Dixie Gem/Hsms/Secs Simulator
User Guide

Jazz Soft, Inc.

1. Revision History

Version	Date	Name	Description
1.00	Aug, 17th, 2009	Hikaru Okada	Created as new document.
1.01	Jul, 13th, 2014	Carl Hikaru Okada	Switched compiler from Visual Studio 2008 (C++) to Visual Studio 2010. Windows 2000 is no longer supported.

2. Table of Contents

1. Revision History 2
2. Table of Contents 3
3. What is Dixie? 5
1.1 System Requirement 5
2 Setup 6
2.1 Install Jazz Soft Semiconductor Solution 6
2.2 Install HASP Driver 9
2.3 Uninstall 14
4. Operation Summary 16
2.4 How to Launch Dixie 16
2.5 Ribbon 18
2.5.1 Home Category 18
2.5.2 Others Category 19
2.6 Communication View 20
2.7 Message View 21
2.8 Message List 22
2.9 Property WIndow 23
2.10 Output Window 24
2.11 Status Bar 25
5. Screen Detauls 26
2.12 Home – View Panel 26
2.12.1 Refresh 26
2.12.2 List Style 27
2.12.3 Font 28
2.13 Home – Communication Panel 30
2.13.1 Connect 30
2.13.2 Settings 31
2.13.3 Script 35
2.13.4 Compile 36
2.14 Home – Message Panel 37
2.14.1 Send 37
2.14.2 Edit 38
2.14.3 Add 39
2.14.4 Delete 40
2.15 Others – Clipboard Panel 41
2.15.1 Paste 41
2.15.2 Cut 41
2.15.3 Copy 41
2.15.4 Delete 41
2.16 Others – View Panel 42
2.16.1 Status Bar 42
2.16.2 Properties 42
2.16.3 Output 42
2.16.4 Message View 42
2.16.5 Communication View 42
6. Tutorial 43
2.17 Initialization Script 43
2.17.1 Specification of Initialization Scenario 43
2.17.2 State Transition 43
2.17.3 Selected Event Procedure 43
2.17.4 Received Event Procedure 44
2.17.5 Receiving Message with Same Stream and Function 44
2.17.6 Entire Source Code 45
2.17.7 Add New Script 47
2.17.8 Test Script 49
7. Script Language Specification 50
2.18 Numbers 50
2.19 Strings 50
2.20 Comments 50
2.20.1 // 50
2.20.2 /* */ 50
2.21 Event Handler Functions 51
2.21.1 OnSelected() 51
2.21.2 OnReceived() 51
2.21.3 OnProblem() 52

2.22 Predefined Functions 53
2.22.1 Send() 53
2.22.2 Reply() 53
2.23 Variable Types 54
2.24 Predefined Objects 54
2.24.1 in 54
2.24.2 out 54
2.24.3 arg 55
2.24.4 cout 55
2.25 User Defined Variables 55
2.25.1 Int type 55
2.25.2 string type 56
2.26 Statements 56
2.26.1 if 56
2.26.2 return 58
2.26.3 Assignment 58
2.26.4 Others 58
2.27 Operators 58

3. What is Dixie?

Dixie is a communication simulator software for GEM, HSMS and SECS.

1.1 System Requirement

- Windows XP, Windows Vista, Windows 7, Windows 8 or Windows 8.1

2 Setup

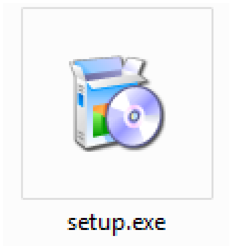
2.1 Install Jazz Soft Semiconductor Solution

Please login Windows with Administrator privilege. If User Account Control (UAC) was enabled in Windows Vista, Windows 7, Windows 8 or Windows 8.1, installation might fail. Please temporarily turn off the UAC.

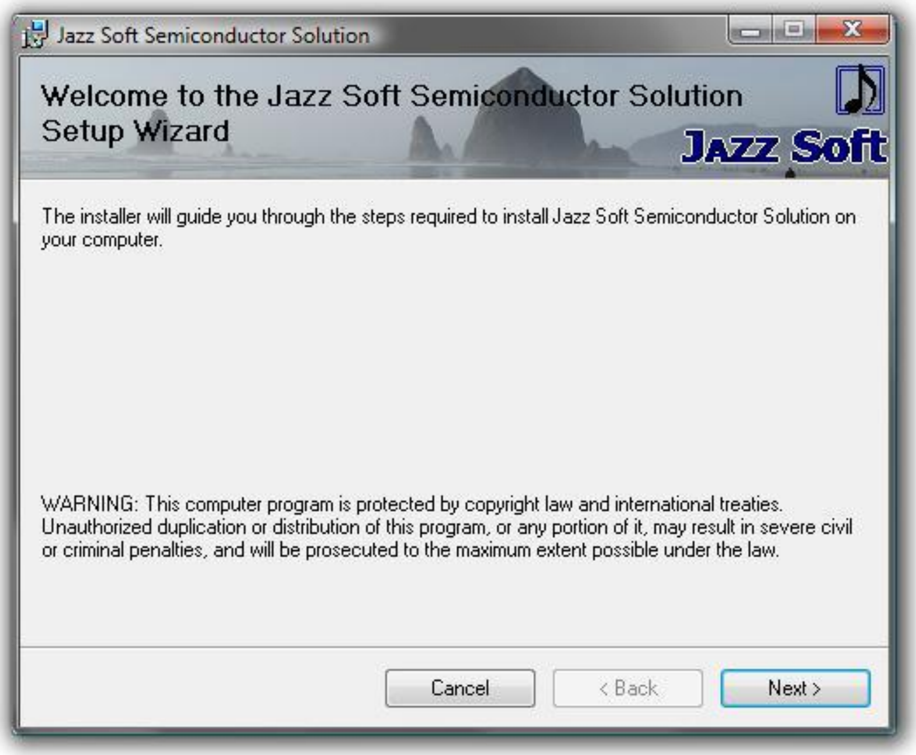
Run setup.exe to install.

1 If previous version of Jazz Soft Semiconductor Solution was installed on the computer, please uninstall it first. Uninstallation may be available in “Programs and Features” from Control Panel.

2 Launch setup.exe. If Microsoft Visual C++ 2008 SP1 redistributable files were not installed, they would be installed first.



3 The setup wizard for Jazz Soft Semiconductor Solution will appear on the screen. Click “Next” button.

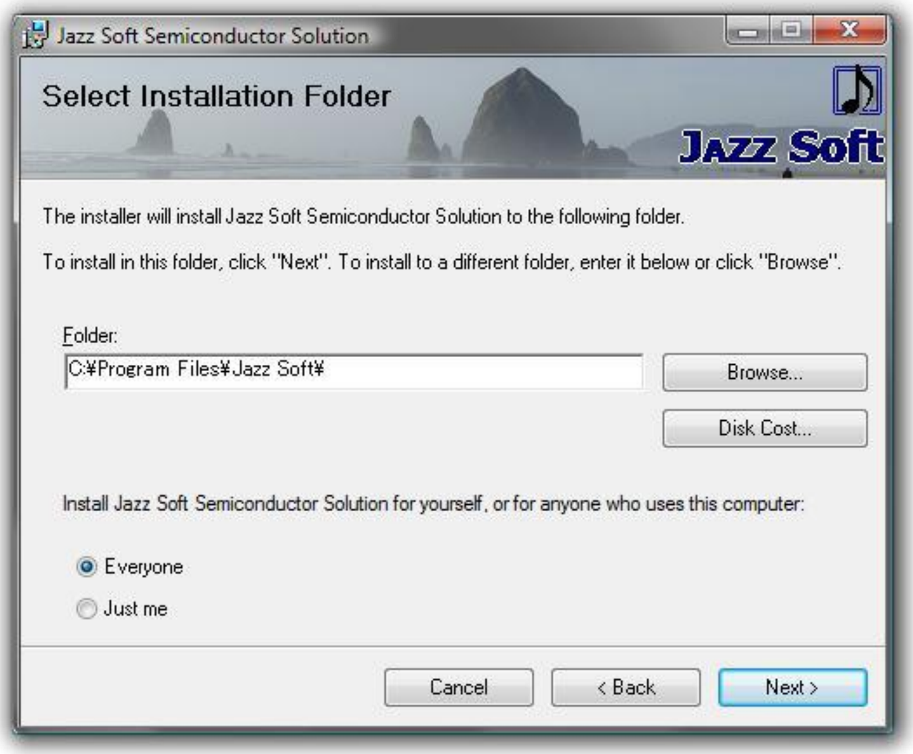


Welcome to the Jazz Soft Semiconductor Solution Setup Wizard

The installer will guide you through the steps required to install Jazz Soft Semiconductor Solution on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

4 Select installation folder and user account.　Usually, don’t change anything and click “Next” button.



Jazz Soft Semiconductor Solution

Select Installation Folder

The installer will install Jazz Soft Semiconductor Solution to the following folder.

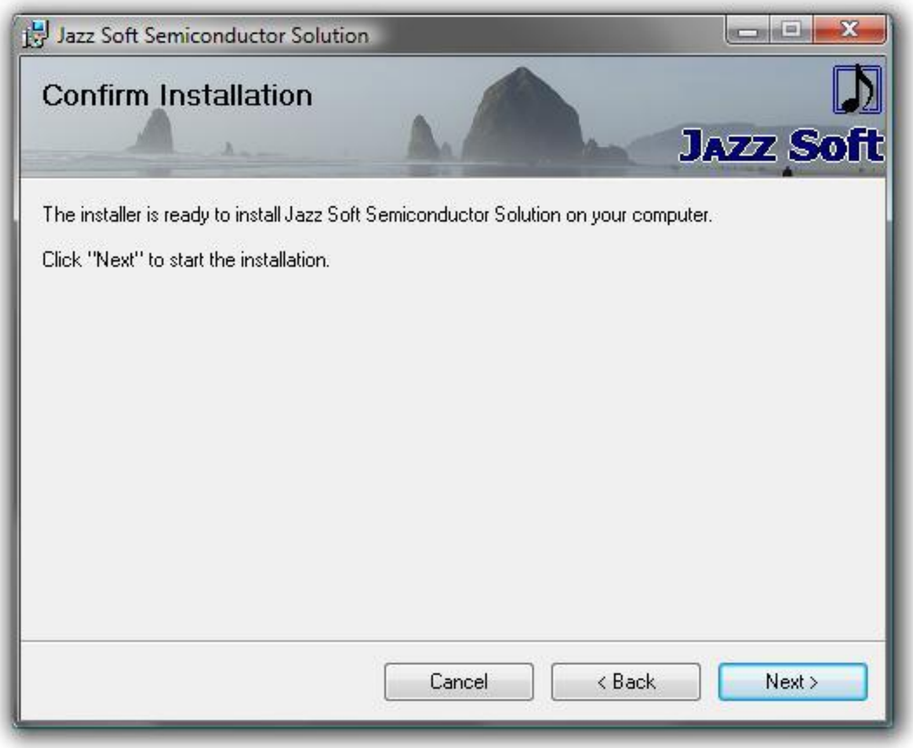
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder:
C:¥Program Files¥Jazz Soft¥

Install Jazz Soft Semiconductor Solution for yourself, or for anyone who uses this computer:

● Everyone
○ Just me

5 Confirmation screen will appear.　Click “Next” button.



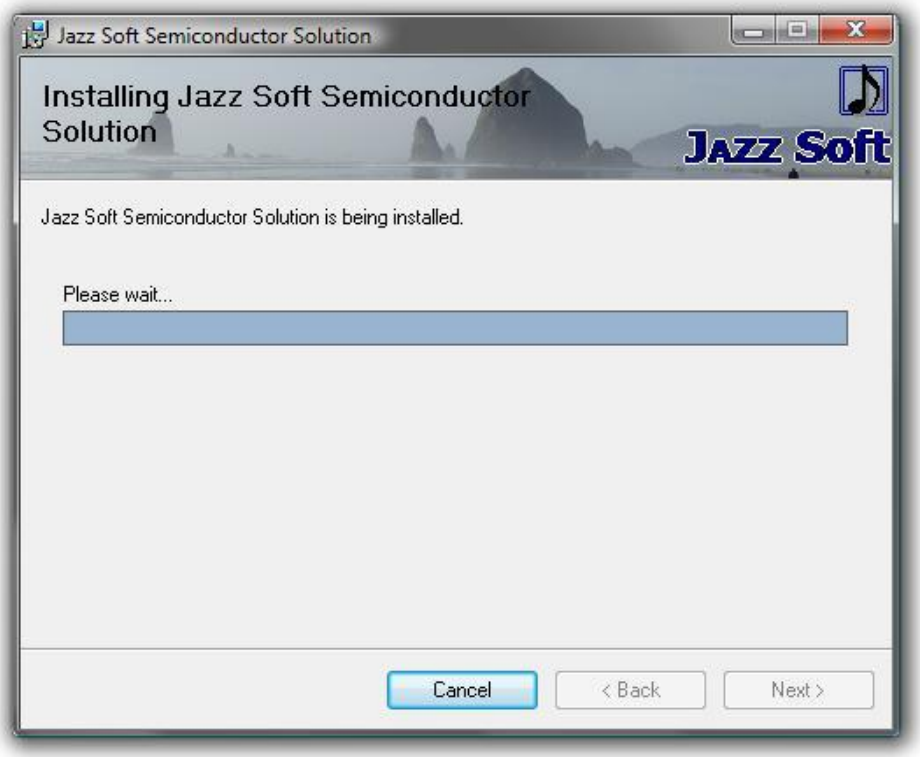
Jazz Soft Semiconductor Solution

Confirm Installation

The installer is ready to install Jazz Soft Semiconductor Solution on your computer.

Click "Next" to start the installation.

6 Actuall installation will start.

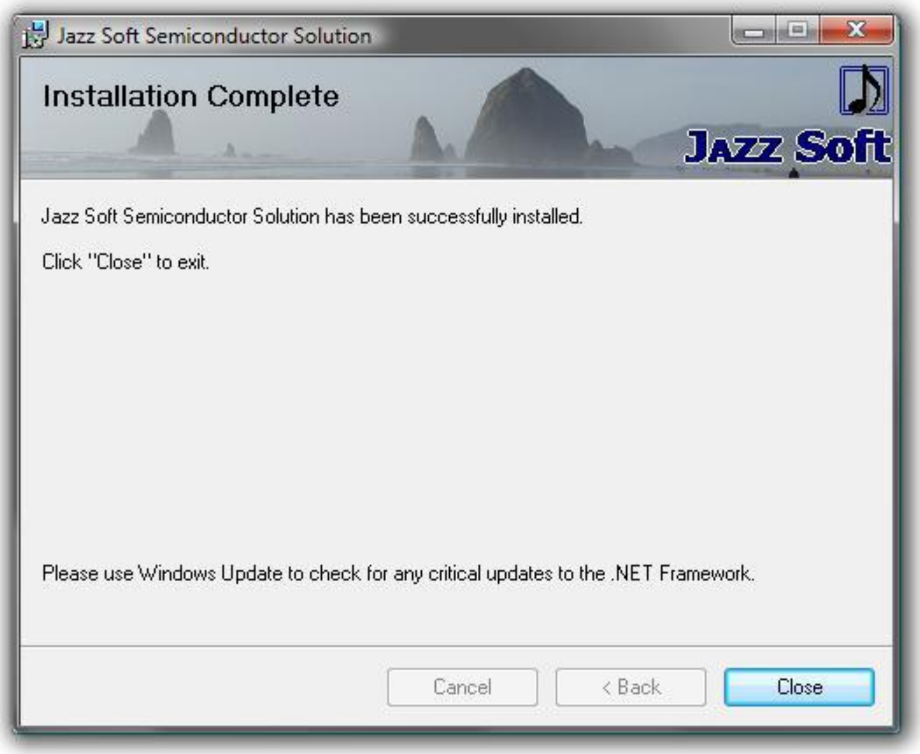


Installing Jazz Soft Semiconductor Solution

Jazz Soft Semiconductor Solution is being installed.

Please wait...

7 Once installation was completed successfully, following screen will appear. Click “Close” button.



Installation Complete

Jazz Soft Semiconductor Solution has been successfully installed.

Click "Close" to exit.

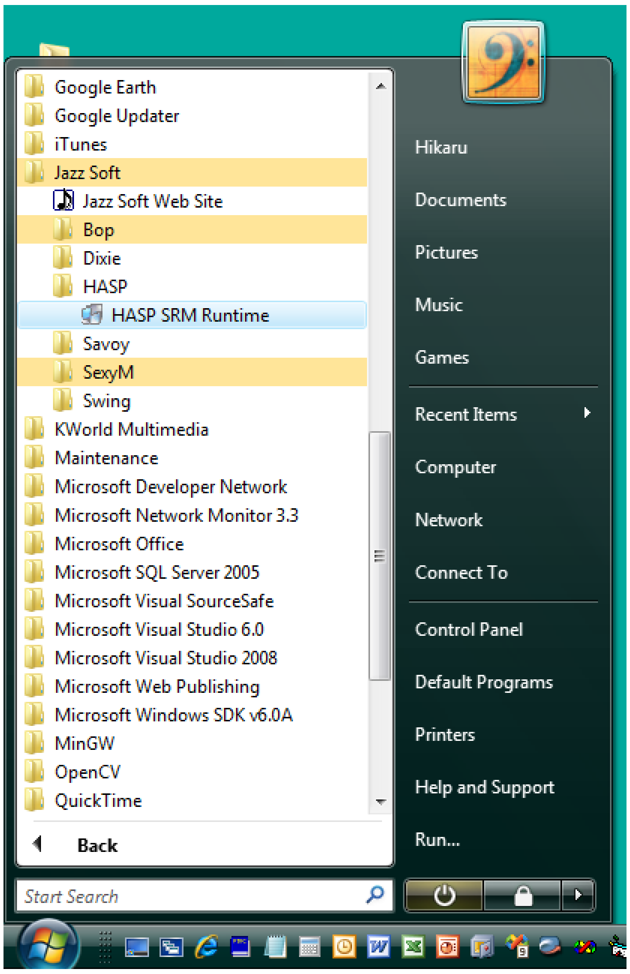
Please use Windows Update to check for any critical updates to the .NET Framework.

2.2Install HASP Driver

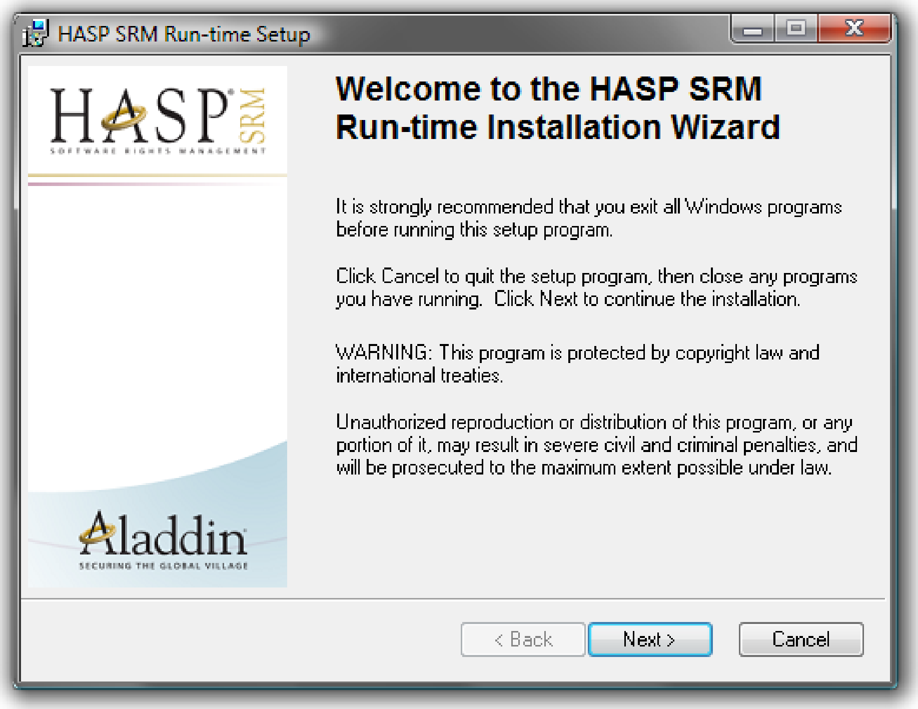
HASP key driver is not necessary for evaluation version, but is necessary to run as retail version. Required software to install HASP driver was copied by installer of Jazz Soft Semiconductor Solution

Please don't insert HASP key when installing HASP driver.

Click "HASP SRM Runtime" icon in [Jazz Soft] – [HASP] from Start menu.



9 Welcome screen will appear. Click “Next” button.



HASP SRM Run-time Setup

Welcome to the HASP SRM Run-time Installation Wizard

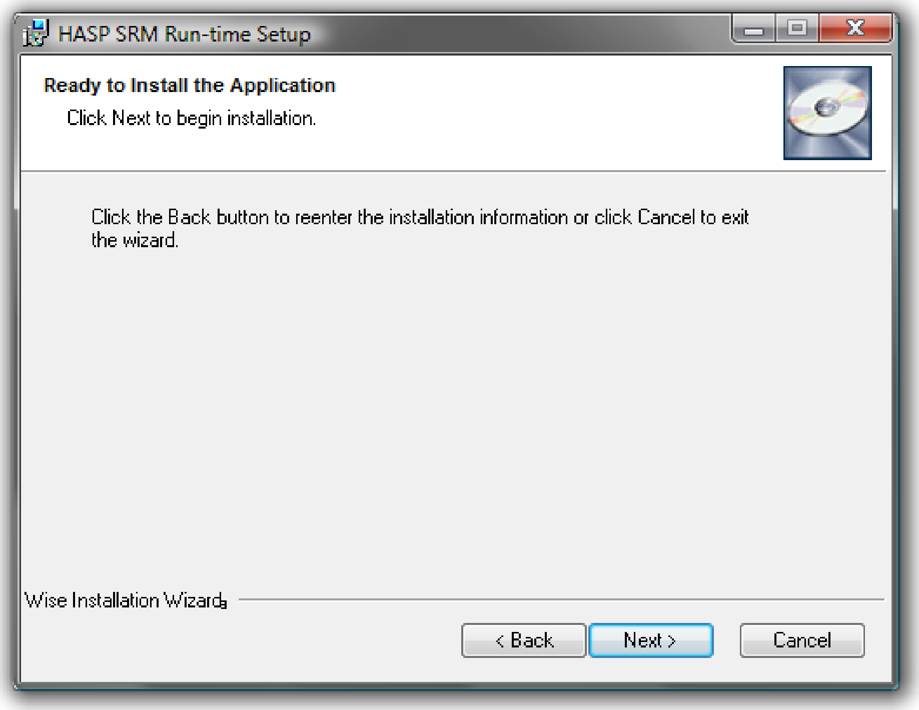
It is strongly recommended that you exit all Windows programs before running this setup program.

Click Cancel to quit the setup program, then close any programs you have running. Click Next to continue the installation.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

10 Confirmation screen will appear. Click “Install” button.



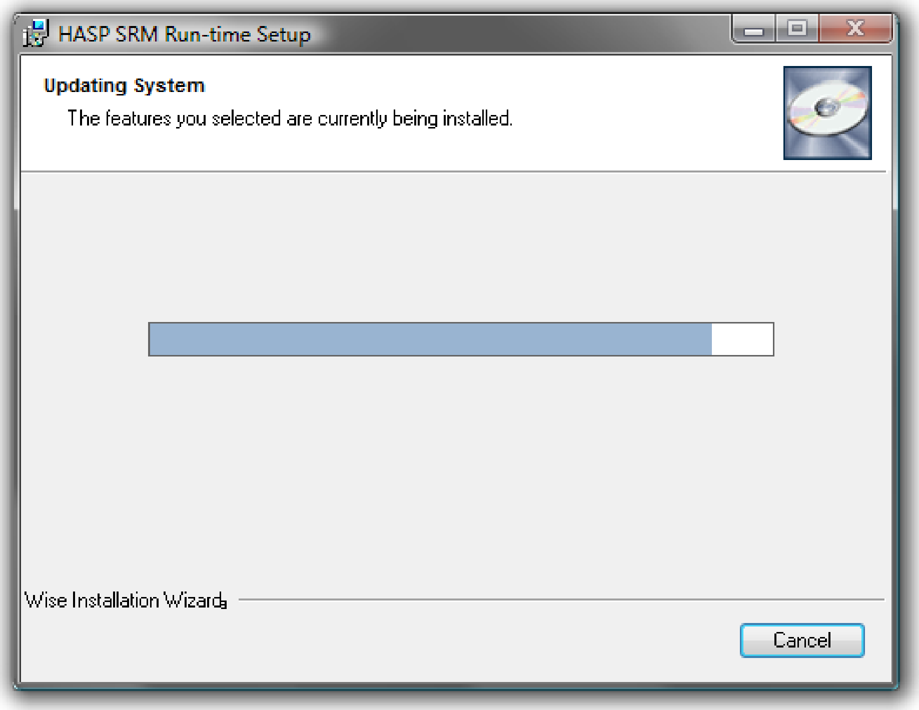
HASP SRM Run-time Setup

Ready to Install the Application

Click Next to begin installation.

Click the Back button to reenter the installation information or click Cancel to exit the wizard.

11 Actual installation will start.



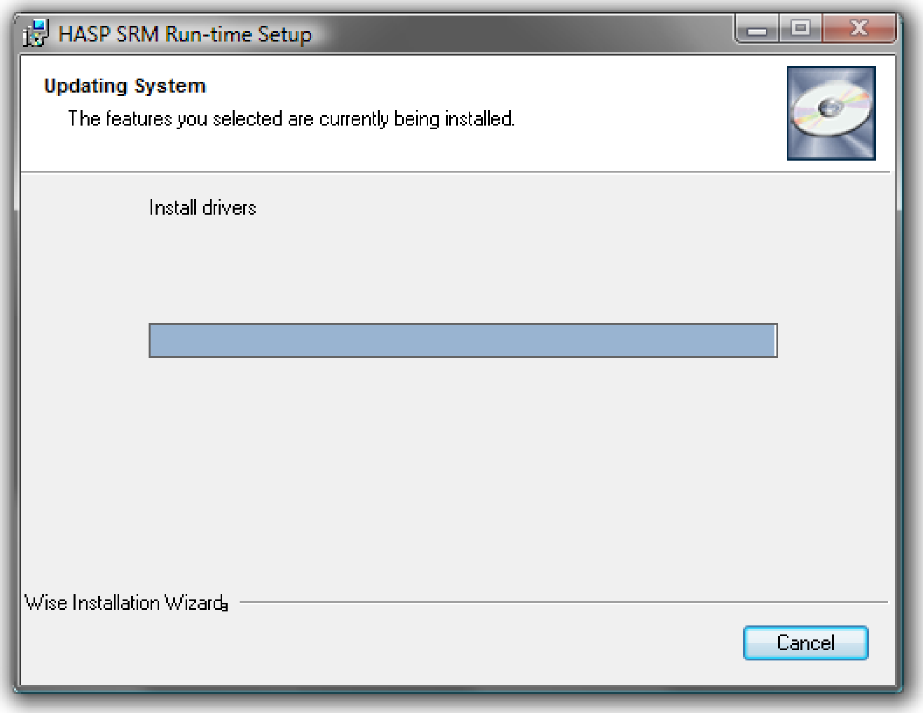
HASP SRM Run-time Setup

Updating System

The features you selected are currently being installed.

Wise Installation Wizard®

12 Text on the screen will change to “Install drivers”.   It may take several minutes.



HASP SRM Run-time Setup

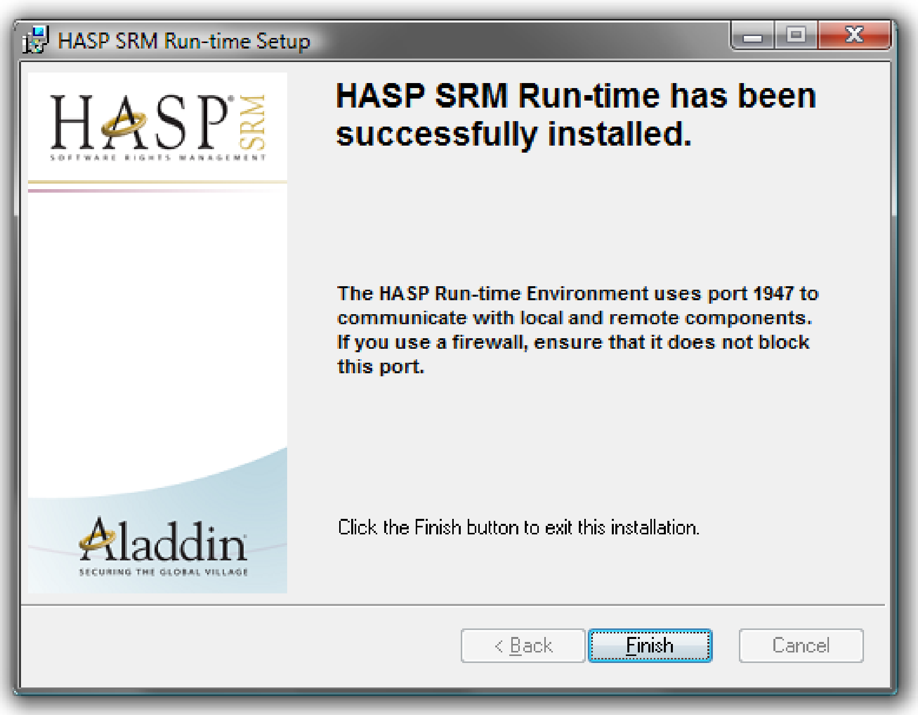
Updating System

The features you selected are currently being installed.

Install drivers

Wise Installation Wizard®

13 Once driver files are copied into computer successfully, following screen will appear. Click “Finish” button.

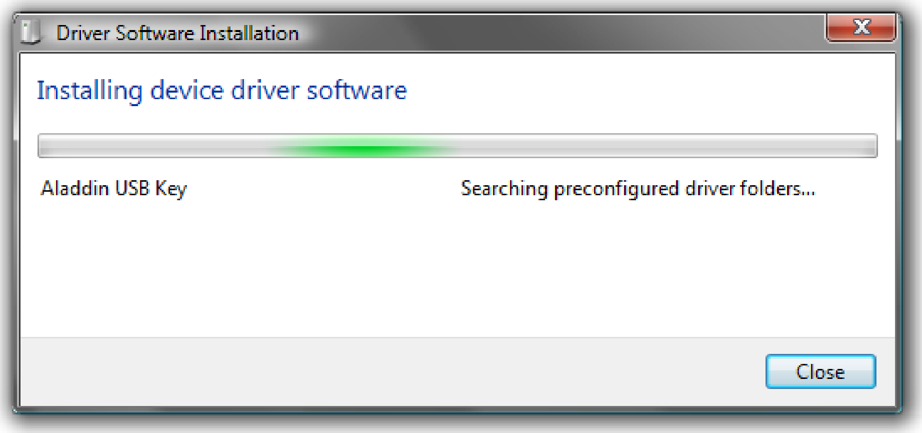

HASP SRM Run-time Setup

HASP SRM Run-time has been successfully installed.

The HASP Run-time Environment uses port 1947 to communicate with local and remote components. If you use a firewall, ensure that it does not block this port.

Click the Finish button to exit this installation.

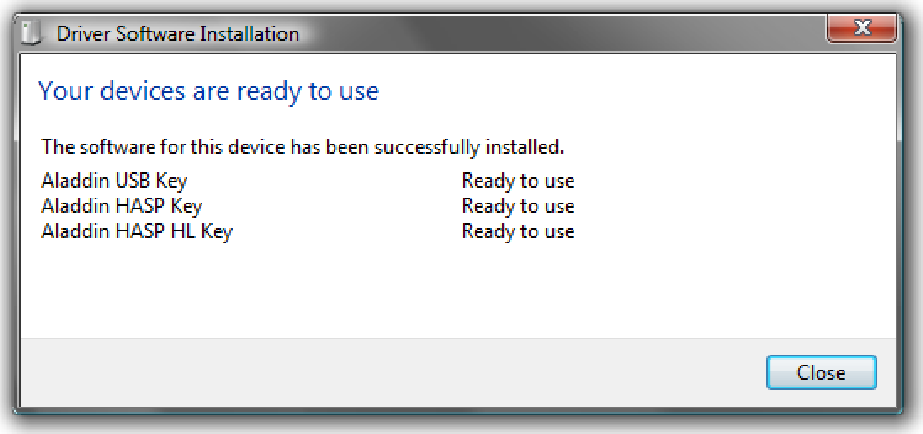
14 Insert HASP key in computer. Following popup dialog box will appear at lower right corner of screen on Windows Vista.



Installing device driver software

Aladdin USB Key　　Searching preconfigured driver folders...

15 Once installation was completed successfully, following screen will appear. Click “Close” button.



Your devices are ready to use

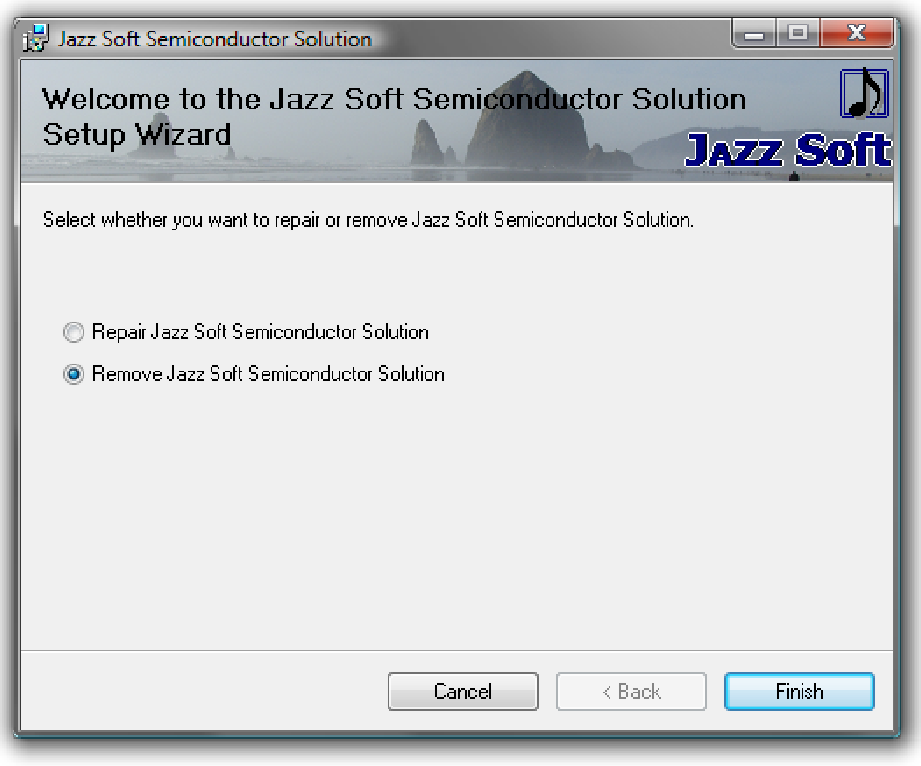
The software for this device has been successfully installed.

Aladdin USB Key	Ready to use
Aladdin HASP Key	Ready to use
Aladdin HASP HL Key	Ready to use

2.3Uninstall

Uninstallation may be available in "Programs and Features" from Control Panel. Or rerun setup.exe again.

1 Run setup.exe (the one you installed). Choose "Remove Jazz Soft Semiconductor Solution" and click "Finish" button.



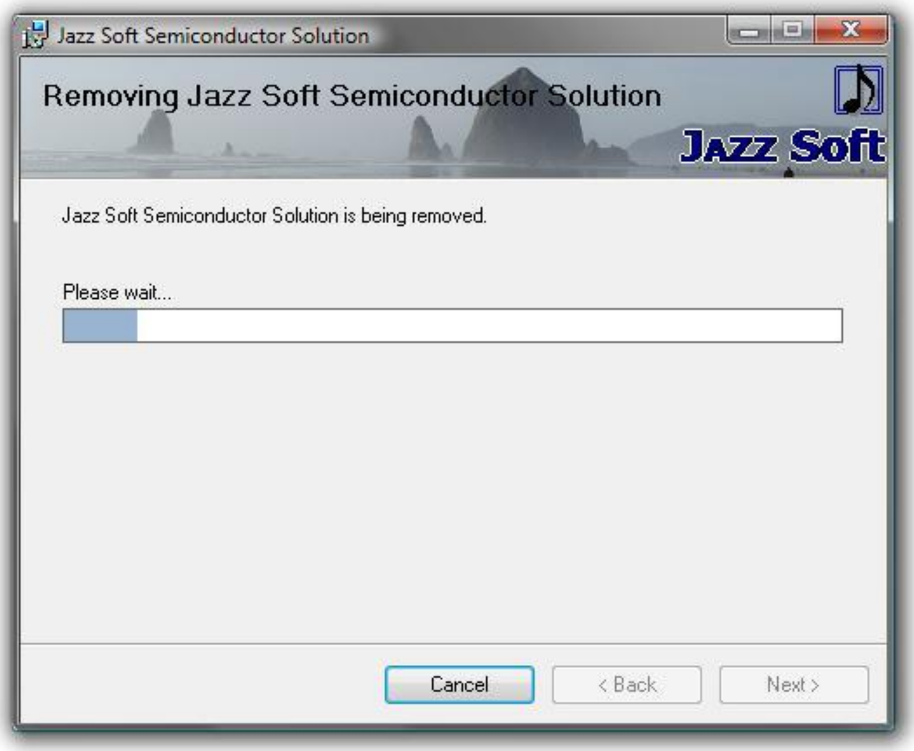
Welcome to the Jazz Soft Semiconductor Solution Setup Wizard

Select whether you want to repair or remove Jazz Soft Semiconductor Solution.

( ) Repair Jazz Soft Semiconductor Solution

(●) Remove Jazz Soft Semiconductor Solution

2 Actual uninstallation will start.

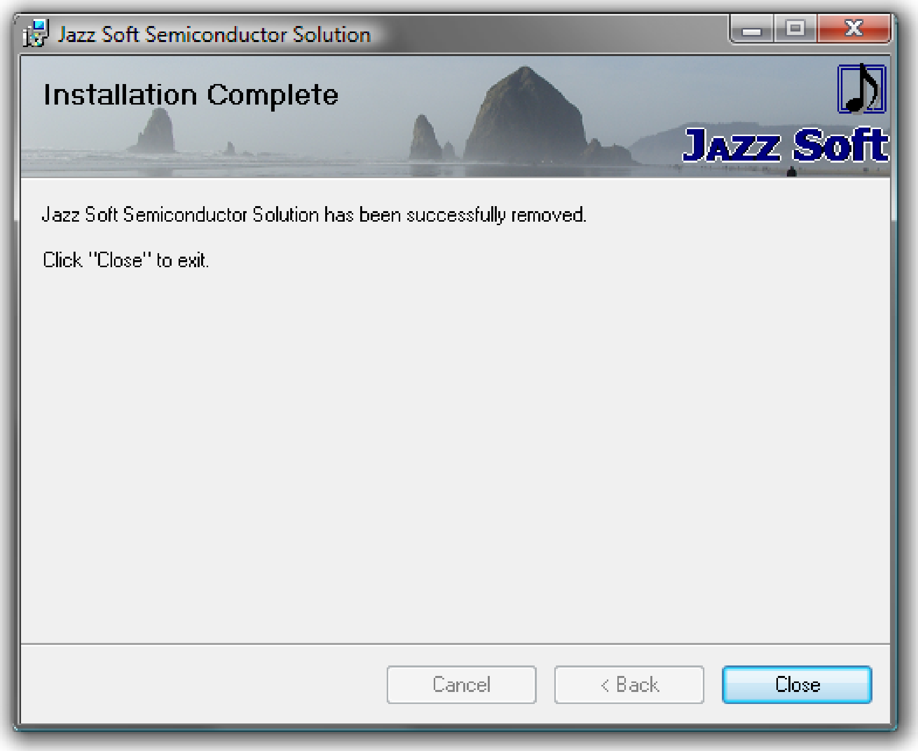

Jazz Soft Semiconductor Solution

Removing Jazz Soft Semiconductor Solution

Jazz Soft Semiconductor Solution is being removed.

Please wait...

3 Once uninstallation was done successfully, following screen will appear. Click “Close” button.


Jazz Soft Semiconductor Solution

Installation Complete

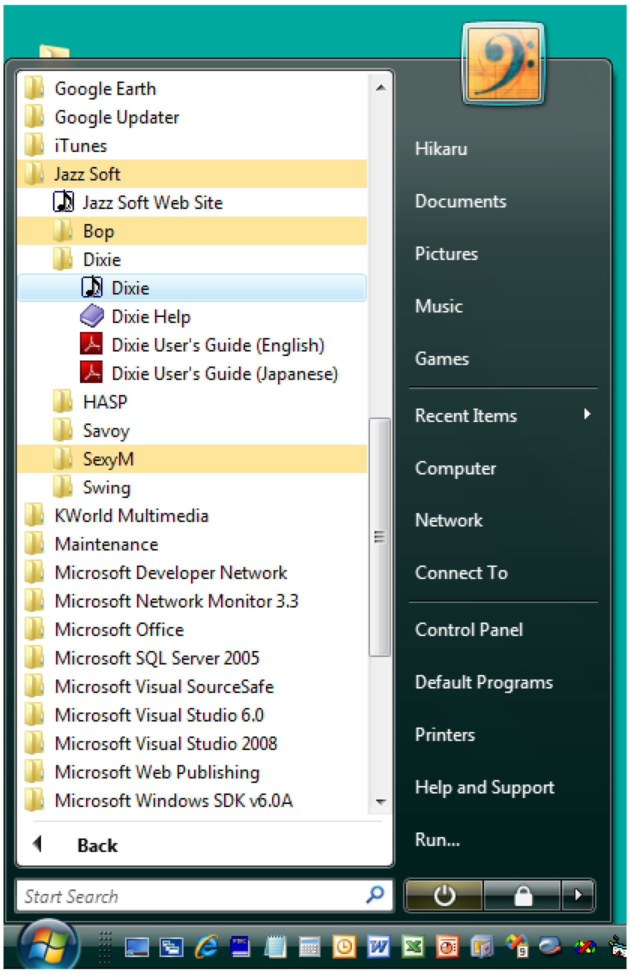
Jazz Soft Semiconductor Solution has been successfully removed.

Click "Close" to exit.

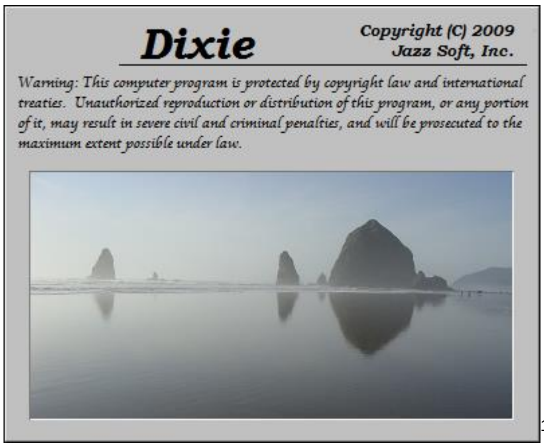
4. Operation Summary

2.4 How to Launch Dixie

1 Click “Dixie” from [Jazz Soft] – [Dixie] in Start menu.



2 Splash screen will appear followed by Dixie main screen.

[1]

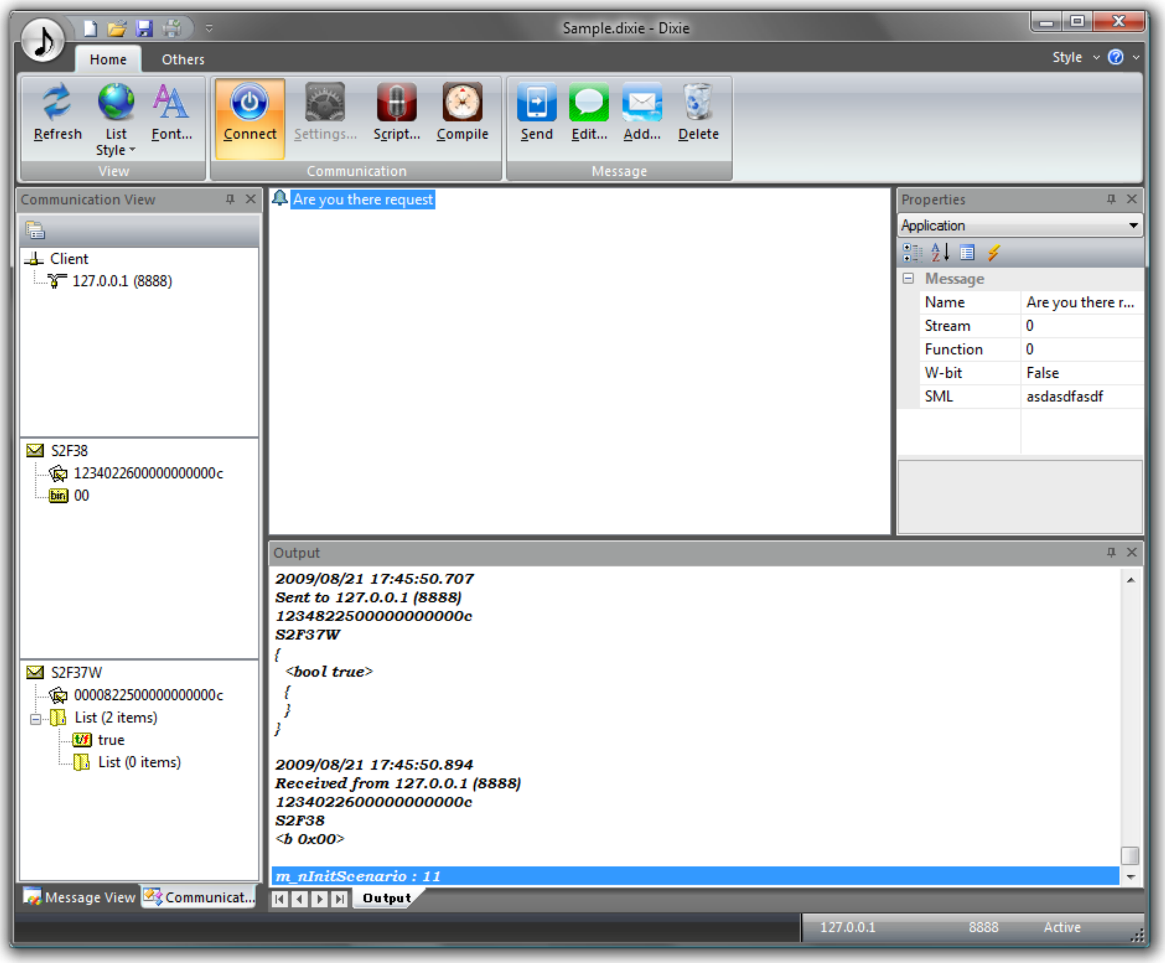
Dixie

Copyright (C) 2009
Jazz Soft, Inc.

Warning: This computer program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

[1] この写真は映画「グーニーズ」の舞台となったアメリカのオレゴン州キャノンビーチで撮影されました。

3 Dixie complies with Microsoft Office 2007 user interface guide line.



```
Client
  127.0.0.1 (8888)

S2F38
  1234022600000000000c
  00

S2F37W
  0000822500000000000c
  List (2 items)
    true
    List (0 items)
```

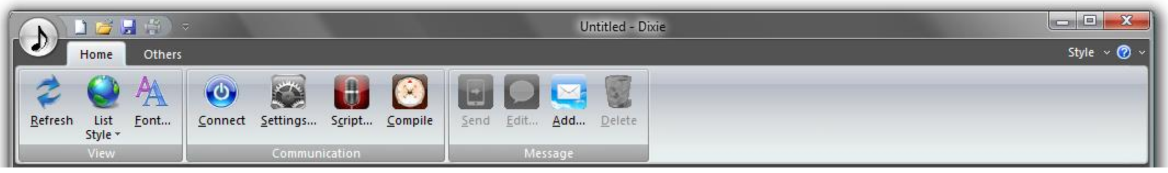
Message	
Name	Are you there r...
Stream	0
Function	0
W-bit	False
SML	asdasdfasdf

```
2009/08/21 17:45:50.707
Sent to 127.0.0.1 (8888)
1234822500000000000c
S2F37W
{
  <bool true>
  {
  }
}

2009/08/21 17:45:50.894
Received from 127.0.0.1 (8888)
1234022600000000000c
S2F38
<b 0x00>

m_nInitScenario : 11
```

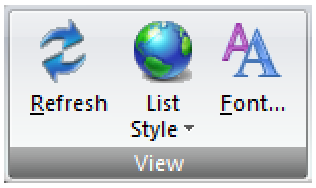
2.5 Ribbon

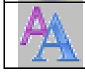
Dixie complies with Microsoft Office 2007 user interface guide line. There is no menu which typical software has it, but there are tabs called ribbon.

2.5.1 Home Category

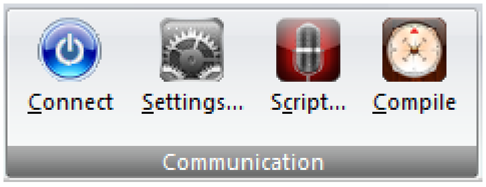


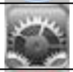
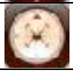
View



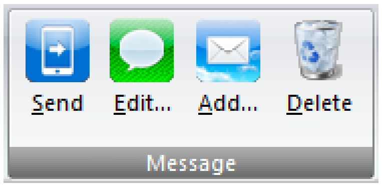
Icon	Item	Description
	Refresh	Refresh message list.
	List Style	Select display style of message list. It should be either “List” or “Detail”.
	Font	Choose font which will be used in output window. Fixed-pitch font is recommended.

Communication



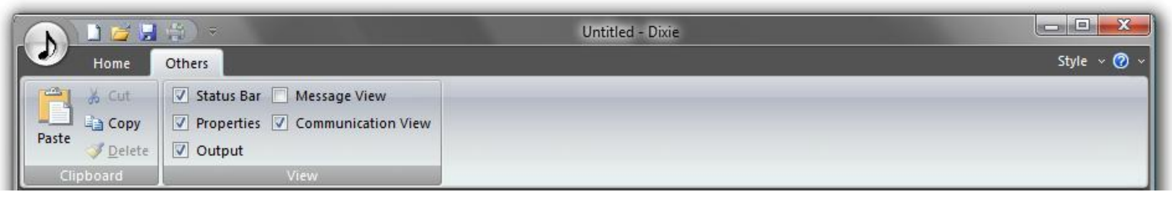
Icon	Item	Description
	Connect	Connect or disconnect the communication.
	Settings	Open option setting dialog box.
	Script	Edit script.
	Compile	Compile all the scripts.

Message

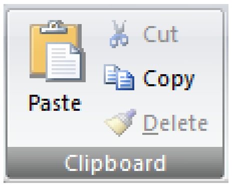


Icon	Item	Description
	Send	Send selected message.
	Edit	Edit selected message.
	Add	Add new message.
	Delete	Delete selected message.

2.5.2 Others Category

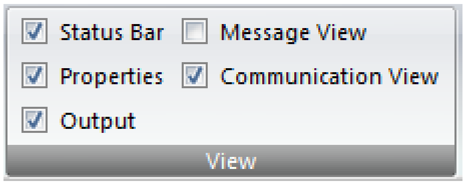


Clipboard



Icon	Item	Description
	Paste	Paste message from clipboard.
	Cut	(not used at the moment)
	Copy	Copy message to clipboard.
	Delete	(not used at the moment)

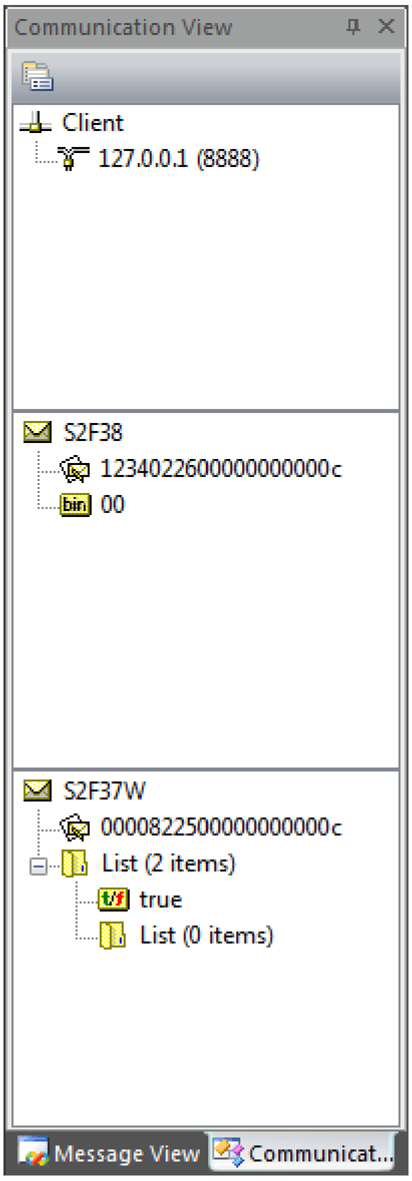
View



Icon	Item	Description
	Status Bar	Show or hide status bar.
	Properties	Show or hide property window.
	Output	Show or hide output window.
	Message View	Show or hide message view.
	Communication View	Show or hide communication view.

2.6 Communication View

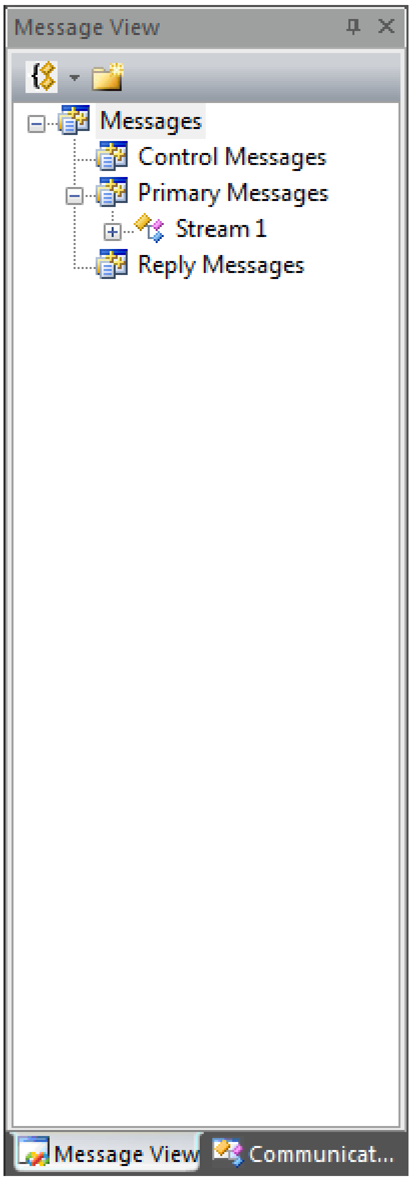
Displays communication status.　From the top, SavoyHsms and 2 SavoySecsII controls.



```
Client
  127.0.0.1 (8888)

S2F38
  1234022600000000000c
  00

S2F37W
  0000822500000000000c
  List (2 items)
    true
    List (0 items)
```

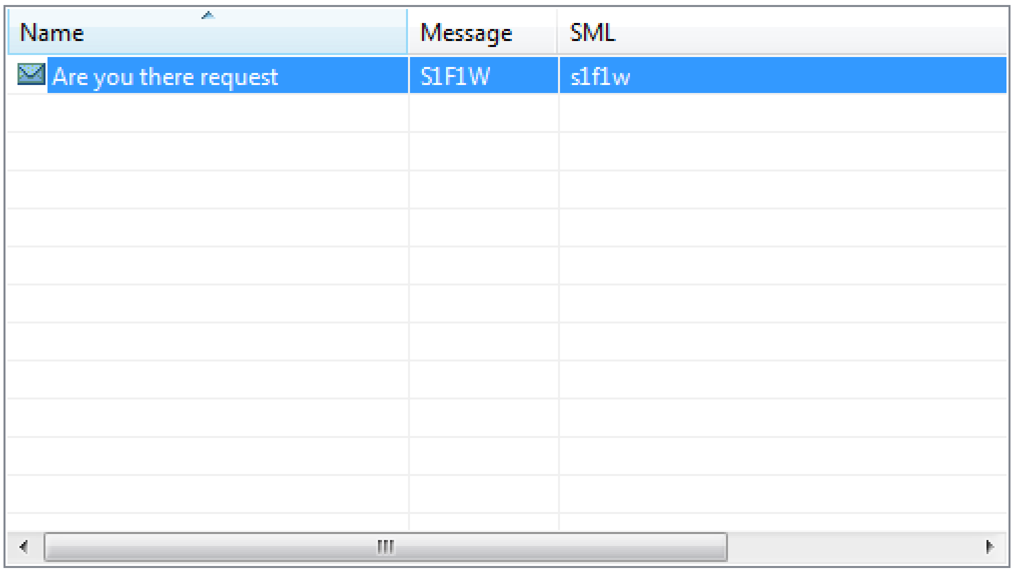
2.7 Message View

Display summary of registered messages. For example, “Primary Messages” is selected, only primary messages are displayed in message list. If user wants to display all the messages, Choose “Messages” (root).



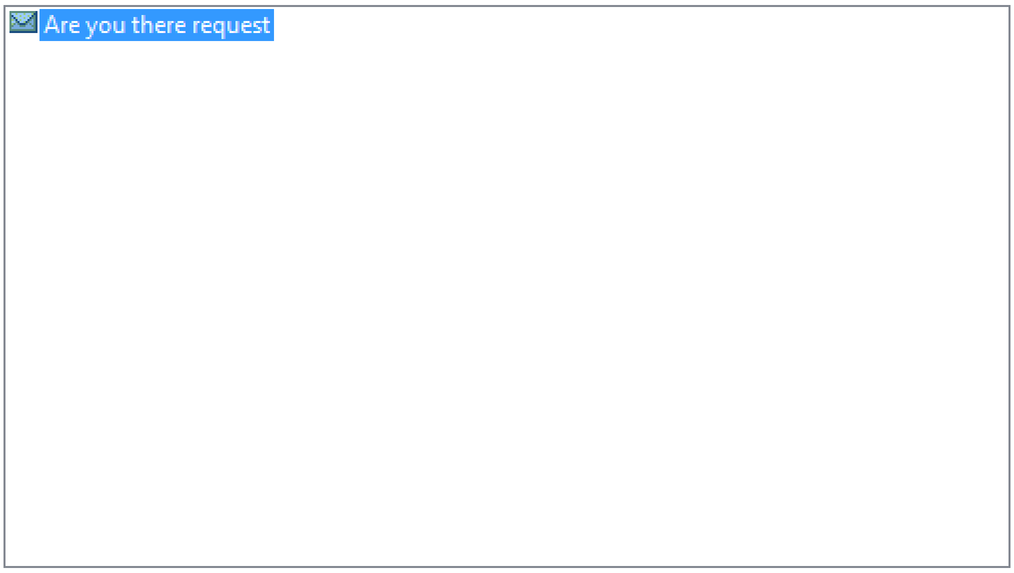
2.8 Message List

Displays list of registered messages. The way of displaying list depends on List Style setting. If “Detail” is selected, detailed information would be displayed as follows.



Name	Message	SML
Are you there request	S1F1W	s1f1w

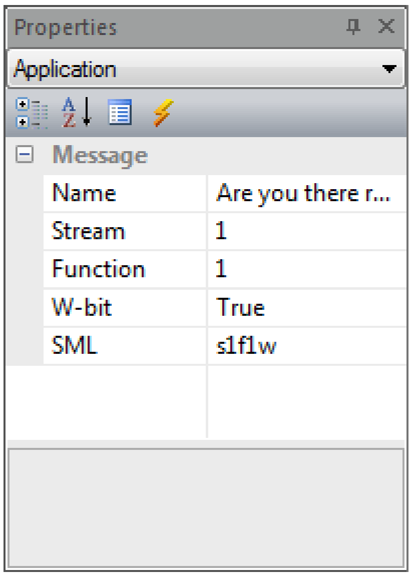
If “List” is selected, only name of the message would be displayed. If the number of message became large, this style may be more convenient.



Are you there request

2.9 Property WIndow

Properties of message are displayed.

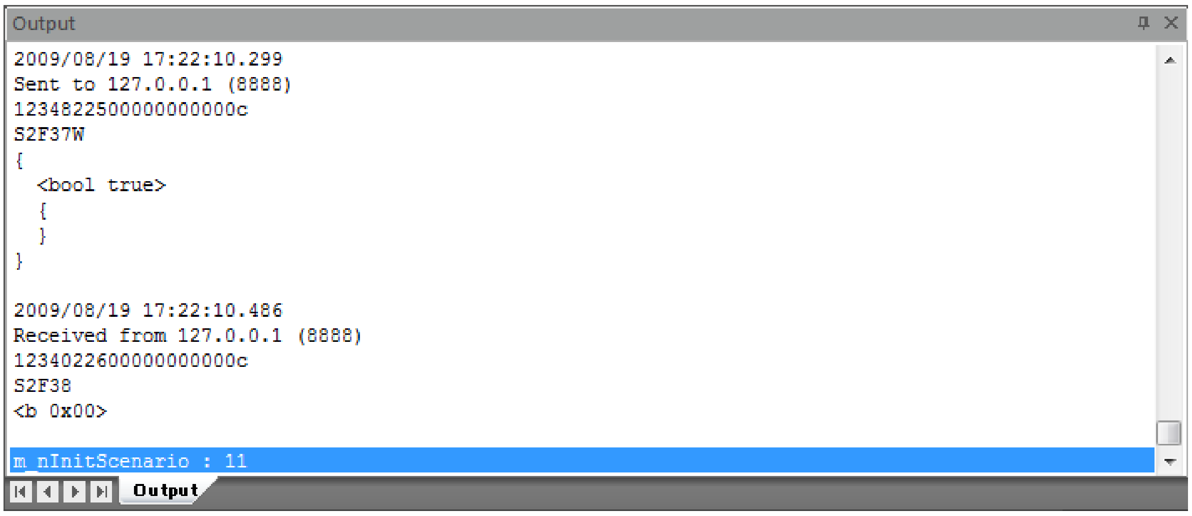


Message

Name	Are you there r...
Stream	1
Function	1
W-bit	True
SML	s1f1w

2.10 Output Window

Communication log and compilation status are displayed.

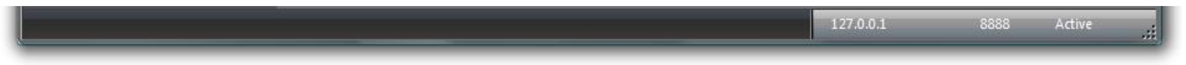


```
2009/08/19 17:22:10.299
Sent to 127.0.0.1 (8888)
1234822500000000000c
S2F37W
{
  <bool true>
  {
  }
}

2009/08/19 17:22:10.486
Received from 127.0.0.1 (8888)
1234022600000000000c
S2F38
<b 0x00>

m_nInitScenario : 11
```

If the bottom of line was selected, output window would be updated when new events happened.

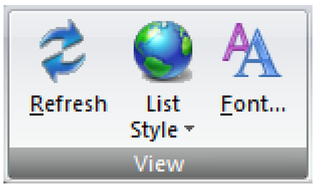
2.11 Status Bar

Display connection status.



5. Screen Detauls

2.12 Home – View Panel

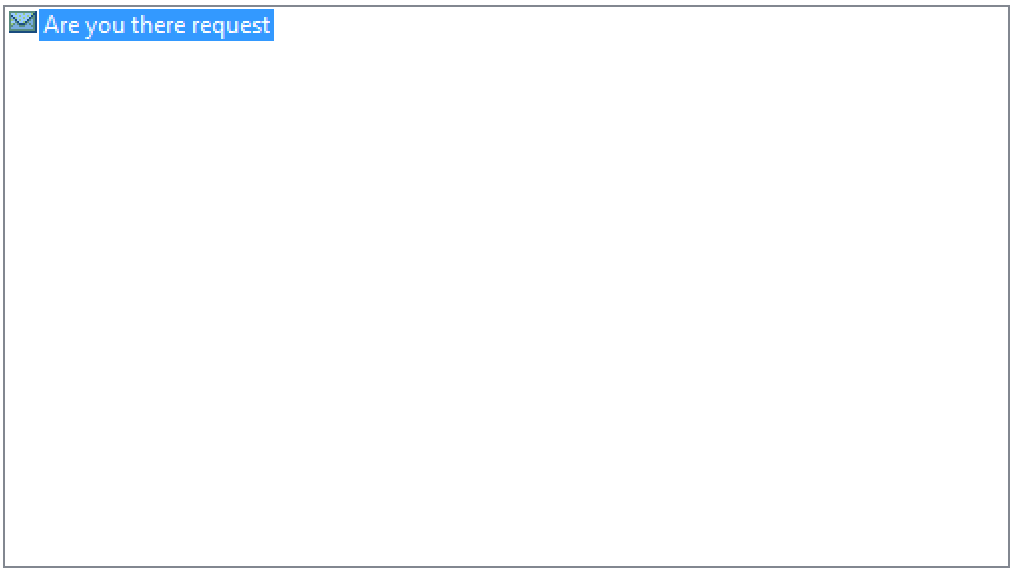


2.12.1 Refresh

Refresh message list.

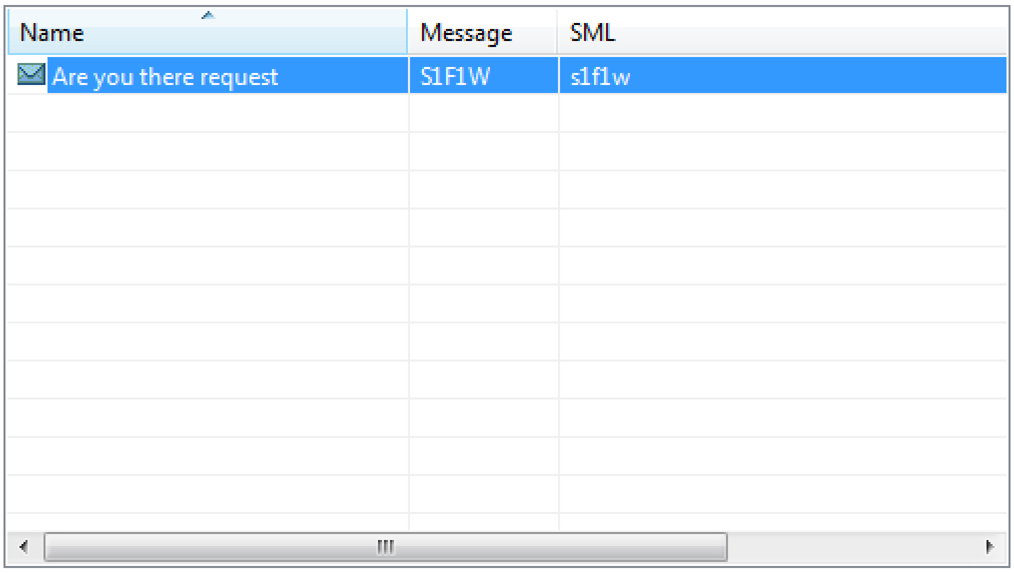
2.12.2 List Style

Select display style of message list. It should be either “List” or “Detail”. If “List” is selected, only name of the message would be displayed.



Are you there request

If “Detail” is selected, detailed information would be displayed as follows.

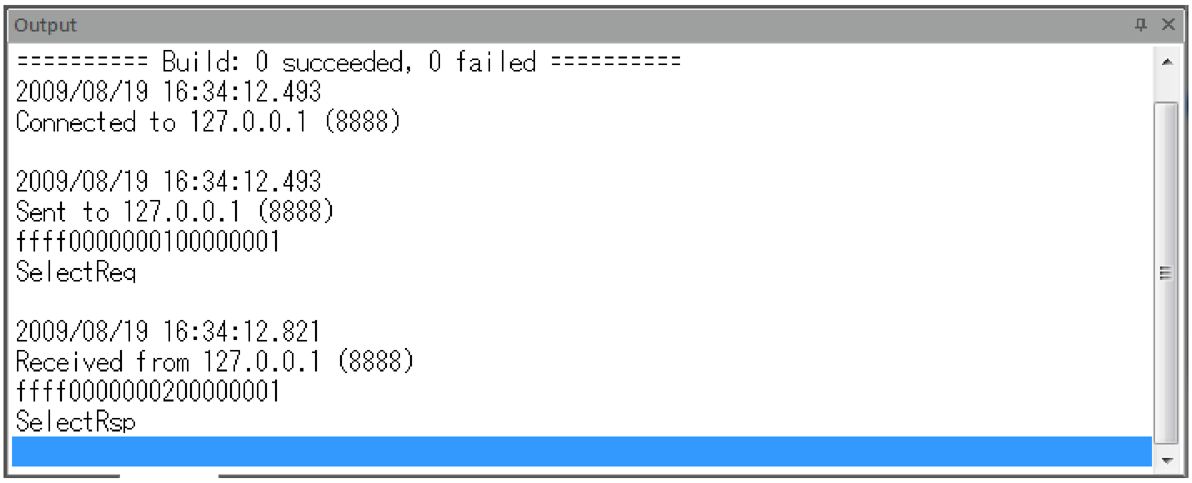


Name	Message	SML
Are you there request	S1F1W	s1f1w

2.12.3 Font

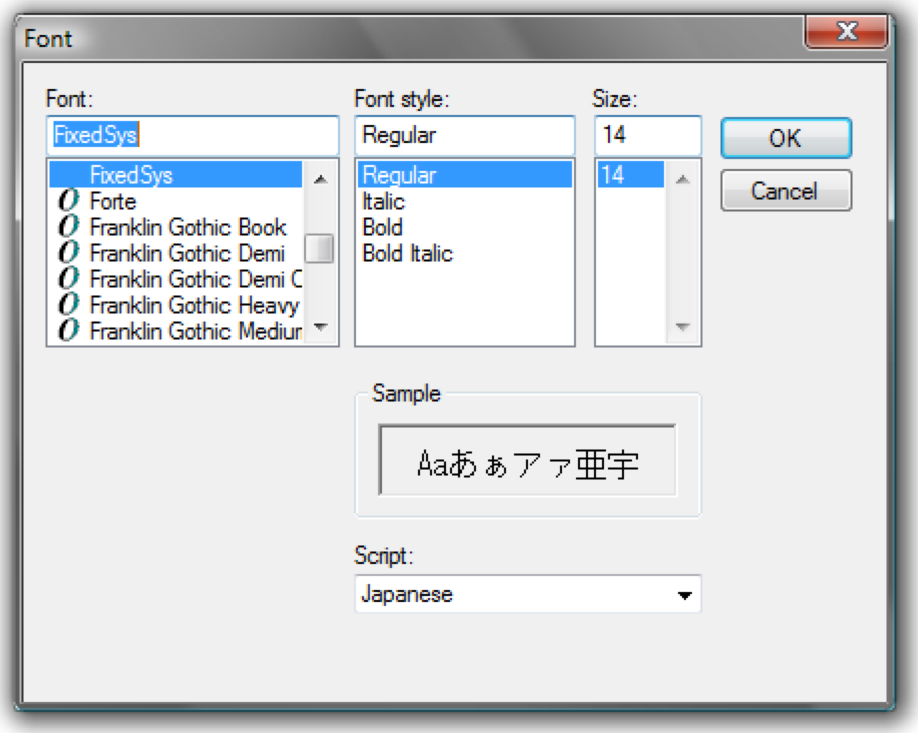
Choose font which is used in output window.　This font will also be used in SML string on Message edit dialog box and script string on script edit dialog box.

At the beginning FixedSys font is selected as follow.

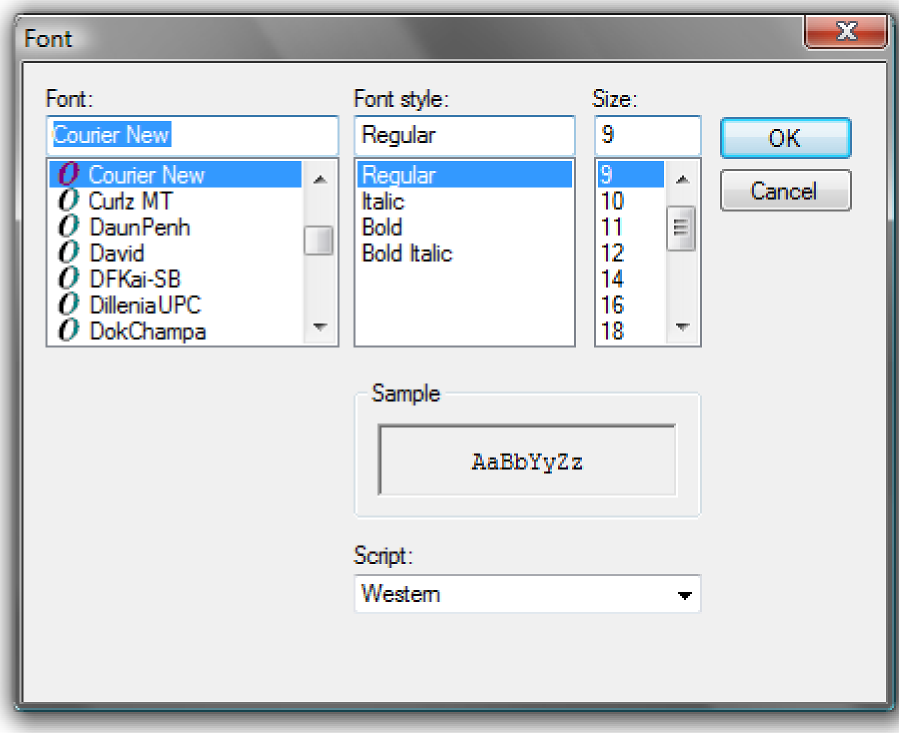


```
========== Build: 0 succeeded, 0 failed ==========
2009/08/19 16:34:12.493
Connected to 127.0.0.1 (8888)

2009/08/19 16:34:12.493
Sent to 127.0.0.1 (8888)
ffff0000000100000001
SelectReq

2009/08/19 16:34:12.821
Received from 127.0.0.1 (8888)
ffff0000000200000001
SelectRsp
```

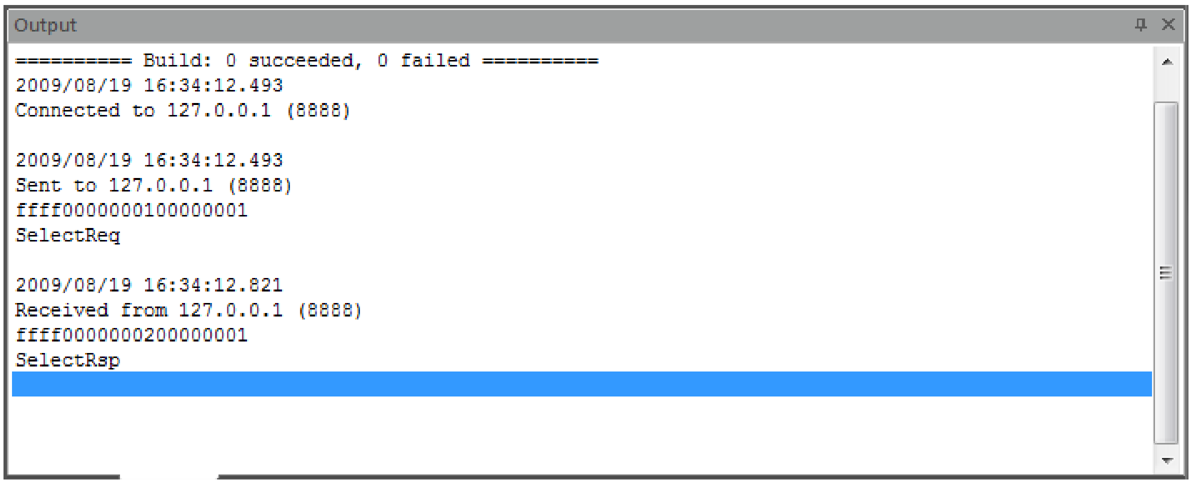
Open font select dialog box.



Let’s change to 9 points Courier New font.　Fixed-pitch would be recommended, since it would be easier to read SML string.

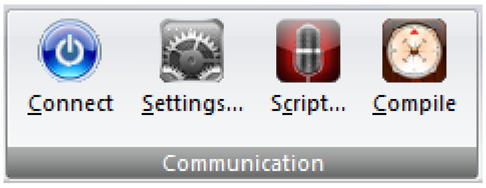


Output window will be updated by using newly selected font.



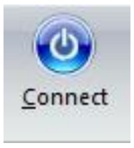
```
========== Build: 0 succeeded, 0 failed ==========
2009/08/19 16:34:12.493
Connected to 127.0.0.1 (8888)

2009/08/19 16:34:12.493
Sent to 127.0.0.1 (8888)
ffff000000100000001
SelectReq

2009/08/19 16:34:12.821
Received from 127.0.0.1 (8888)
ffff000000200000001
SelectRsp
```

2.13 Home – Communication Panel

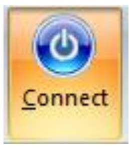


2.13.1 Connect

Connect or disconnect communication. When communication is disconnected, the button is displayed in gray color as follow.



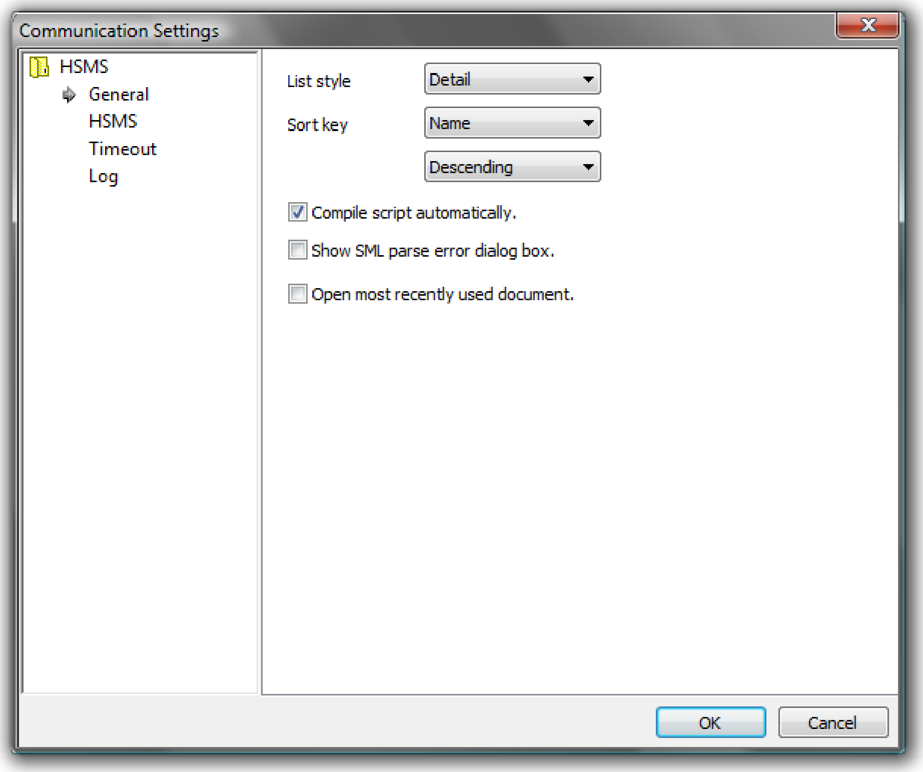
When communication was established, the button is displayed in orange color as follow.



2.13.2 Settings

Open option setting dialog box.　When connection is active, this button will be disabled.

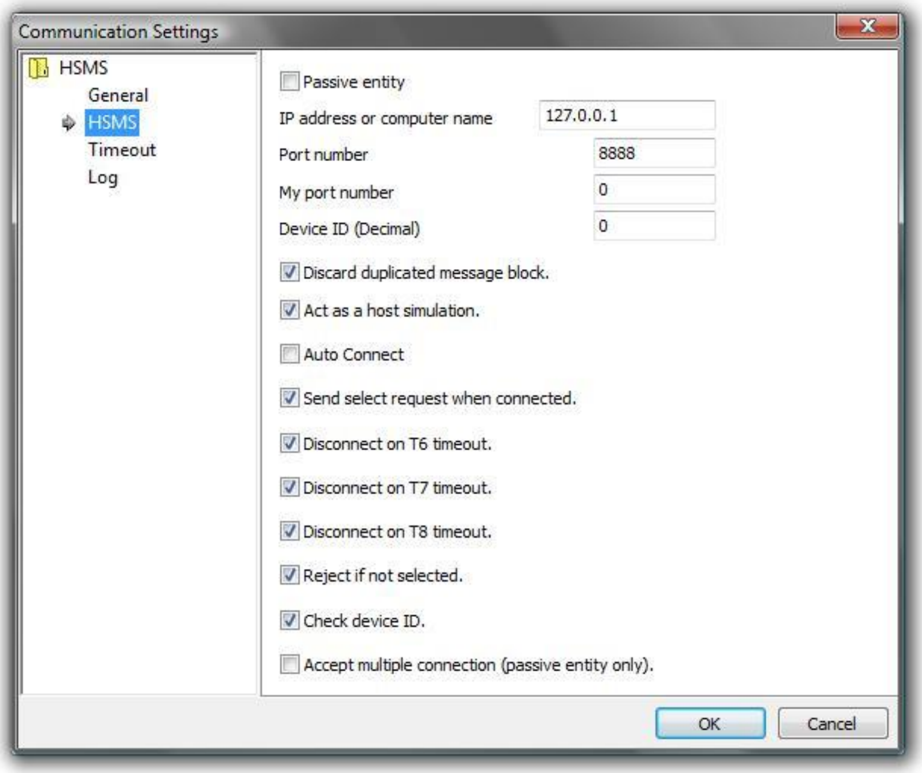
General



List style: Detail
Sort key: Name
Descending
☑ Compile script automatically.
☐ Show SML parse error dialog box.
☐ Open most recently used document.

Item	Description
List style	Choose display style in message list.　It should be either “List” or “Detail”.　If “List” was selected, only name would appear in list.　If “Detail” was selected, all the information will be displayed.
Sort key	Choose sort key for message list.　It should be one of “Name”, “Message” or “SML”.
	Choose sort order for message list.　It should be one of “Ascending”or “Descending”.
Compile script automatically.	If at least one script was modified, all the scripts would be compiled automatically.　However, all the scripts are compiled at the time Dixie loaded .dixie file, whatever this setting is.
Show SML parse error dialog box.	If message was modified, check SML string.　If there is an error in syntax, error message box will pop up.
Open most recently used document.	Most recently used .dixie file will automatically be loaded next time user launch Dixe.　This setting will be stored in Windows system registry.

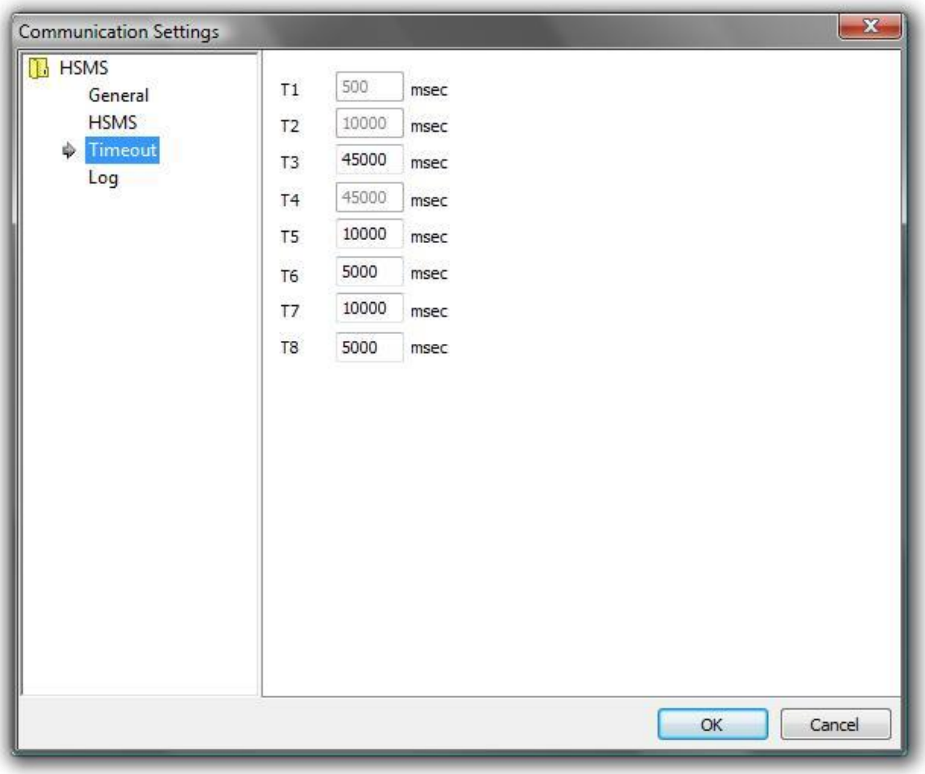
HSMS



- [ ] Passive entity
- IP address or computer name: 127.0.0.1
- Port number: 8888
- My port number: 0
- Device ID (Decimal): 0
- [✓] Discard duplicated message block.
- [✓] Act as a host simulation.
- [ ] Auto Connect
- [✓] Send select request when connected.
- [✓] Disconnect on T6 timeout.
- [✓] Disconnect on T7 timeout.
- [✓] Disconnect on T8 timeout.
- [✓] Reject if not selected.
- [✓] Check device ID.
- [ ] Accept multiple connection (passive entity only).

Item	Description
Passive entity	Select passive entity (server) or active entity (client).
IP address or computer name	Server IP address or computer name.　If this setting is “” (empty), Dixie assumes it as local computer.　This setting will be ignored, if the setting is passive entity.
Port number	Server port number.　This setting will be ignored, if the setting is passive entity.
My port number	Specify local port number.　It is strongly recommend using 0 on active entity.
Device ID (Decimal)	Device ID.
Discard duplicated message block.	If exactly same message arrived more than once in a row, the second or later one would be ignored.
Act as a host simulation.	If checked, Dixie acts as host simulator.　Otherwise, Dixie acts as equipment simulator.
Auto Connect	Automatically connect when Dixie boots up.
Send select request when connected.	If TCP/IP connection was established, send select request automatically.　In HSMS specification, active entity must send select request.
Disconnect on T6 timeout.	If T6 timeout occurred, disconnect connection.
Disconnect on T7 timeout.	If T7 timeout occurred, disconnect connection.
Disconnect on T8 timeout.	If T8 timeout occurred, disconnect connection.
Reject if not selected.	If entity is not selected, ignore all SECS-II data messages and reply reject request.
Check device ID.	If device ID doesn’t match, ignore message.　If Dixie acts as equipment simulator, S9F1 will be sent.
Accept multiple connections.	Accept more than one connection.　This setting will be ignored, if the setting is active entity.

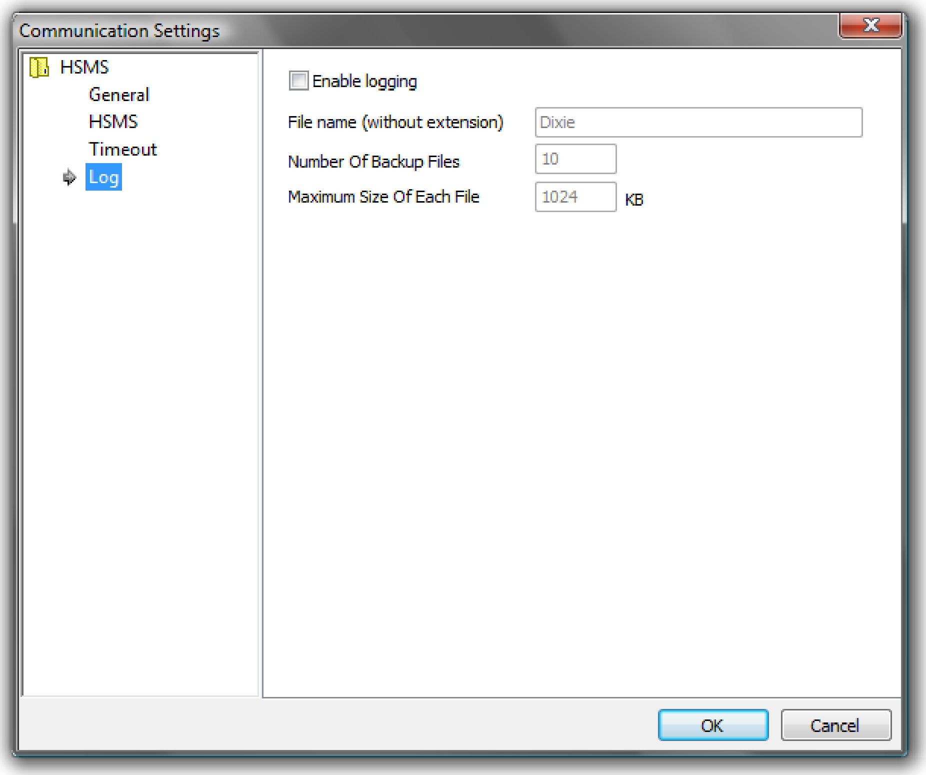
Timeout



T1 500 msec
T2 10000 msec
T3 45000 msec
T4 45000 msec
T5 10000 msec
T6 5000 msec
T7 10000 msec
T8 5000 msec

Item	Description
T1	(not used at the moment)
T2	(not used at the moment)
T3	T3 timer in milliseconds.
T4	(not used at the moment)
T5	T5 timer in milliseconds.
T6	T6 timer in milliseconds.
T7	T7 timer in milliseconds.
T8	T8 timer in milliseconds.

Log



Enable logging [ ]

File name (without extension): Dixie

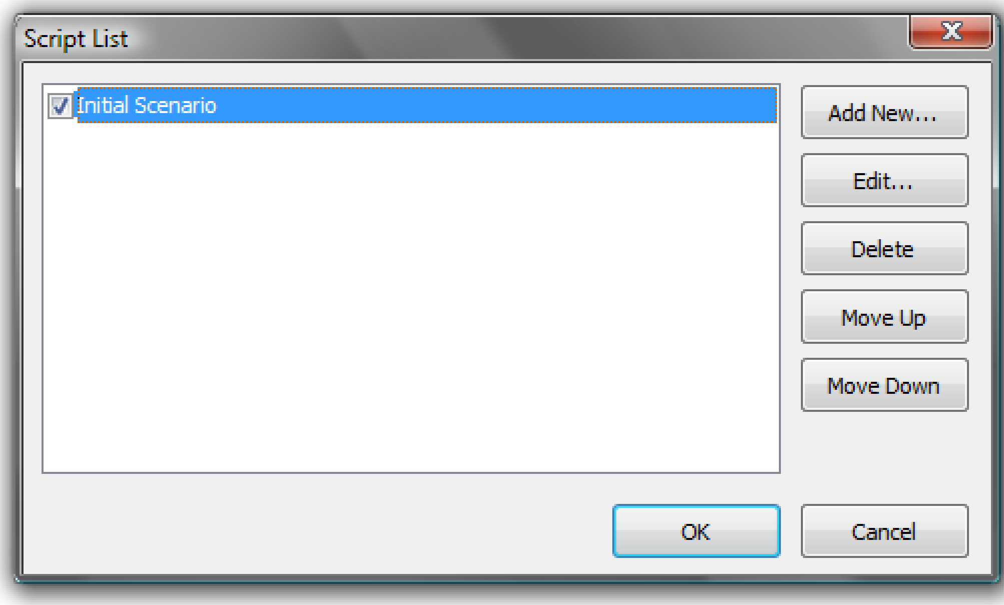
Number Of Backup Files: 10

Maximum Size Of Each File: 1024 KB

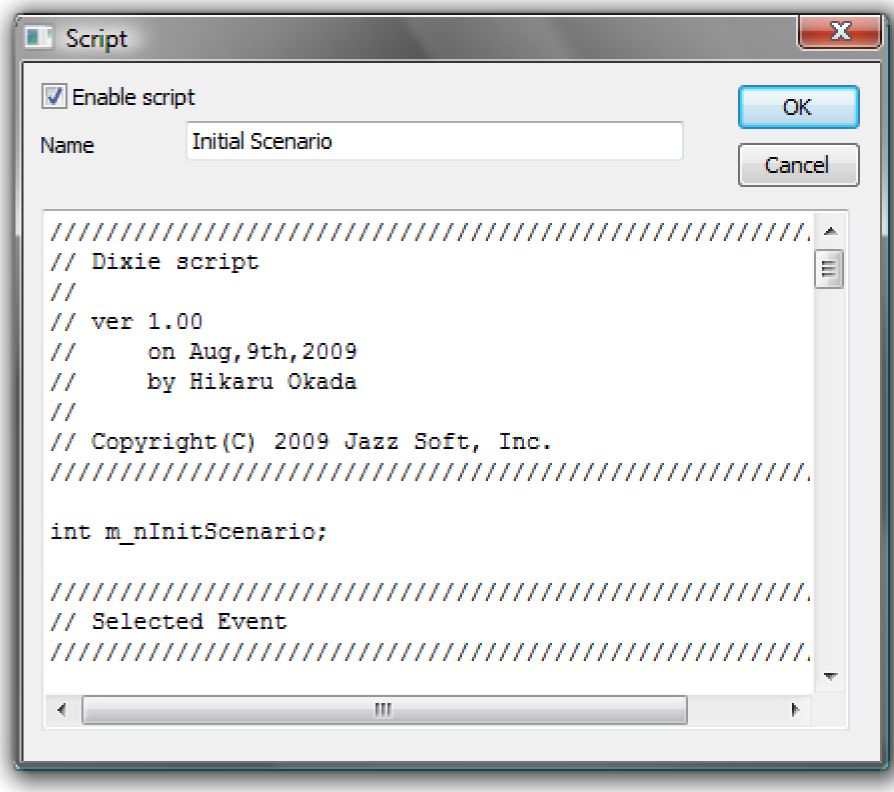
Item	Description
Enable logging	Write communication details into log file.
File name (without extension)	Log file name.　Don't include file extension.　Dixie will append file extension.
Number of backup files	Number of backup files.　If the number of backup file exceeded to this number, Dixie would delete the backup file from oldest one.
Maximum size of each file	Log file size in kilobytes.　If log file size exceeded to this number, Dixie would make backup file.

2.13.3 Script

Edit script. If multiple scripts are registered, upper script has higher priority to be executed.



Edit source code of script. This dialog box is resizable. Since there is no fancy editing feature on this dialog box, it is recommended to edit script source code on other editor software such as Visual Studio 2008 and then copy & paste it.

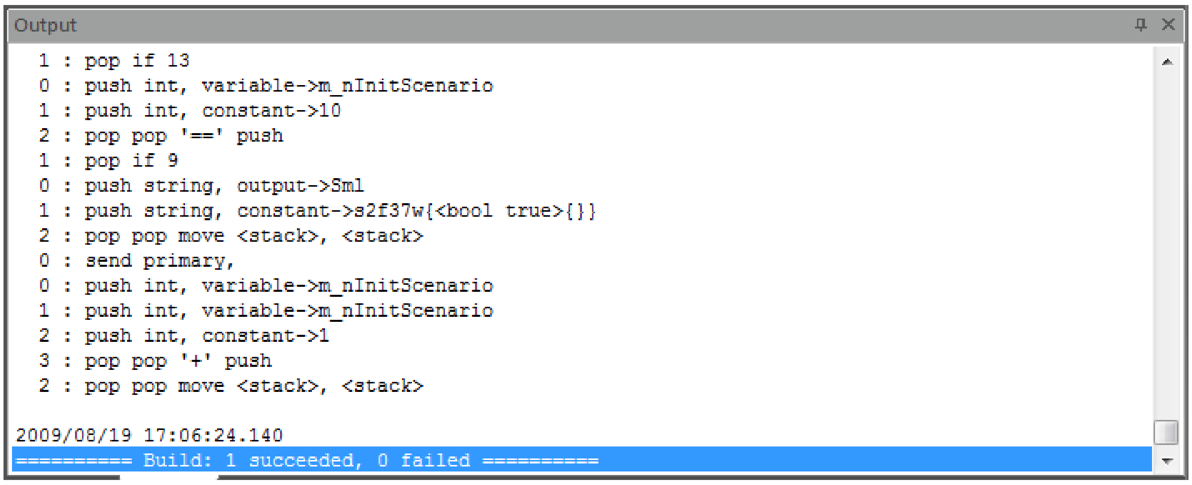


```
/////////////////////////////////////////////////////////
// Dixie script
//
// ver 1.00
//      on Aug,9th,2009
//      by Hikaru Okada
//
// Copyright(C) 2009 Jazz Soft, Inc.
/////////////////////////////////////////////////////////

int m_nInitScenario;

/////////////////////////////////////////////////////////
// Selected Event
/////////////////////////////////////////////////////////
```

2.13.4 Compile

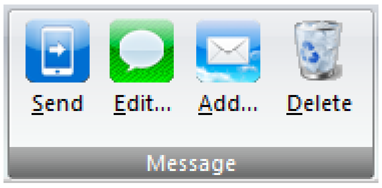
Compile script. If multiple scripts were registered, all the scripts would be compiled. If syntax error occurred during compiling, Dixie would terminate compilation and discard all compiled binaries. If the message "0 failed" is displayed as a compilation result in output window, it indicates compilation was successfully completed.

The mnemonic code of compiled binary will be dumped in output window. The number on the left means stack depth.



```
  1 : pop if 13
  0 : push int, variable->m_nInitScenario
  1 : push int, constant->10
  2 : pop pop '==' push
  1 : pop if 9
  0 : push string, output->Sml
  1 : push string, constant->s2f37w{<bool true>{}}
  2 : pop pop move <stack>, <stack>
  0 : send primary,
  0 : push int, variable->m_nInitScenario
  1 : push int, variable->m_nInitScenario
  2 : push int, constant->1
  3 : pop pop '+' push
  2 : pop pop move <stack>, <stack>

2009/08/19 17:06:24.140
========== Build: 1 succeeded, 0 failed ==========
```

2.14 Home – Message Panel



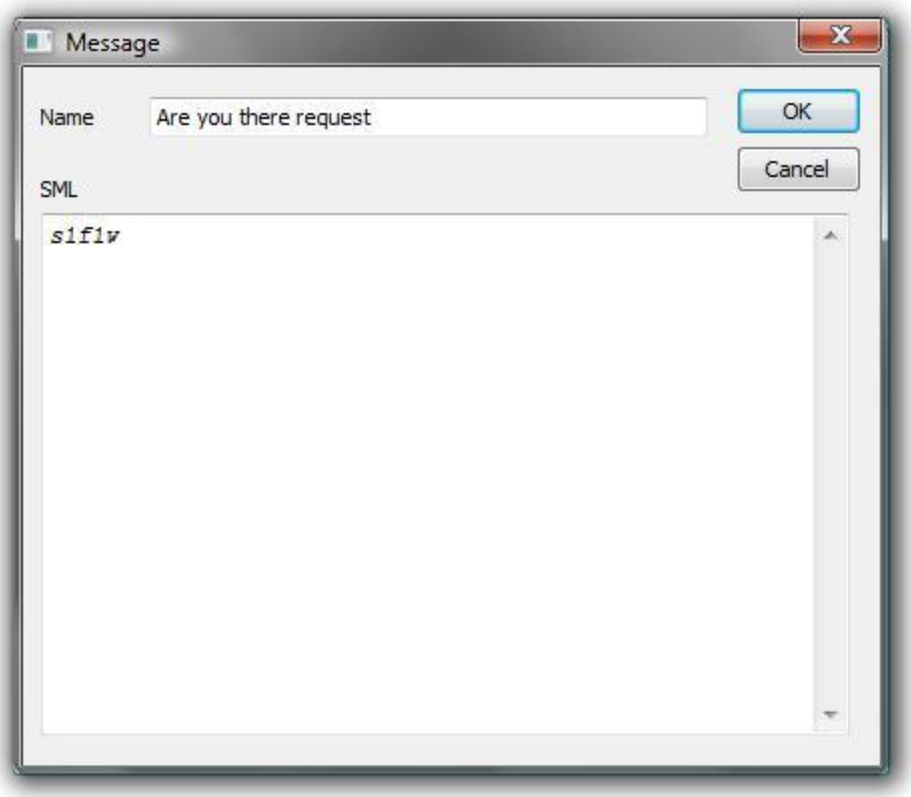
2.14.1 Send

Send selected message. If no message was selected, this button would be disabled.

2.14.2 Edit

Edit selected message. If no message was selected, this button would be disabled.

This dialog box is resizable. Since there is no fancy editing feature on this dialog box, it is recommended to edit SML string on other editor software such as Visual Studio 2008 and then copy & paste it.



Name Are you there request

SML

```
s1f1v
```

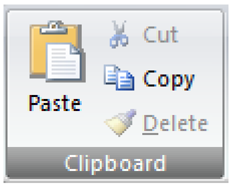
2.14.3 Add

Add new message.

2.14.4 Delete

Delete selected message.   If no message was selected, this button would be disabled.

2.15 Others – Clipboard Panel



2.15.1 Paste

Paste message from clipboard.   If Dixie message was not copied in clipboard, this button would be disabled.

2.15.2 Cut

This function is not implemented at the moment.

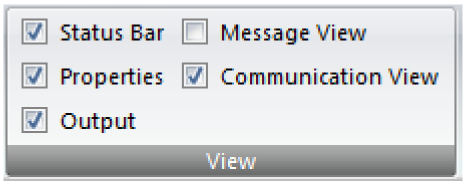
2.15.3 Copy

Copy currently selected message to clipboard.   If message was not selected, this button would be disabled.

2.15.4 Delete

This function is not implemented at the moment.

2.16 Others – View Panel



2.16.1 Status Bar

Show or hide status bar.

2.16.2 Properties

Show or hide property window.

2.16.3 Output

Show or hide output window.

2.16.4 Message View

Show or hide message view.

2.16.5 Communication View

Show or hide communication view.

6. Tutorial

2.17 Initialization Script

This tutorial will guide you through how to write a script. Let’s make a host simulator which will send series of GEM initialization sequence.

2.17.1 Specification of Initialization Scenario

The specification of initialization scenario is as follows. Dixie acts as host simulator and the tool is a general wafer inspection tool.

The aim of this script is to run scenario when Dixie was booted and connected with equipment.

#	Dixie			Equipment
1	S1F13	→→→→→		
1a			←←←←←	S1F14
2	S1F1	→→→→→		
2a			←←←←←	S1F2
3	S2F37 (disable event)	→→→→→		
3a			←←←←←	S2F38
4	S2F33 (delete report)	→→→→→		
4a			←←←←←	S2F34
5	S2F33 (define report)	→→→→→		
5a			←←←←←	S2F34
6	S2F35 (link report)	→→→→→		
6a			←←←←←	S2F36
7	S2F43 (disable spool)	→→→→→		
7a			←←←←←	S2F44
8	S5F3 (enable alarm)	→→→→→		
8a			←←←←←	S5F4
9	S2F15 (equipment constant)	→→→→→		
9a			←←←←←	S2F16
10	S1F3 (system variable)	→→→→→		
10a			←←←←←	S1F4
11	S2F37 (enable event)	→→→→→		
11a			←←←←←	S2F38

2.17.2 State Transition

Each line of above scenario is considered a “state”. For instance, if S1F13 was sent, state would transit to “waiting for S1F14” state. Once S1F14 arrived, then send S1F1 and transit to “waiting for S1F2” state. To know each state is important. This state transition idea is very useful not only for script processing, but also for making communication software with Savoy for host or equipment.

There are numbers at the beginning of each line. These are state numbers. There are total 12 states (11 + additionally “do nothing” state) in this tutorial.

Use variable to control state. Since state is numeric, use integer variable. Under Dixie naming convention, all the variable names should start at “m_”.

Dixie Script

```
int m_nInitScenario;
```

2.17.3 Selected Event Procedure

To proceed with scenario, determine wether state can be transitable by checking events such as “connected with equipment” and/or “received S1F2”.

The trigger to start scenario is the event “HSMS communication became selected state”. Write OnSelected() event handler function in order to capture this event. It may be a good idea to output a comment which describes summary of scenario.

Dixie Script

```
bool OnSelected()
{
    cout << "Init scenario started";
}
```

The first state is “send S1F13 and wait for S1F14”.　Set SML string of S1F13 and call Send() function.　And then set 1 to m_nInitScenario.

Dixie Script

```
    out.SML = "s1f13w{}";
    Send();
    m_nInitScenario = 1;
```

2.17.4 Received Event Procedure

Next, it is needed to capture the event for reception of S1F14.　Write OnReceived() event handler function to capture the event.　As a comment, output the content of m_nInitScenario.

Dixie Script

```
bool OnReceived()
{
    cout << "m_nInitScenario : " + m_nInitScenario;
}
```

Check stream and function number of incoming message by using “if” statement.　If stream and function match to S1F14, that would be the expected message.　But Dixie might receive S1F14 in unexpected context.　Check to make sure m_nInitScenario is 1.

Next state is “send S1F1 and wait for S1F2”.　Set SML string fo S1F1 and call Send() function to send message. And don’t forget to increment m_nInitScenario by 1.

Dixie Script

```
    if(in.Stream == 1 && in.Function == 14)
    {
        // S1F14
        if(m_nInitScenario == 1)
        {
            out.SML = "s1f1w";
            Send();
            m_nInitScenario = m_nInitScenario + 1;
        }
    }
```

2.17.5 Receiving Message with Same Stream and Function

Looking at scenario carefully, S2F37 and S2F33 transaction will occur twice respectively.　Equipment will reply S2F38 and S2F33 respectively, however, how can Dixie discriminate which is which?　The answer is also to use m_nInitScenario variable.

For example, the states to send S2F34 are state 4 and state 5.　Just use “if” statement to sort by state.

Dixie Script

```
    if(in.Stream == 2 && in.Function == 34)
    {
        // S2F34
        if(m_nInitScenario == 4)
        {
            out.SML = "s2f33w{<bool true>}";
            Send();
            m_nInitScenario = m_nInitScenario + 1;
        }
        else
        if(m_nInitScenario == 5)
        {
            out.SML = "s2f35w{}";
            Send();
            m_nInitScenario = m_nInitScenario + 1;
        }
    }
```

2.17.6 Entire Source Code

Writing procedure can mechanically be done to complete initialization script. It is recommended to insert a lot of comments among script same as software development. Following is an entire source code of script.

Dixie Script

```
////////////////////////////////////////////////////////////
// Dixie script
//
// ver 1.00
//      on Aug,9th,2009
//      by Hikaru Okada
//
// Copyright(C) 2009 Jazz Soft, Inc.
////////////////////////////////////////////////////////////

int m_nInitScenario;

////////////////////////////////////////////////////////////
// Selected Event
////////////////////////////////////////////////////////////
bool OnSelected()
{
    cout << "Init scenario started";

    out.SML = "s1f13w{}";
    Send();
    m_nInitScenario = 1;
}

////////////////////////////////////////////////////////////
// Received Event
////////////////////////////////////////////////////////////
bool OnReceived()
{
    cout << "m_nInitScenario : " + m_nInitScenario;

    if(in.Stream == 1 && in.Function == 14)
    {
        // S1F14
        if(m_nInitScenario == 1)
        {
            out.SML = "s1f1w";
            Send();
            m_nInitScenario = m_nInitScenario + 1;
        }
    }
```

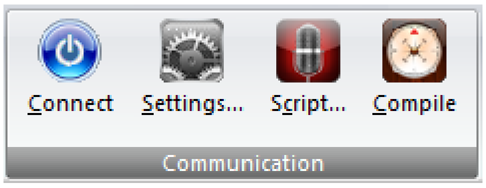
```
else
if(in.Stream == 1 && in.Function == 2)
{
    // S1F2
    if(m_nInitScenario == 2)
    {
        out.SML = "s2f37w{<bool false>{}}";
        Send();
        m_nInitScenario = m_nInitScenario + 1;
    }
}
else
if(in.Stream == 2 && in.Function == 38)
{
    // S2F38
    if(m_nInitScenario == 3)
    {
        out.SML = "s2f33w{<bool false>}";
        Send();
        m_nInitScenario = m_nInitScenario + 1;
    }
    else
    if(m_nInitScenario == 11)
    {
        m_nInitScenario = 0;
    }
}
else
if(in.Stream == 2 && in.Function == 34)
{
    // S2F34
    if(m_nInitScenario == 4)
    {
        out.SML = "s2f33w{<bool true>}";
        Send();
        m_nInitScenario = m_nInitScenario + 1;
    }
    else
    if(m_nInitScenario == 5)
    {
        out.SML = "s2f35w{}";
        Send();
        m_nInitScenario = m_nInitScenario + 1;
    }
}
else
if(in.Stream == 2 && in.Function == 36)
{
    // S2F36
    if(m_nInitScenario == 6)
    {
        out.SML = "s2f43w{}";
        Send();
        m_nInitScenario = m_nInitScenario + 1;
    }
}
else
if(in.Stream == 2 && in.Function == 44)
{
    // S2F44
    if(m_nInitScenario == 7)
    {
        out.SML = "s5f3w{<b 0x80><u4>}";
        Send();
        m_nInitScenario = m_nInitScenario + 1;
    }
}
else
```

```
        if(in.Stream == 5 && in.Function == 4)
        {
            // S5F4
            if(m_nInitScenario == 8)
            {
                out.SML = "s2f15w{}";
                Send();
                m_nInitScenario = m_nInitScenario + 1;
            }
        }
        else
        if(in.Stream == 2 && in.Function == 16)
        {
            // S2F16
            if(m_nInitScenario == 9)
            {
                out.SML = "s1f3w{}";
                Send();
                m_nInitScenario = m_nInitScenario + 1;
            }
        }
        else
        if(in.Stream == 1 && in.Function == 4)
        {
            // S1F4
            if(m_nInitScenario == 10)
            {
                out.SML = "s2f37w{<bool true>{}}";
                Send();
                m_nInitScenario = m_nInitScenario + 1;
            }
        }
    }
```

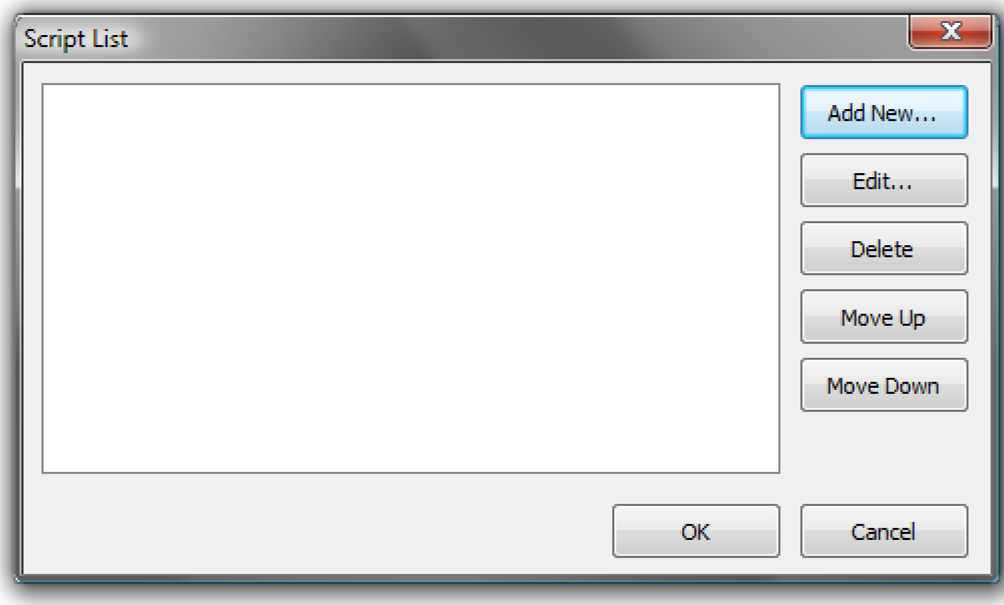
2.17.7 Add New Script

It must be easy to input above script using text editor such as Visual Studio 2008, and paste the script on Dixie script editor. Notepad can be used for text editor. If nothing was installed on the computer, just type it directly in Dixie.

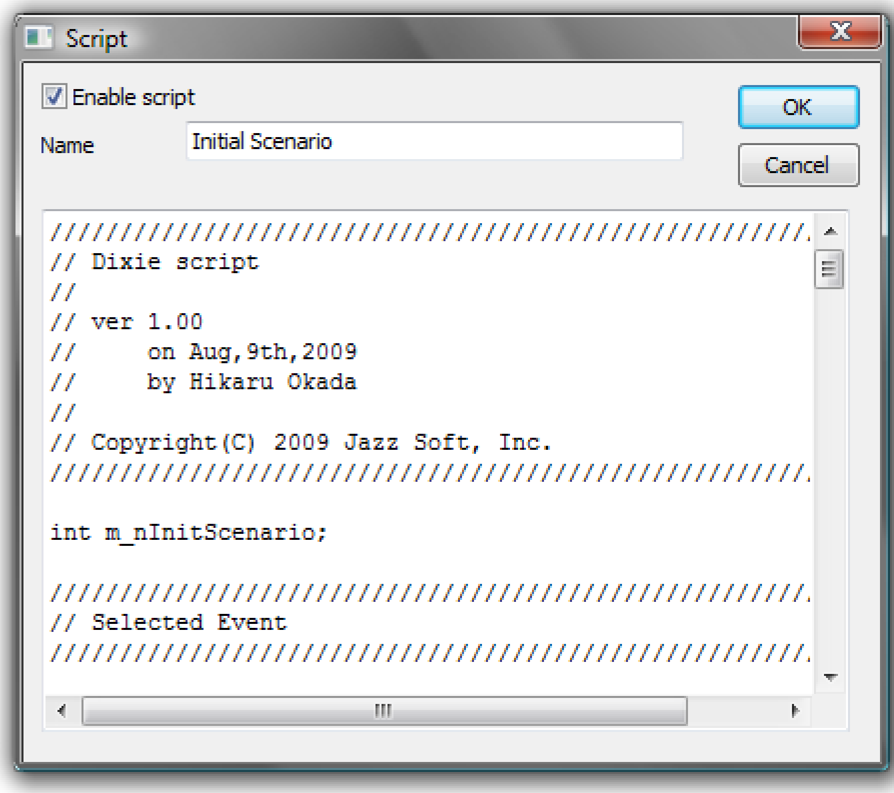
1 Launch Dixie and click “Script…” button from [Home] – [Communication] in ribbon.



2 Click “Add New…” button.



3 Input (or paste) script and click “OK” button.

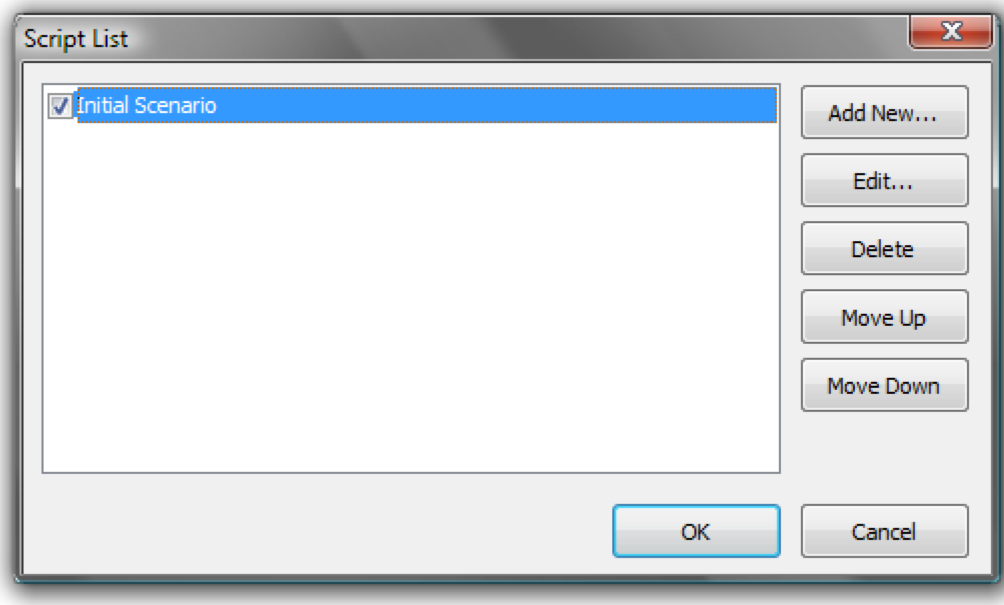


☑ Enable script

Name: Initial Scenario

```
////////////////////////////////////////////////////////
// Dixie script
//
// ver 1.00
//      on Aug,9th,2009
//      by Hikaru Okada
//
// Copyright(C) 2009 Jazz Soft, Inc.
////////////////////////////////////////////////////////

int m_nInitScenario;

////////////////////////////////////////////////////////
// Selected Event
////////////////////////////////////////////////////////
```

4 New script was created.



2.17.8 Test Script

To test script, connecting with actual equipment would be the best. But also it is possible to setup 2 Dixie configurations; one for host simulatior, and one for equipment simulator.

7. Script Language Specification

Dixie script is an object-oriented language in order to control Dixie and communication.

2.18 Numbers

Number is 32-bit signed integer type. It is possible to use decimal expression, hexadecimal expression starting at “0x”. Also “true” (=1) and “false” are available.

Example

Dixie Script

```
    out.Stream = 1;
    out.Function = 0x0d;
    out.Wbit = true;
```

2.19 Strings

String is a collection of literal ASCII characters surrounded by double quotation marks. String can be concatenated.

Example

Dixie Script

```
    out.Sml =
        “s1f13w”
        “{“
        “      <b 0>”
        “      {“
        “      }”
        “}”
        ;
```

2.20 Comments

The comment syntax in Dixie is almost same as C++ language. User can use multi-byte characters such as Japanese in comment.

2.20.1 //

Characters from “//” (double slash) to the end of line would be treated as comment.

Dixie Script

```
    // HSMS connection has been selected
```

2.20.2 /* */

Characters from “/*” to “*/” would be treated as comment.

Dixie Script

```
/*
    HSMS connection has been selected
```

```
*/
```

2.21 Event Handler Functions

If specified condition was determined, Dixie would invoke event. It is possible to let Dixie take specified action by writing event handler function in script.

2.21.1 OnSelected()

This function will be called when HSMS select request was successful.

Syntax

Dixie Script

```
bool OnSelected( bEnabled )
```

The value of bEnabled is one of followings.。

Value	Description
true	Function is enabled.
false	Function is not enabled.
(empty)	Same as “true”.

If the function is disabled, script will not be executed.

Example

Dixie Script

```
bool OnSelected()
{
    cout << "OnSelected()";

    // S1F13
    out.Sml="s1f13w{<a'Script'><a’1.00’>}";
    Send();
    return true;
}
```

Remarks

See Also

2.21.2 OnReceived()

This function will be called when SECS-II data message arrived.

Syntax

Dixie Script

```
bool OnReceived( bEnabled )
```

The value of bEnabled is one of followings.。

Value	Description
true	Function is enabled.
false	Function is not enabled.
(empty)	Same as “true”.

If the function is disabled, script will not be executed.

Example

Dixie Script

```
bool OnReceived()
{
    cout << "OnReceived()";

    if(in.Stream==1)
        if(in.Function==1)
            cout << "stream is 1 and function is also 1";
        else
            cout << "stream is 1 but function is not 1";
    else
        if(in.Function==1)
            cout << "stream is not 1 but function is 1";
        else
            cout << "stream is not 1 and function is also not 1";
}
```

Remarks

See Also

2.21.3 OnProblem()

This function will be called when error occurred.

Syntax

Dixie Script

bool OnProblem(*bEnabled*)

The value of bEnabled is one of followings.。

Value	Description
true	Function is enabled.
false	Function is not enabled.
(empty)	Same as “true”.

If the function is disabled, script will not be executed.

Example

Dixie Script

```
bool OnProblem()
{
    cout << "OnProblem()";
    return false;
}
```

Remarks

See Also

2.22 Predefined Functions

There are predefined functions in Dixie script.

2.22.1 Send()

Send a message.

Syntax

```
Dixie Script

void Send( string strMessageName )
```

"strMessageName" is the name of preregistered message.　If abbreviated, the content of "out" object would be sent.

Example

```
Dixie Script
```

Remarks

See Also

2.22.2 Reply()

Reply a message.　Header information would be updated by referring "in" object.

Syntax

```
Dixie Script

void Reply( string strMessageName )
```

"strMessageName" is the name of preregistered message.　If abbreviated, the content of "out" object would be sent.

Example

```
Dixie Script
```

Remarks

See Also

2.23 Variable Types

There are 2 variable types; integer type and string type.　Types should match between left and right of operator in case of assignment or comparison.

2.24 Predefined Objects

Following objects are defined.

2.24.1 in

SecsII object which holds incoming message.

Integer type

Member	Description
BlockNumber	Block number.
DeviceID	Device ID.
Function	Function number.
NodeCount	Number of subitems in node.　Read-only.
NodeType	Node type.　Read-only.
Ptype	P type.
Stype	S type.
SessionID	Session ID.
SourceID	Source ID.
Stream	Stream number.
SystemBytes	System bytes.　Session ID and transaction ID.
TransactionID	Transaction ID.

Boolean type

Member	Description
Ebit	End bit.
Rbit	Reverse bit.
Wbit	Wait bit.

String type

Member	Description
Msg	Hexadecimal expression of message.
Node	Node descriptor.
NodeValue	The content of node.　Read-only.
NodeValueHex	The content of node in hexadecimal expression.　Read-only.
Sml	Message in SML expression.
Suggested	Suggested reply message in hexadecimal expression.　Read-only.
Xml	Message in XML expression.　Not supported at the moment.

2.24.2 out

SecsII object which holds outgoing message.　Member variables are same as “in” object.

2.24.3 arg

Argument object which will passed to event handler function.

Member

Member	Description
IPAddress	IP address.
PortNumber	Port number.
ErrorCode	Integer value of error code.　This member is available only on OnProblem() event.
AdditionalInfo	Additional information.　This member is available only on limited error code on OnProblem() event.

Example

Dixie Script

```
    cout << "IP Address:          " + arg.IPAddress;
    cout << "Port Number:         " + arg.PortNumber;
    cout << "Error Code:          " + arg.ErrorCode;
    cout << "Additional Info: " + arg.AdditionalInfo;
```

2.24.4 cout

Display text string in output window.

Syntax

Dixie Script

void cout::operator =(*string strText*)

Example

Dixie Script

```
    cout << "This is a sample.";
```

2.25 User Defined Variables

User can define variables.　Variable name should start at “m_”.

2.25.1 Int type

32-bit signed integer type.

Syntax

Dixie Script

int *VariableName* ;

Example

```
Dixie Script

int m_nTest;
```

2.25.2 string type

ASCII string type.

Syntax

```
Dixie Script

string VariableName ;
```

Example

```
Dixie Script

string m_strTest;
```

2.26 Statements

Statement is a set of order to control script.

2.26.1 if

“if” statement starts at “if” followed by condition expression, and then one or more statements, same as C/C++ language. User can write “else” block for not matched condition.

Syntax

```
Dixie Script

if ( Condition )
    Statement
```

```
Dixie Script

if ( Condition )
    Statement
else
    Statement
```

```
Dixie Script

if ( Condition )
{
    Statements
}
```

```
Dixie Script

if ( Condition )
{
    Statements
}
else
{
    Statements
}
```

Example

```
Dixie Script

    if(in.Stream==1)
    {
        // Matched condition
    }

    if(in.Stream==1)
    {
        // Matched condition
    }
    else
    {
        // Not matched condition
    }
```

If statement is only one, "{" "}" can be abbreviated for statement block.

Example

```
Dixie Script

    if(in.Stream==1)
        if(in.Function==1)
            Reply("s1f2");
```

"else" will be associated with nearest "if" statement.

Example

```
Dixie Script

bool OnReceived()
{
    cout << "OnReceived()";

    if(in.Stream==1) ---------------------------------------------- (A)
        if(in.Function==1) ---------------------------------------- (B)

        else ------------------------------------------------------ (1)

    else ---------------------------------------------------------- (2)
        if(in.Function==1) ---------------------------------------- (C)

        else ------------------------------------------------------ (3)

}
```

3 "else" expressions appear in above example. (1) is associated with (B), (2) is (A), and (3) is (C) respectively.

2.26.2 return

"return" statement will exit event handler function and pass processing back to Dixie. Either "true" or "false" will be returned to Dixie.

Syntax

Dixie Script

```
return ReturnValue ;
```

Example

Dixie Script

```
    return true;
    return false;
```

2.26.3 Assignment

Assignment statement assigns value into member variable of object, or user-defined variable. Use "=" mark. The type of left side of assignment operator and right side of assignment operator should be the same.

Syntax

Dixie Script

```
Receptor = Nominator ;
```

Example

Dixie Script

```
    out.Stream = 3;
    out.Sml = "s1f1w";
```

2.26.4 Others

It is possible to write predefined functions and object in statement.

Example

Dixie Script

```
    Reply("s1f2");
    cout << "This is a sample.";
```

2.27 Operators

Operators are basically left associative. Followings are defined. Precedence is from lower to higher in the list.

Operator	Description
=	Assignment
&&	AND condition

||	OR condition
==	Equal condition
!=	Not equal condition
>	Grater than
>=	Grater than or equal to
<	Less than
<=	Less than or equal to
+	Add
-	Subtract
*	Multiply
/	Devide
%	Modulus
&	Logical AND
|	Logical OR
-	Minus sign

Processing and association will be determined by operator precedence.

Example

Dixie Script

```
out.Function = 1 + 2 * 3 + 4;
```

Above statement contains 3 operators; “=”, “+” and “*”, but implicitly. Following is explicitly rewrote using “(“ and “)”.

Example

Dixie Script

```
out.Function = ( 1 + ( 2 * 3 ) + 4 );
```