can be given new names different from the names of the original objects.	
Move	Objects are moved between data banks.
The moved objects can be given new names different from the names of the original objects.	
Note: Please note that it is not possible to move objects from a sequential data bank.	
Delete	Objects are deleted from an indexed data bank.
List	Objects in a data bank are listed.
Archive	Objects are moved between different data banks as by the operation move, but the creation date of the object is kept.
If the data bank is stored in another directory than the one from which the program is run, the directory must be specified separately. Objects can be specified according to the following conditions, which can be combined arbitrarily.	
Object name	You can give a specific object name, a group name or a name with 'wildcards'.

In a name with 'wildcards', an asterisk can replace any number of characters and a per cent sign can replace exactly one character.

Example:

```
TB-272-*-1
*272*
TB%72*
```

A group name is given by zero or several characters followed by an asterisk.

Example:

```
TB272*
TB*
```

When a specific object name (without wild cards) or a group name is given, it is possible to rename the object(s) by giving a new name in the same way as the original name.

Example:

```
Object name TB272*
New object name
TB114*
```

· Object code 1	Objects with the specified object code 1 are treated.
· Object code 2	Objects with the specified object code 2 are treated.
· After date	Objects created after this date are treated.
· Before date	Objects created before this date are treated.

Error messages are given on the standard output device, normally the terminal from where the program was initiated. The messages are self-explanatory.

A list of the performed operations is produced in a file named sa004.lst.

SP304 - Object Utility

The program sp304 makes it possible to do manipulations on objects, like listing and unlocking them. It is a console application, interacting with the user by asking questions and prompting for user inputs. The program is designed to be used for error debugging and for recovering from error situations.

Data Bank Name

When the program starts, the user is asked to key in the name of the data bank to fetch objects and information from. Logical names can be used.

Cmd

When the prompt 'Cmd: ' is shown, the available commands can be listed by typing HELP. The different commands are described below. All commands can be abbreviated down to a level where they still are unique.

HELP

The HELP command lists the available commands on the screen, with a short description of the commands.

EXIT

The EXIT command will make the program exit.

OUTPUT

Listing of objects are by default done on the terminal screen, but the user can list in a list file by using the OUTPUT command. The name of the file is prompted for. Not specifying the output file implies listing on the screen.

LIST

The LIST command dumps the data of the object in the current list file (see OUTPUT above). The listing format corresponds to the element definitions documented in the system description, and is not further explained here. The listings are mainly intended to be studied by system administrators when error debugging, testing, etc. The program keeps on prompting for object names until no name is entered.

BRIEF

The BRIEF command dumps the data of the object in the same way as the command LIST, without the data part.

STATUS

The STATUS command can be used when checking if objects are locked in the data bank and for unlocking locked objects. The command must be used with the greatest care and only when the locks are known to be caused by an error situation. The program keeps on prompting for object names until no name is entered. Wild cards *,% (asterisk, percent) where * means 0-n of any character and % means exactly 1 of any character. Objects that are not locked are listed on the screen, locked objects makes the program ask the user if it is OK to unlock them.

UNLOCK

See STATUS above. The only difference is that only locked objects will be listed.

The following paragraph shows a program session where all the available commands of sp304 are used.

```
Data bank name: SB_PSDB
Cmd: HELP
-- Commands -------------------------------------------------
-- EXIT    exit program --
-- HELP    display this text --
-- BRIEF   list object, without data part --
-- LIST    list object, including data part --
-- OUTPUT  redefine list file of object --
-- STATUS  display object status and optionally unlock --
-------------------------------------------------------------
Cmd: OUTPUT
Filename: dump.log
%SP304-I-LISTFILE, the new list file is dump.log.
Cmd: LIST
Objectname: SB=230=M101
%SP304-S-LIST, SB=230=M101 listed.
Objectname:
Cmd: STATUS
Objectname: SB=230=M101
%SP304-I-NLCK, SB=230=M101 is not locked.
Objectname: SB=*=CS*
```

```
%SP304-I-NLCK, SB=330=CS1 is not locked.
%SP304-I-NLCK, SB=330=CS17 is not locked.
%SP304-I-NLCK, SB==CS19 is not locked.
%SP304-I-EXTLCK, SB==CS9 is locked by an other user.
Forced unlock (Y/N): Y
%SP304-I-NLCK, SB==CS9 is not locked.
Objectname:
Cmd: EXIT
```