

SWBus Troubleshooting

or

What Do I Do If I Have Problems Running The SWBus?

Table of Contents

1	Installation Problems	2
1.1	Installation on the Windows platform.....	2
1.2	Installation on Unix platforms	2
2	Initial Start-up Problems	2
2.1	Problem: Control won't start.....	2
2.1.1	Identifying why control won't start	3
2.1.2	Another control is already running.....	3
2.1.3	Missing entries in the services database	4
2.1.4	General error message	4
2.1.5	Problems with backup file.....	5
2.2	Problem: Processes won't connect to control.....	5
2.2.1	Using controlutil.....	5
2.2.2	How does the SWBus know where control is located?	6
2.2.3	How is SWBus-based processes identified?	7
2.2.4	How do SWBus-based processes connect?	7
2.3	Problem: Process won't connect to each other	7
2.3.1	Verifying that both processes are connected to control	7
2.3.2	Test case : SWBus example programs	8
2.3.3	An example: SbCCRemote and ~notify	8

1 Installation Problems

The SWBus is delivered on a CD and it can also be downloaded from the Internet, www.ife.no/swbus. The expected results of installing the SWBus on a computer are:

- A SWBus home directory containing the subdirectories and files described in section 3.1 in the SWBus User's Guide.
- The services database updated according to the description in section 3.2 in the User's Guide
- Environment variables set according to the description in section 3.2 in the User's Guide.

1.1 Installation on the Windows platform

On the Windows platform the SWBus is delivered with an installation program starting automatically when the CD is inserted in the PC. If the SWBus is downloaded from the Internet, start the installation program by double-clicking the downloaded executable.

The installation program will create the files and directories, update the services database and set the mandatory environment variables. If the installation program does not work properly, please ask for support at softbus@hrp.no.

1.2 Installation on Unix platforms

On Unix platforms, the SWBus is delivered as a set of tar files. Unpack the tar files according to the instructions on the CD or the SWBus download page on the Internet.

When the files have been unpacked, the environment variables must be set manually according to the instructions in the User's Guide.

The services database must also be updated manually according to the instructions in the User's Guide. Please note that this operation often requires super-user privileges. Contact your local system administrator for help.

2 Initial Start-up Problems

This section describes how to solve problems appearing when trying to run simple SWBus based applications. It describes problems related to installation and configuration, and after completing this section you should be able to successfully compile and run the example programs provided with the SWBus.

If you can't make the examples compile and run, please follow the instruction in this section to identify and solve your problem.

2.1 Problem: Control won't start

Control is a small program providing the mapping from process names to host and port information. It is required to run on some computer in the network. All SWBus-based programs contact control when invoking the function SbInit, and register with their name, host and port.

On the Windows platform, control is implemented as a service. To check if it is running, open your start-menu and select *Settings – Control Panel – Administrative Tools – Services*. This will open the standard Windows listing of all services on your computer. Scroll the list and search for “*SoftwareBus Control*”. The service’s status should be “*Started*”, otherwise control is not running. If it is running, move on to section 2.2. If not, follow the instructions in the sub-sections below.

On Unix platforms, control is implemented as an ordinary program. It should normally be started as a background process and be left running “forever”. If it for some reason fails to start properly, please follow the instructions below.

2.1.1 Identifying why control won’t start

Follow the steps below to figure out why control won’t start.

1. Open a terminal window (MS-Dos shell on Windows platform)
2. Go to the directory where control is located by typing the following command:
Unix platforms: `cd $BUSPATH/bin/$ARCH/control`
Windows: `cd %BUSPATH%\bin\%ARCH%`
3. If the typing in step 2 is not recognised as a valid command, please check your environment variables, ref. section 3.2 in the SWBus User’s Guide. On Windows, the command may fail if you have installed the SWBus in a path containing spaces. If so, check your environment variables manually and then go to the intended directory.
4. Type the following command to start control:
`./control`
5. Control should respond with the following output and then run silently:
`Control (SWBus Release 3.3 March 2002)`
Note that control is not supposed to provide any more output. Please refer to section 2.2.1 to learn how to get output from control.
6. If control starts, but immediately responds with an error message, the following sub-sections provide hints and solutions to solve the problem depending on the error message.

2.1.2 Another control is already running

Only one instance of control can run on a host simultaneously. If the error message from control says that another control is already running, control has detected that the port `softbus_name` is used by another process. This other process is probably another instance of control running on the computer. To find its process id:

- Unix platform: Use the command “`ps -ef`” in your terminal window and search for control.
- Windows: Start the Task Manager by right-clicking on the taskbar and select the “Processes” tab.

If the running control is not used by anybody, it can be killed. However, care should be taken to verify whether somebody else is using the existing process. See section 2.2.1 for instructions on how to check which processes are connected to control.

2.1.3 Missing entries in the services database

As described in section 3.2 in the User's Guide, the SWBus requires three entries in the services database. One of these entries, `softbus_name`, is used by control. If control was unable to retrieve this entry, it issues an error message and fails to start.

The function used to retrieve the entry is the `getservbyname()` function. Please refer to its manual page for details. The program `$BUSPATH/bin/$ARCH/testComputerConfig` can be used to test your computer's configuration.

2.1.4 General error message

If control for some other reason fails to initialise, it issues an error message saying "Error opening server port '`softbus_name`'. (Error = `<SWBusError>`, `<SystemError>`)". The number `SWBusError` refers to one of the errors listed below.

Name	Value	Description
<code>ERR_C_GLH</code>	11	Unknown local hostname
<code>ERR_C_GLS</code>	12	Unknown local servicename
<code>ERR_C_GRH</code>	13	Unknown remote hostname
<code>ERR_C_GRS</code>	14	Unknown remote servicename
<code>ERR_C_CSO</code>	15	socket call failed
<code>ERR_C_BSO</code>	16	bind call failed
<code>ERR_C_LSO</code>	17	listen call failed
<code>ERR_C_CNS</code>	18	connect call failed
<code>ERR_C_SMH</code>	19	send message head failed
<code>ERR_C_SMT</code>	20	send message tail failed
<code>ERR_C_SML</code>	21	send message length wrong
<code>ERR_C_SMR</code>	22	send message receiver unknown
<code>ERR_C_RMH</code>	23	receive message head failed
<code>ERR_C_RMT</code>	24	receive message tail failed
<code>ERR_C_ANC</code>	25	accept call failed
<code>ERR_C_COC</code>	26	close old connection failed
<code>ERR_C_SNM</code>	28	Set Non-blocking mode failed
<code>ERR_C_RNM</code>	29	Reset Non-blocking mode failed
<code>ERR_C_SEL</code>	30	select call failed
<code>ERR_C_TLM</code>	31	Too long message
<code>ERR_C_NAM</code>	32	alloc call failed
<code>ERR_C_SAM</code>	33	Set Async Mode failed
<code>ERR_C_SPG</code>	33	Set Process Group failed
<code>ERR_C_OVF</code>	34	Connection overflow

ERR_C_FCT	35	fcntl call failed
ERR_C_SMF	36	fifo full (nonblocking mode)
ERR_C_SID	37	failed to retrieve socket id
ERR_C_SSO	38	setsockopt call failed

On Unix, the value `SystemError` in the error message equals the value of the global system variable `errno` set by the system after a failing system call. On Windows, it equals the value returned from the `WSAGetLastError()` function. The value of `SystemError` may identify the reason for the failing system command listed in the table above.

The most commonly reported error code has been the `ERR_C_GLH`, number 11. This error is caused by an incorrect computer set-up. To test your computer set-up, use the program `$BUSPATH/bin/$ARCH/testComputerConfig`.

2.1.5 Problems with backup file

When control starts, it looks for a file containing information on processes and their hosts and ports from the previous run of control. This way, a system can continue to operate even if control for some reason is terminated and restarted. At start-up, control verifies that it can read and write this file, and also that a temporary backup file created prior to each writing is non-existent. If such a temporary backup file exists, control issues an error message and exits. The error message says: "A backup file (.bk) exists for <fileName>. Clean up manually before restarting control".

If you are sure that there is no currently running system relying on control, then the backup file could simply be removed and control can be restarted. However, if a system is running, the contents of the backup file and the ordinary file should be compared with the current state of the running system to ensure that the ordinary file contains the correct information before restarting control.

The most probable reason why a backup file exists, is that control has been killed manually right after the killing of a connected process. To avoid such problems, our general advise is to let control run forever like a daemon.

2.2 Problem: Processes won't connect to control

Making control run properly is the first mandatory step towards establishing a SWBus connection between two processes. If you are able to run control as described in section 2.1, you are ready to follow the instructions in this section.

2.2.1 Using controlutil

`Controlutil` is a utility program for control which can be used to check the current status of control, and to send simple commands. Please refer to its manual page for all details about its use.

If your processes won't connect, the first step towards finding the solution is to check if `controlutil` is able to connect to control. The steps below describes the procedure:

1. Verify that you are able to run control both on the intended host of control and on the host of your not-connecting process (if different). Ref. section 2.1 for details.
2. Kill control on the host of your not-connecting process and make sure control is running on the intended host of control.
3. Open a terminal window (Windows: MS-Dos shell) on the host of your not-connecting process and go to the \$BUSPATH/bin/\$ARCH directory.
4. Type the command


```
controlutil -h <hostOfControl> -q
```
5. Controlutil will then normally respond with the following message


```
controlutil (SWBus Release 3.3 March 2002 )
Process information from control (3.3) at <hostOfControl>:
1: <procesName> host:<host> port:<port> pid:<pid> version:<version>
2: .....
There are N processname(s) registered, M of these are connected
```

Note that the actual number of processes in the list above might be zero.

If controlutil responds as indicated above, connection to control is established and information is retrieved successfully. Continue with section 2.2.2.

If controlutil does not respond as indicated above, it issues an error message describing the problem in plain text, and it also provides some detailed error reporting to enable further investigations if the plain text description is not enough. The detailed error reporting includes error codes (Error=X,Y) as described in section 2.1.4.

2.2.2 How does the SWBus know where control is located?

In order for a SWBus-based process to start properly it must locate and connect to control. The previous section showed how to use the utility program controlutil to test whether it was possible to establish such a connection. In that test, we explicitly stated the host on which control was running, using the -h option of controlutil. But what about other SWBus-based processes? How do they know where control is located?

When a SWBus-based process is started, it invokes the SWBus API function SbInit(), please refer to its manual page for details. The function takes three arguments, the second argument specifies the host where control is supposed to be running.

If a value is passed for the second argument, this value will be used as the host name. If NULL or the empty string is passed, the contents of the environment variable CONTROLHOST will be used as the host name. If CONTROLHOST is unset, the name of the local host will be used for the host name.

When the host name has been extracted according to the rules described above, the process will try to connect to control at the specified host, just like controlutil. However, on Unix platforms, if no control is found and the specified host equals the local host, the process will try to start control automatically and then try to connect once more.

Practice has shown that using the environment variable CONTROLHOST is the most convenient way of specifying where to find control. The exercise in section

2.2.1 can be repeated avoiding the use of the `-h` option and relying on the environment variable `CONTROLHOST` instead. Note that the output from `control` includes the name of the host on which `control` is running.

2.2.3 How is SWBus-based processes identified?

As explained in the previous section, a SWBus-based process invokes the function `SbInit()` when starting. The first argument to `SbInit` is the name of the process, and this name must be unique within the scope of `control`. In other words, two processes can not register with the same `control` if they have identical names. If they try, `control` will refuse to register the last process, and `SbInit()` will fail.

2.2.4 How do SWBus-based processes connect?

The SWBus API function `SbOpen()` is used to initiate a connection between two SWBus-based processes. Please refer to its manual page for details. This section explains what is going on behind the scene when one process (the client) tries to connect to another process (the server).

When a process invokes `SbInit()` it initialises the internal SWBus data structures and registers itself with `control`. The information sent to `control` includes the process name, the host name and the port. `Control` stores this information in its internal data structures, and people can view the contents of these data structures by the use of `controlutil` as described in section 2.2.1.

When a client initiates a connection to a server, the name of the server is used for identification. The client first sends a request to `control` asking for the host and port registered with the specified process name. `Control` replies with the requested information if present, and the client uses this information to establish a direct connection to the server. If the requested information is not present in `control`, the client will try again later, at intervals specified by the environment variable `BUSRETRY`.

2.3 Problem: Process won't connect to each other

Even if both processes can connect to `control`, they still may fail to connect to each other. This section describes possible causes.

2.3.1 Verifying that both processes are connected to control

The first thing to check when discovering that two processes won't connect to each other is whether they are both connected to the same `control`. `Controlutil` is well suited for this. The procedure is as follows:

1. Start `controlutil` with the `-q` option (and optionally the `-h` option), ref. section 2.2.1.
2. Verify from the output that both processes are registered and connected simultaneously.
3. Verify from the output that the processes have compatible SWBus versions. SWBus compatibility information is provided in the `releaseNotes` for each SWBus version.

If the two processes are not registered with the same `control`, they can never connect. If both processes seem to have started properly, but they never connect, the most likely explanation is that they are connected to two different `control`s.

2.3.2 Test case : SWBus example programs

Having obtained the information presented in this document, it is now time to verify if two SWBus-based processes are able to connect in your environment. The basic SWBus example, described in chapter 4 of the SWBus User's Guide, is a good candidate for this testing. Source code for this example can be found in the \$BUSPATH/examples directory. Compile and run it according to the instructions in the README file.

If the example programs don't connect and run as expected, use the information provided in this document to find the cause of the problem. If you are still unable to solve the problem, please contact softbus@hrp.no for support. Please state your HW and SW platform, describe how far into the procedure described in this document you have been able to go and attach as much information as possible from the error messages.

If the example programs do connect and run as expected it is time to test various configurations. For instance having control, the server and the client running on different computers in your network and using CONTROLHOST to identify where control is running.

2.3.3 An example: SbCCRemote and ~notify

The SWBus class SbCCRemote has a method called ~notify that is invoked whenever the state of the connection towards another SWBus process has changed. Please have a look at the manual page of SbCCRemote for details.

The example provided in \$BUSPATH/examples/connect.c shows how the ~notify method should be implemented.