World Programming
An ALTAIR Company

Altair SLC Link
user guide

Version: 2023.6

Contents

Introduction

Altair SLC Link is the collective term for the technology used to provide a client/server facility. Using Altair Analytics Workbench on your local PC (the client), you can connect to a remote Altair SLC server to run SAS language programs. The resulting output - datasets, logs, and so on - can then all be viewed and manipulated inside Altair Analytics Workbench (through the **Output Explorer**, **Results Explorer**, and **Server Explorer** views), just as if the work had been executed locally.

Altair Analytics Workbench can be installed on a completely different platform from the linked server platform. For example, you might have local Workbench installations on Windows workstations linked to multiple Linux servers. Altair SLC Link can also be used from Altair Analytics Workbench on one remote server to link to a further remote server.

Programs written in the SAS language language require no special language statements to use Altair SLC Link, although paths may need to be edited to reflect data locations on the remote machines.

Benefits

The benefits of Altair SLC Link include:

- Low-powered desktops and laptops can make transparent use of Altair SLC installations on more powerful servers.
- Workstations can take advantage of versions of Altair SLC installed on cloud, grid or cluster facilities.
- Altair Analytics Workbench users can run multiple concurrent programs on multiple local and remote Altair SLC servers.
- Security is enhanced as programs can be developed and run without data leaving the data centre.
- Analytics can be performed on data that has been generated elsewhere, using the Workbench's feature-rich local user interface.
- Organisations with distributed facilities can share the resources of remote servers set up in Altair Analytics Workbench.

Dependencies and usage

Altair SLC Link operates over an SSH (Secure Shell) connection and provides connectivity between installations of Altair Analytics Workbench and remote installations of Altair SLC on all common server platforms except z/OS.

Setup and Configuration

There are 3 items required for a Altair SLC Link client/server setup: an SSH (Secure Shell) server, the Altair SLC software itself, and sufficient license keys to cover the setup.

SSH server

Altair SLC Link requires the use of an SSH connection between the server and client machines. This is not supplied with Altair SLC. However:

- For a UNIX (Linux or AIX) server machine, the built-in SSH daemon can be used.
- For a Windows server machine, the third party SSH facility is available separately from Bitvise (refer to the *System Administration Guide* (page 15) for details).

Altair SLC software

Altair SLC is supplied as a single installation file that contains all the features required for use with Altair SLC Link, including Altair Analytics Workbench, **Java Runtime Environment**, and the licensable Altair SLC component.

Note:
You will need installation files that are suitable for the operating systems on both server and client machines. For example, if you have a Linux server machine and Windows client machines, you will need the Linux installation file for the server, and the Windows installation files for the clients. Refer to the relevant platform installation guide for full details of the Altair SLC installation process.

Licence key

In order for Altair SLC to be able to execute a SAS language program, the Altair SLC server component needs to be activated by the application of a license key.

- If you want to restrict the execution of programs to the linked server, then you only need a license for the server installation of Altair SLC.
- If you want to be able to execute programs on either the client machine or the linked server, then you need separate licence keys for the server and the client installations.

Client User Guide

Link Explorer and Workflow Link Explorer

The **Link Explorer** in the SAS Language Environment allows you to manage connections to Altair SLC hosts and any **Altair SLC Server**s on those connections. Similarly, the **Workflow Link Explorer** in the Workflow Environment allows you to manage connections to Altair SLC hosts and any **Engine**s on those connections.

The **Link Explorer** and **Workflow Link Explorer** views are the key views if you are using *Link* – the collective term for the technology used to provide a client/server architecture. These views allow you to connect to a remote machine with an **Altair SLC Server** installed to run SAS language programs or Workflows. The output from the remote server can be viewed and manipulated in Altair Analytics Workbench, in the same manner as a locally executed program or Workflow.

Displaying the Link Explorer view or Workflow Link Explorer view

To display the **Link Explorer** or **Workflow Link Explorer**: from the **Window** menu click **Show View** and then click **Link Explorer** ( ) when in the SAS Language Environment perspective, or **Workflow Link Explorer** ( ) when in the Workflow Environment perspective.

Objects displayed

There are two node types visible in this view:

1. A *Connection node* ( ): this is the root node and represents a connection to a host machine. The connection can either be a local connection or a connection to a remote machine. The connection can contain one or more **Altair SLC Server** nodes. There can only be one local connection, but many remote connections.
2. A *Server node* ( ), representing the Processing Engine installation where you can run your SAS language programs or Workflows.

Managing the connections and servers

If you right-click on an open connection, the following options are available:

- Open or close a local or remote host connection.
- Export a host connection (see *Exporting a host connection definition* (page 9)).

- Define a new **Altair SLC Server** (see *Defining a new Processing Engine* (page 10)).
- Import or export an **Altair SLC Server** definition (see *Importing an Altair SLC Server definition* (page 11) or *Exporting an Altair SLC Server definition* (page 11)).

Right-clicking on a server gives you options to stop, start or restart that server.

A server can be moved or copied to a different host connection. A drag-and-drop operation moves the server from its current host connection to the target host connection. Press and hold the **Ctrl** key at the same time to copy the server to the new host connection.

Properties

Right-clicking a connection node and selecting **Properties** will open a **Properties** window which has two tabs:**Directory Shortcuts** and **Workflow properties**.

- The **Directory Shortcuts** tab allows you to enter shortcuts for the host's filing system, which appear in the **File Explorer** window.
- The **Workflow properties** tab contain the following options:
  - **Temporary file path** - Enables you to enter a path for temporary files created by Workflows
  - **Limit working dataset storage** - Limits the storage space for working datasets created by Workflows.

Commands (top right corner of the view)

The view contains the following commands

- **Collapse All** – collapse the view to show just the root nodes.
- **Create a new remote host connection** – add a new remote connection.
- **Import a host connection definition** –import a connection definition file. Host connections can be also imported by dragging Connection Definition Files (with the suffix `*.cdx`) onto the **Link Explorer**.

Connecting to a remote Processing Engine

How to create a link from Altair Analytics Workbench to an **Altair SLC Server** on a remote machine.

The **Altair SLC Server** must exist before a connection can be made from Altair Analytics Workbench. The remote server requires a licensed copy of the **Altair SLC Server**. If the remote server is Windows, it requires an SSH server; this is not necessary for Linux due to native SSH support.

Client/server installation summary

Before connecting to a remote server, you require the hostname for the remote server, the user name and password log on credentials, and the full **Altair SLC Server** installation path on the remote server.

To create a connection to a remote server:

1. Create a connection to the remote host. See *Creating a new remote host connection* (page 7).
2. Add an **Altair SLC Server** to this remote host. See *Defining a new Processing Engine* (page 10).

Creating a new remote host connection

A remote host connection enables you to access a remote host's file system using the **File Explorer** view and link to a remote Processing Engine to run Workflows or SAS language programs.

New remote hosts will display in both **Link Explorer** view and **Workflow Link Explorer**, but links to **Altair SLC Server**s can only be viewed in the SAS Language Environment perspective, and Workflow Engines in the Workflow Environment.

Before creating a new connection, you will need:

- SSH access to the server machine that has a licensed **Altair SLC Server** installation.
- The installation directory path for the **Altair SLC Server** installation.

You may need to contact the administrator to obtain this information, and to ensure that you have access. For more details about SSH authentication, see the *Altair SLC Link user guide and reference*.

To create a new remote host connection:

1. Click the **Altair** menu, click **Link** and then click **New Remote Host Connection**. The **New Server Connection** dialog is displayed.
2. Select **New SSH Connection (3.2 and later – UNIX, MacOS, Windows)** and click **Next**.

3. Complete the **New Remote Host Connection (3.2 and later)** dialog box as follows:

   a. In **Hostname**, enter the domain name or IP address of the remote server machine.

   b. Unless modified by your system administrator, the default **Port** value (*22*) should be left unchanged.

   c. In **Connection name** enter a unique name to be displayed in Altair Analytics Workbench for this connection. By default this entry will be copied from your entry in the **Hostname** box.

   d. In **User name**, enter your user ID on the remote host.

   Click the required check box:

Option	Description
Enable compression	Controls whether or not data sent between the Altair Analytics Workbench and the remote connection is compressed.
Verify Hostname	Confirms whether the host you have specified in the **Hostname** entry exists.
Open the connection now	Whether the connection is automatically opened immediately
Open the connection automatically on Workbench startup	Controls whether the connection is automatically opened when the Altair Analytics Workbench is started.

4. Click **Next** to define the connection directory shortcuts (if required):

   a. Click **Add** to display the **Directory Shortcut** dialog box.

   b. In Directory Name enter the displayed shortcut name, and in **Directory Path** the full path of the target directory. Click **OK** to save the changes.

   c. Enter the displayed shortcut name in **Directory Name**, and the full path in **Directory Path** and click **OK**.

5. Click **Finish** to save the changes.

   If you have not previously validated the authenticity of the remote host, an **SSH2 Message** dialog box is displayed requesting confirmation of the new connection.

6. In the **Password Required** dialog enter your **password** for the remote machine and click **OK**.

The **Link Explorer** view contains an entry for the new connection. To run a Workflow or SAS language program, you will now need to define a new processing engine (see *Defining a new Processing Engine* (page 10)).

Advanced connection options

Controlling the environment variables available to a remote Processing Engine.

By default, the Processing Engine process inherits the interactive shell environment for the user under whose ID the server is being started. The default environment may not include everything required by the Processing Engine, for example database client shared libraries may not be defined for the server.

On Unix/Linux systems, this might require that any special environment variables for the Processing Engine are added to the *LD_LIBRARY_PATH*; or the `/etc/profile` or `~/.profile` modified so that database client shared libraries can be loaded by the Processing Engine.

An alternative on Unix/Linux is to place a `wpsenv.sh` file in the Altair SLC installation directory, or in your home directory. This file is automatically run before the server is launched

On Windows, any special environment variables for the **Altair SLC Server** should be configured through the **Control Panel**, as either system or user environment variables.

Note:
All processes running as the user will see the created variables, and there is no way to specify that they should only be visible to the Altair SLC process.

Exporting a host connection definition

A host connection definition can be exported from Altair Analytics Workbench to enable sharing in a workgroup or to copy connection definitions between workspaces.

To export host connection definitions:

1. In the **Link Explorer**, select the connection for which definitions will be exported.
2. Right-click the connection and click **Export Host Connection** in the shortcut menu.
3. In the **Export to File** dialog box, navigate to the save location, enter a **File name** and click **Save**. The exported file `.cdx` contains the connection definition in XML format.

The host connection definition is saved to the selected file, which can be shared in the workgroup.

You can select multiple connections in the **Link Explorer** view and export them all to the same definition file to share more than one connection definition.

See *Importing a host connection definition* (page 10) for how to import the connection definitions into a workspace.

Importing a host connection definition

A host connection definition can be imported into Altair Analytics Workbench to use consistent host definitions in a workgroup or between workspaces.

This task assumes that you have previously exported some connection definitions to a file, or have been provided with an export file created by someone else in your work group.

1. Click the **Altair** menu, click **Link**, and then click **Import Remote Host Connection**.
2. In the **Import Host Connection** dialog box, select the required connection definition file (`.cdx`) and click **Open**.

The host connection definition is imported from the selected file.

If there are any name clashes, the imported connection definition is automatically renamed so that it has a unique name within your list of connections.

Defining a new Processing Engine

In the SAS Language Environment, Altair Analytics Workbench allows you to manually create multiple **Altair SLC Server**s on a local or remote host connection. In the Workflow Environment, only one Engine can be created for each host.

New Processing Engines can only be defined for an active host connection.

To define a new Processing Engine:

1. In Workflow Link Explorer or **Link Explorer** view, right-click on the host you wish to create an engine on, and select **New Altair SLC Server** (SAS Language Environment), or **New Engine** (Workflow Environment).
2. In **New Altair SLC Server** or **New Remote Engine** dialog box, enter a unique display name in **Server name** or **Engine Name** (or accept the default suggestion).

   If you are defining a new remote **Altair SLC Server**, then in the **Base** Altair SLC **install directory** box enter the path to the Altair SLC base installation directory on the remote server.
3. Click **Finish** save the changes.

The **Link Explorer** view contains an entry for the new **Altair SLC Server**.

Exporting an Altair SLC Server definition

In the SAS Language Environment, an **Altair SLC Server** definition can be exported from Altair Analytics Workbench to enable sharing in a workgroup or to copy connection definitions between workspaces.

To export an **Altair SLC Server** definition:

1. In the **Link Explorer** view, select the server for which the definition will be exported.
2. Right-click the server and click **Export Host Connection** in the shortcut menu.
3. In the **Export to File** dialog box, navigate to the save location, enter a **File name** and click **Save**. The exported file `.sdx` contains the connection definition in XML format.

The **Altair SLC Server** definitions will be saved to the selected file, which can be shared in the workgroup.

You can select multiple servers in the **Link Explorer** view and export them all to the same definition file to share more than one connection definition.

See *Importing an Altair SLC Server definition* (page 11) for how to import the server definitions into a workspace .

Importing an Altair SLC Server definition

In the SAS Language Environment, an **Altair SLC Server** definition can be imported into Altair Analytics Workbench to use consistent definitions in a workgroup or between workspaces.

An **Altair SLC Server** definition can only be imported into a host connection. If you need to create a host definition, see *Creating a new remote host connection* (page 7).

To import an **Altair SLC Server** definition

1. Click the **Altair** menu, click **Link**, and then click **Import Remote Host Connection**.
2. In the **Import from File** dialog box, select the required server definition file (`.sdx`) and click **Open**.

The **Altair SLC Server** definitions are imported from the selected file.

If there are any name clashes, the imported server definition is automatically renamed so that it has a unique name within your list of **Altair SLC Server**s.

Connecting to an LSF cluster

How to create a link from Altair Analytics Workbench to an **Altair SLC Server** on an IBM Spectrum LSF cluster.

The LSF cluster must exist before a connection can be made from the Altair Analytics Workbench on your local machine. The LSF cluster requires a licensed copy of the **Altair SLC Server** and an installation of Altair SLC on each execution host.

Client/server installation summary

Before connecting to an LSF cluster, you require the hostname for a submission host, the user name and password log on credentials, and the full **Altair SLC Server** installation path on the submission host.

To create a connection to an LSF cluster:

1. Create a connection to the remote host. See *Creating a new connection to an LSF cluster* (page 12).
2. Add an **Altair SLC Server** to this remote host. See *Defining a new Processing Engine on an LSF connection* (page 14).

Creating a new connection to an LSF cluster

A connection to an LSF cluster enables you to access the file system of the machine you have connected to using the **File Explorer** view and link to a remote Processing Engine to run Workflows or SAS language programs.

New LSF connections are displayed in both the **Link Explorer** view and the **Workflow Link Explorer**. For more information about the connection node see *Creating a new remote host connection* (page 7).

Before creating a new connection, you will need:

- Access to the cluster with a licensed **Altair SLC Server** installation.
- The installation directory path for the **Altair SLC Server** installation.

You may need to contact the administrator to obtain this information, and to ensure that you have access. For more details about SSH authentication, see the *Altair SLC Link user guide and reference*.

To create a new remote host connection:

1. Click the **Altair** menu, click **Link** and then click **New Remote Host Connection**.

   A **New Server Connection** window is displayed.
2. Select **New IBM Spectrum LSF Connection (4.3 and later)** and click **Next**.

3. Complete the **New IBM Spectrum LSF Connection** dialog box as follows:
    a. In **LSF Submission host**, enter the domain name or IP address of a submission host in the LSF cluster.
    b. Unless modified by your system administrator, the default **Port** value (*22*) should be left unchanged.
    c. In **Connection name**, enter a unique name to be displayed in Altair Analytics Workbench for this connection. By default this entry will be copied from your entry in the **Hostname** box.
    d. In **User name**, enter your user ID for the remote host.

    Click the required check box:

Option	Description
Open the connection now	Specifies whether the connection is automatically opened immediately.
Open the connection automatically on Workbench startup	Controls whether the connection is automatically opened when the Altair Analytics Workbench is started.
Verify Hostname	Confirms whether the host you have specified in the **Hostname** entry exists.

4. If required, click **Next** to define connection directory shortcuts:
    a. Click **Add** to display the **Directory Shortcut** dialog box.
    b. In **Directory Name**, enter the displayed shortcut name. In **Directory Path** enter the full path of the target directory. Click **OK** to save the changes.
5. Click **Finish** to save the changes.
6. In the **Password Required** dialog, enter your **password** for the remote cluster and click **OK**.
7. To runWorkflows on the LSF cluster, you need to specify a temporary file path for the connection:
    a. Right-click the new connection and select **Connection**, then select **Properties** and **Workflow Properties**.
    b. Under **Temporary File Path** enter a path to store temporary files, this must be a path that every machine in the LSF cluster has access to.
    c. Click **Finish** to save the changes.

The **Link Explorer** view contains an entry for the new connection. To run a SAS language program or Workflow, you will need to define a new processing engine (see *Defining a new Processing Engine on an LSF connection* (page 14)).

Defining a new Processing Engine on an LSF connection

In the SAS Language Environment, you can create multiple **Altair SLC Server**s on your LSF connection. In the Workflow Environment, only one Engine can be created for each LSF connection.

New Processing Engines can only be defined for an active host connection.

To define a new Processing Engine:

1. In **Link Explorer** view or **Workflow Link Explorer**, right-click on the host you wish to create an engine on, and select **New Altair SLC Server** (SAS Language Environment), or **New Engine** (Workflow Environment).
2. In New **Altair SLC Server** or **New Remote Engine** dialog box, enter a unique display name in **Server name** or **Engine Name** (or accept the default suggestion).

   If you are defining a new remote **Altair SLC Server**, then in the Base Altair SLC install directory box enter the path to the Altair SLC base installation directory on the remote server.

   Note:

   The base Altair SLC install directory must exist on each execution host in the LSF cluster
3. Click **Finish** save the changes.

The **Link Explorer** view contains an entry for the new **Altair SLC Server**.

System Administration Guide

This part of the guide is intended for system administrators who are responsible for the configuration of Altair SLC Communicate and Altair SLC Link, and, more specifically, for server authentication and the generation, deployment and verification of any required public and private keys.

Note:

Altair SLC Communicate and Altair SLC Link are separate features that can run independently. That is to say, you do not need one in order to run the other. You would use Altair SLC Communicate to run selective code on remote server hosts or a z/OS mainframe, and Altair SLC Link to run entire programs, via the Workbench GUI, on remote server hosts only. The two features are described together here as they both require means of remote authentication.

The following is a summary of the authentication methods as they apply to both Altair SLC Communicate and Altair SLC Link.

Authentication Method	Altair SLC Communicate	Altair SLC Link
Password	Yes	Yes
Public key with passphrase and keychain agent	Yes	Yes
Public key with passphrase and no keychain agent	No	Yes
Kerberos	Yes	Yes
Telnet on z/OS	Yes	No

SSH (Secure Shell) from a Windows client

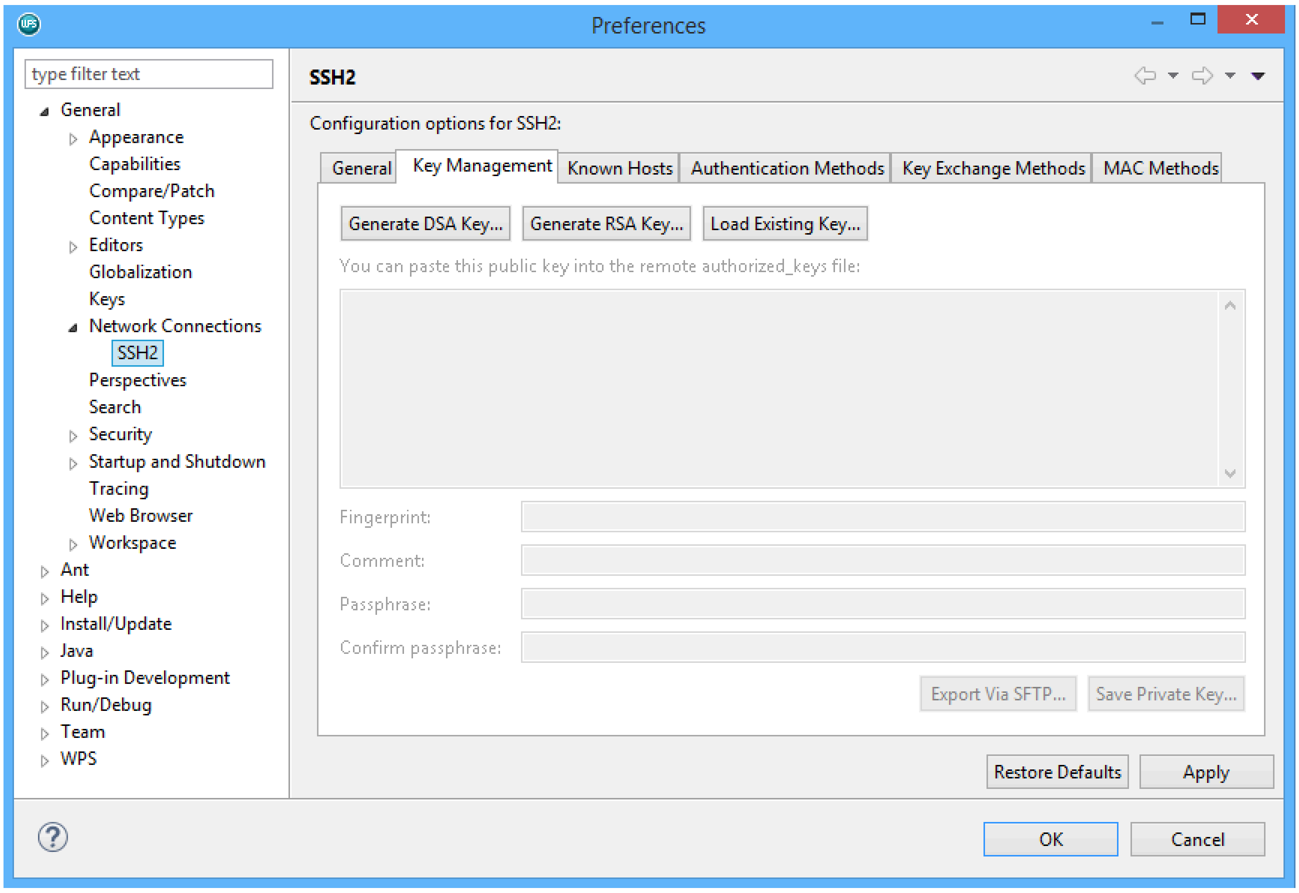
This section covers the use of using an SSH with both Altair SLC Communicate and Altair SLC Link to create and maintain con nections between server and client machines.

Before you access a remote host via Altair SLC Communicate or Altair SLC Link, it is recommended that you test the connection to the remote host with an external SSH client, for example, Bitvise.

If you would like to use *Public key authentication* (page 22), and keys have not already been generated on the server, then you may also wish to download PuTTYgen (refer to *Key generation using PuTTYgen* (page 23)). If you intend to use public keys with a **passphrase**, and you are using Altair SLC Communicate, you will also need to download a keychain agent such as Pageant (refer to *Passphrase authentication using Pageant* (page 37)). The use of such an agent for **passphrases** is not necessary with Altair SLC Link, although it may be desirable if you are connecting to multiple servers.

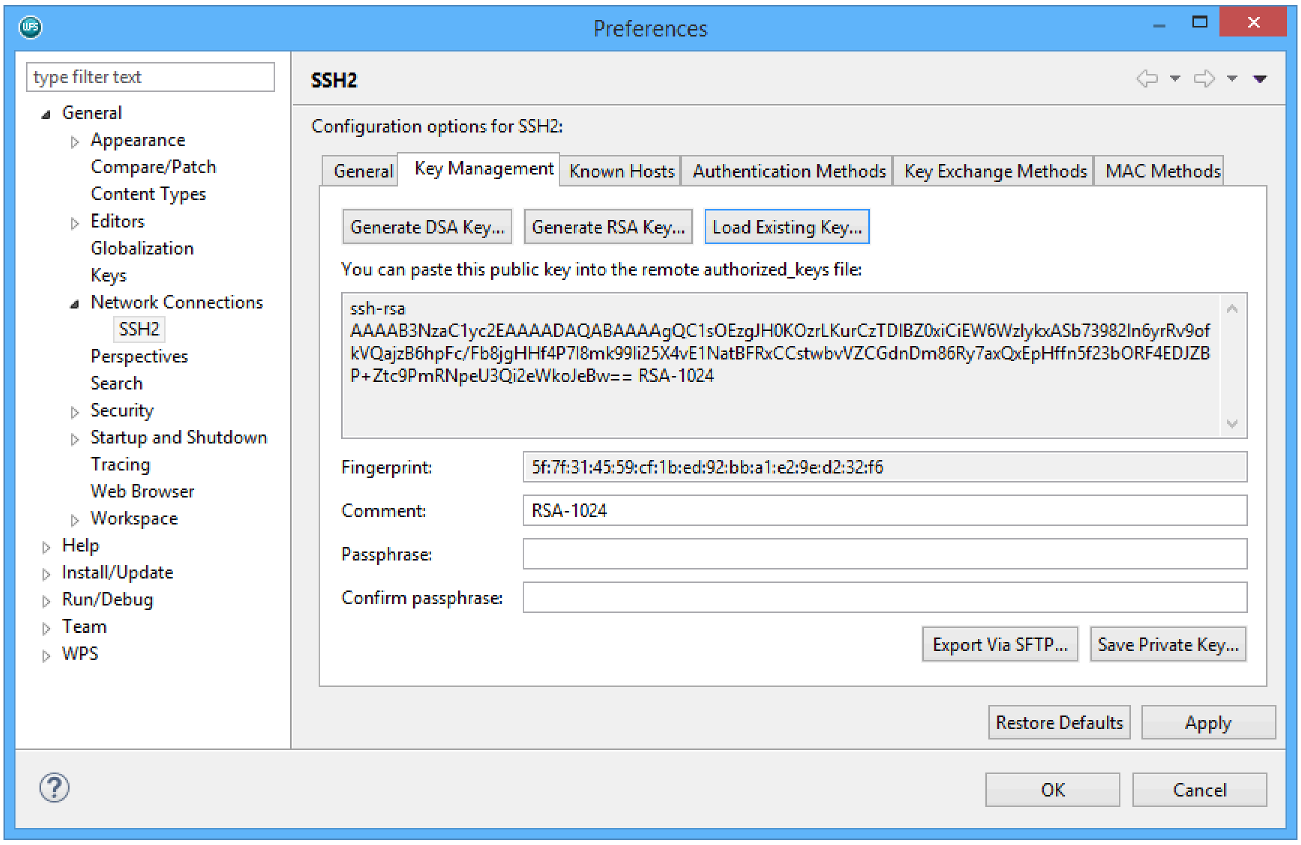
If you are using Altair SLC Link and already have a private key, and wish to apply it, then proceed as follows:

1. On the Altair Analytics Workbench main menu, select **Window** ➤ **Preferences** and, in the left-hand pane of the subsequent **Preferences** dialog, expand the **General** ➤ **Network Connections** ➤ **SSH2** nodes.
2. Select the **Key Management** tab of the **Preferences** dialog:



3. Click **Load Existing Key...**.

4. Browse to the required private key and select it, to display the screen shown in the following example:



Configuration options for SSH2:

You can paste this public key into the remote authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAAAgQC1sOEzgJH0KOzrLKurCzTDIBZ0xiCiEW6WzlykxASb73982In6yrRv9of
kVQajzB6hpFc/Fb8jgHHf4P7l8mk99li25X4vE1NatBFRxCCstwbvVZCGdnDm86Ry7axQxEpHffn5f23bORF4EDJZB
P+Ztc9PmRNpeU3Qi2eWkoJeBw== RSA-1024
```

Fingerprint: 5f:7f:31:45:59:cf:1b:ed:92:bb:a1:e2:9e:d2:32:f6

Comment: RSA-1024

Passphrase:

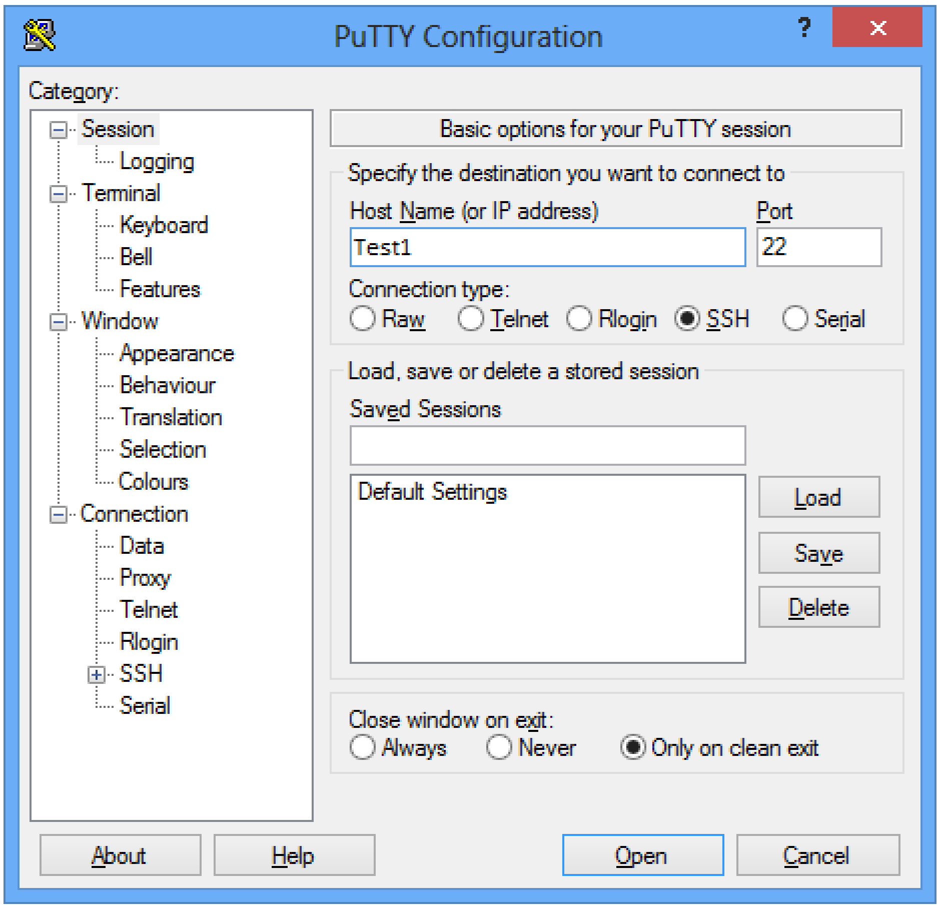
Confirm passphrase:

5. If you wish to apply a passphrase to your private key file, complete the **Passphrase** and **Confirm passphrase** fields.
6. Click **OK** to save your changes and dismiss the **Preferences** window.

Password authentication using PuTTY

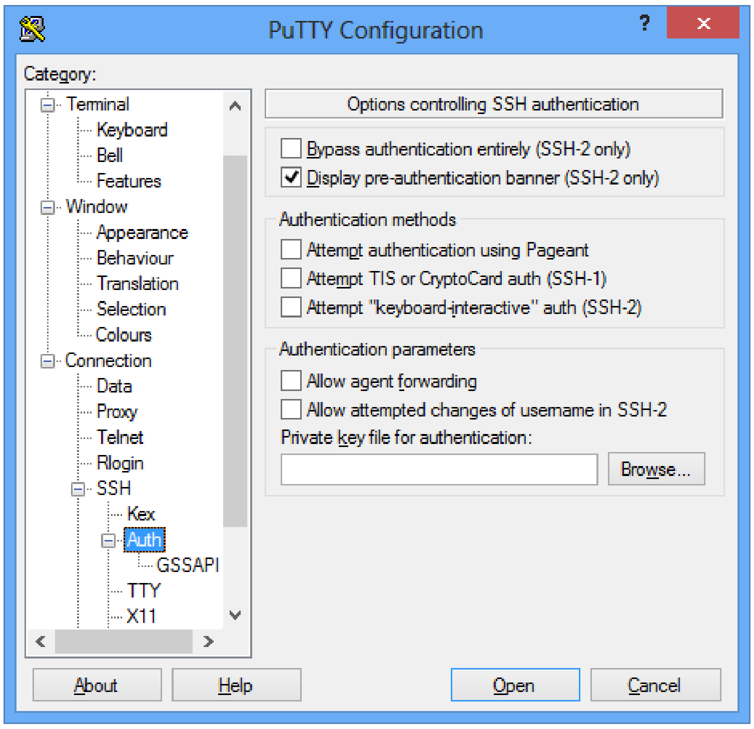
Manual SSH sign-on provides the opportunity to perform host key validation. For increased security, Altair SLC performs host key validation during SSH sign-on. To do this relies upon host key acceptance having already been performed by an external SSH client. Altair SLC validates the host key it receives against the same database as is used by the external SSH client. On Windows clients, the default is to use the PuTTY host key database stored in the Windows registry, so it is necessary to log onto the remote host using the PuTTY SSH client to validate the host key and add it to the host key database before attempting to make a connection.

1. Launch the PuTTY client and enter the host name in the main **Host Name (or IP address)** entry field:



Host Name (or IP address): Test1
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]

2. Expand the **SSH ➤ Auth** configuration page from the category list on the left, and ensure that nothing is selected under **Authentication methods** and that the **Private key file for authentication** field is empty:



Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

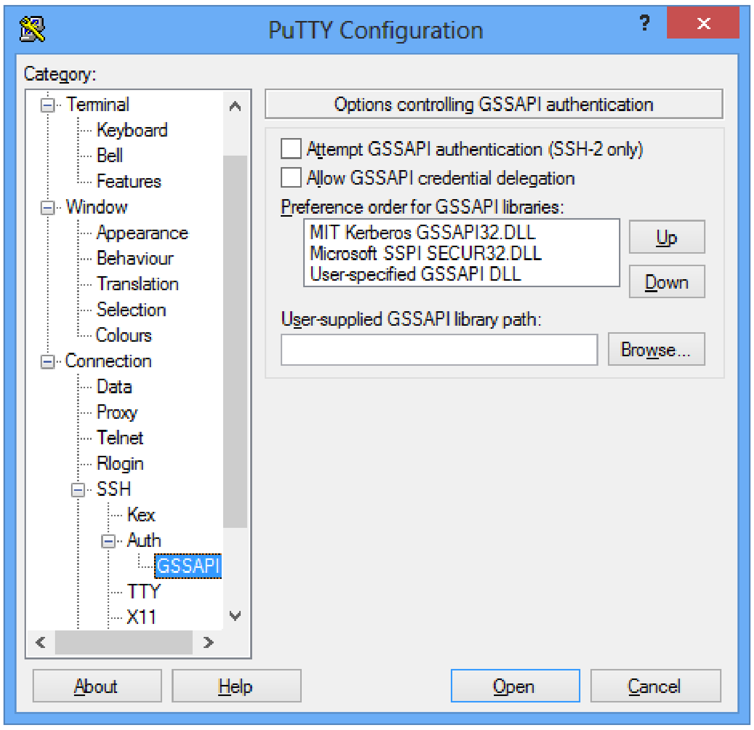
- [ ] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [ ] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:

3. Select the **GSSAPI** page and ensure that **Attempt GSSAPI authentication** is not selected:



Options controlling GSSAPI authentication

☐ Attempt GSSAPI authentication (SSH-2 only)
☐ Allow GSSAPI credential delegation

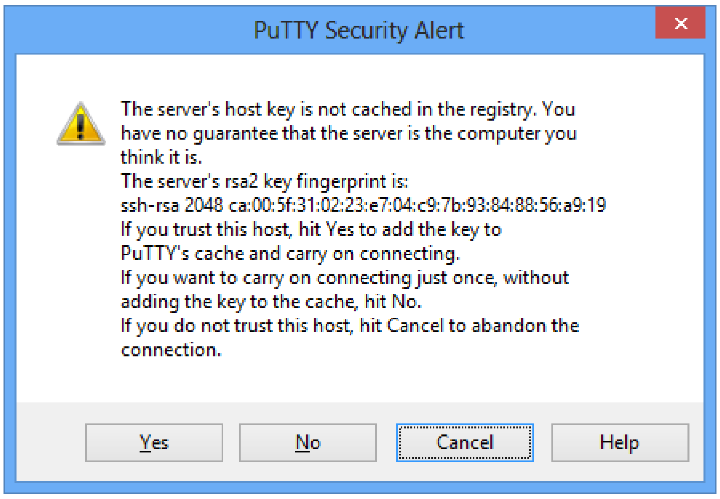
Preference order for GSSAPI libraries:
MIT Kerberos GSSAPI32.DLL
Microsoft SSPI SECUR32.DLL
User-specified GSSAPI DLL

User-supplied GSSAPI library path:

These checks ensure that only password authentication is being used, and that we have not inadvertently selected some more involved authentication mechanism.

4. Click on **Open** and when prompted, type in your password and press `Enter`.

   If this is the first time you have signed on to this particular host, an alert of the following kind will be displayed:


PuTTY Security Alert

The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 ca:00:5f:31:02:23:e7:04:c9:7b:93:84:88:56:a9:19
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

   At this point you should confirm that this is indeed the correct fingerprint for the host, and, assuming that it is, click **Yes** to accept the key permanently. If everything has worked, you will be logged in via a terminal session to your remote host. The host key will have been validated and stored in the Windows registry and is used to perform validation when you launch Altair SLC.

   Note:

   This host key validation is not public key authentication - they are two entirely separate things. Host key validation simply gives you an opportunity to confirm that the host to which you are connecting is, indeed, the host to which you intended to connect.

5. If you are using Altair SLC Link, create the required host connection and remote host server through Altair Analytics Workbench. If you are using Altair SLC Communicate, sign on via the `SIGNON` statement - you need to specify either the IDENTITYFILE statement option or the SSH_IDENTITYFILE system option, for example:

   ```
   SIGNON <servername> SSH
   USERNAME="<username>"
   password="<password>"
   LAUNCHCMD="/home/installs/wps/bin/wps -dmr ";

   RSUBMIT;
   %PUT &SYSHOSTNAME;
   ENDRSUBMIT;
   SIGNOFF;
   ```

   Alternatively:

   ```
   OPTIONS SSH_IDENTITYFILE="C:\Users\techwriter\.ssh\wpscommunicate.ppk";
   SIGNON <servername> SSH
   password="<password>"
   LAUNCHCMD="/home/installs/wps-3.2/bin/wps -dmr";

   RSUBMIT;
   %PUT &SYSHOSTNAME;
   ENDRSUBMIT;
   SIGNOFF;
   ```

   Note:

   You cannot use either **IDENTITYFILE** or **SSH_IDENTITYFILE** if you are using *Passphrase authentication using Pageant* (page 37).

Public key authentication

This method is more secure than using a simple password, and is sometimes called *password-less* authentication.

With SSH, authentication using keys not only improves security in general, but also, in the case of Altair SLC Communicate, it avoids having user names and passwords committed to source code (even in obfuscated or encrypted form).

This authentication method relies upon a cryptographic key-pair, where the private key resides on (and never leaves) the client machine, and the public key is installed on the SSH server to which the client needs to connect, or, in the case of Windows, on **Bitvise SSH Server**. The SSH protocol uses the key-pair to establish the identity of the client and perform the authentication.

Two methods are described by which the keys can be generated on a Windows client:

- *Key generation using PuTTYgen* (page 23)
- *Key generation using Altair Analytics Workbench* (page 26)

Following the generation of the public keys, they should be placed on the remote server in accordance with *Deploying public keys on the remote SSH server* (page 28), or, in the case of a Windows server, *Deploying public keys on Bitvise SSH Server* (page 30).

If you wish to connect to multiple servers, without having to remember or enter your password for each system, then you should also use *Passphrase authentication using Pageant* (page 37).

The validity of the key-pairs should then be checked in accordance with *Remote host access verification using PuTTY* (page 35).

Note:

Public key authentication can be used with both Altair SLC Communicate and Altair SLC Link. However, for Altair SLC Communicate, if you are not using a keychain agent such as Pageant, there cannot be a **passphrase** on the private key file, as there is currently no interactive mechanism to prompt for it during authentication.

Note:

You need to ensure that public key authentication is not disabled on the client machine.

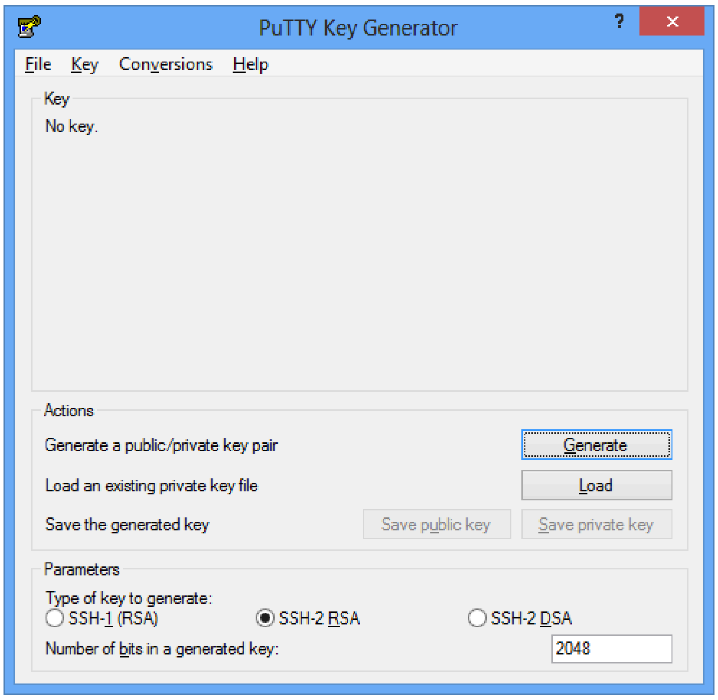
Key generation using PuTTYgen

To generate a key-pair, which is the combination of the private key and the public key for asymmetric encryption, proceed as follows:

Note:

You should be aware that, as an alternative, you can also use Altair Analytics Workbench to generate key-pairs (refer to *Key generation using Altair Analytics Workbench* (page 26)).

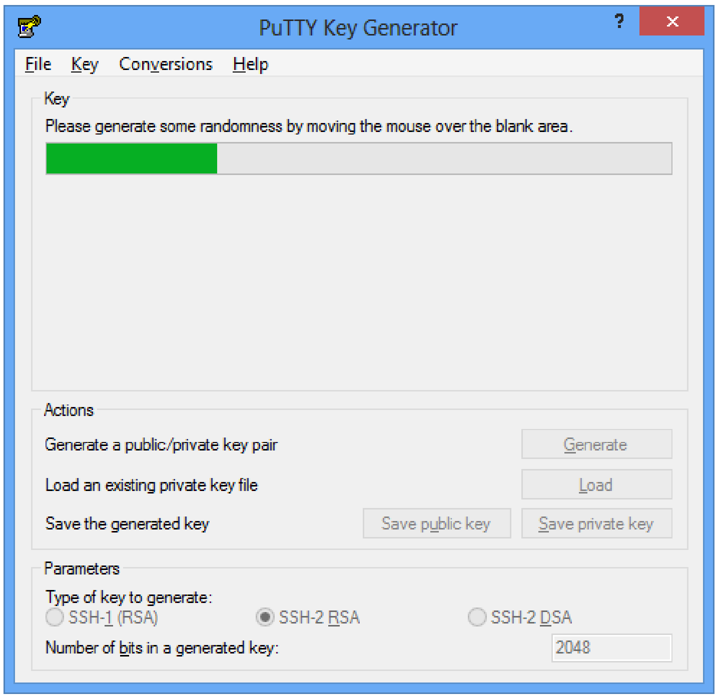
1. Launch the PuTTYgen tool (available via the same sources as PuTTY).



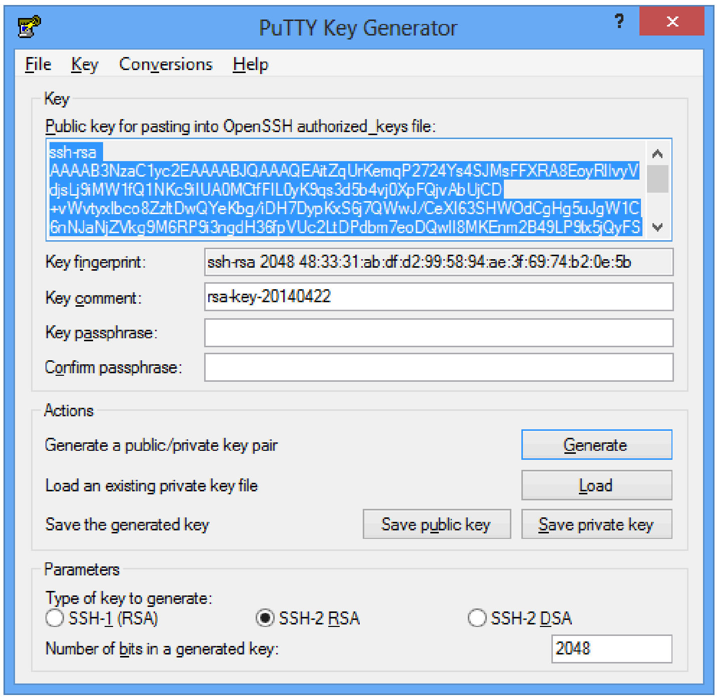
Important:

It is recommended that you use the parameter **SSH-2 RSA** with a minimum key length of **2048**.

2. Click the **Generate** button and move the mouse within the indicated area to generate some randomness - this will act as a seed for your key-pair:


PuTTY Key Generator
File Key Conversions Help
Key
Please generate some randomness by moving the mouse over the blank area.
Actions
Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key
Parameters
Type of key to generate:
SSH-1 (RSA)
SSH-2 RSA
SSH-2 DSA
Number of bits in a generated key:
2048

3. The system generates a key-pair:



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAitZqUrKemqP2724Ys4SJMsFFXRA8EoyRIlvyV
djsLj9iMW1fQ1NKc9iIUA0MCtfFIL0yK9qs3d5b4vj0XpFQjvAbUjCD
+vWvtyxlbco8ZzltDwQYeKbg/iDH7DypKxS6j7QWwJ/CeXI63SHWOdCgHg5uJgW1C
6nNJaNjZVkg9M6RP9i3ngdH36fpVUc2LtDPdbm7eoDQwlI8MKEnm2B49LP9tx5jQyFS
```

Key fingerprint: ssh-rsa 2048 48:33:31:ab:df:d2:99:58:94:ae:3f:69:74:b2:0e:5b

Key comment: rsa-key-20140422

Key passphrase:

Confirm passphrase:

Type of key to generate: SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 2048

4. If you wish to apply a passphrase to your private key file, complete the **Key passphrase** and **Confirm passphrase** fields. You will need to do this if you are going to be using *Passphrase authentication using Pageant* (page 37).

Note:

If you are using Altair SLC Communicate, do not enter a **passphrase** unless you are going to be using *Passphrase authentication using Pageant* (page 37). If you are using Altair SLC Link, you can enter a **passphrase** and use it either with or without *Passphrase authentication using Pageant* (page 37).

5. Click **Save private key**. If you did not enter a **passphrase**, you will be asked to confirm that you wish to save the key without a **passphrase**. The resultant file is in PuTTY's native format (`*.PPK`), and, when you are prompted to save the file to a folder, you should ensure that it is stored in the `.ssh` folder in your user profile.

Note:

Ensure that the permissions on your private key file are such that only you can read it. This file is essentially your password, so it is important that no-one else can access the file.

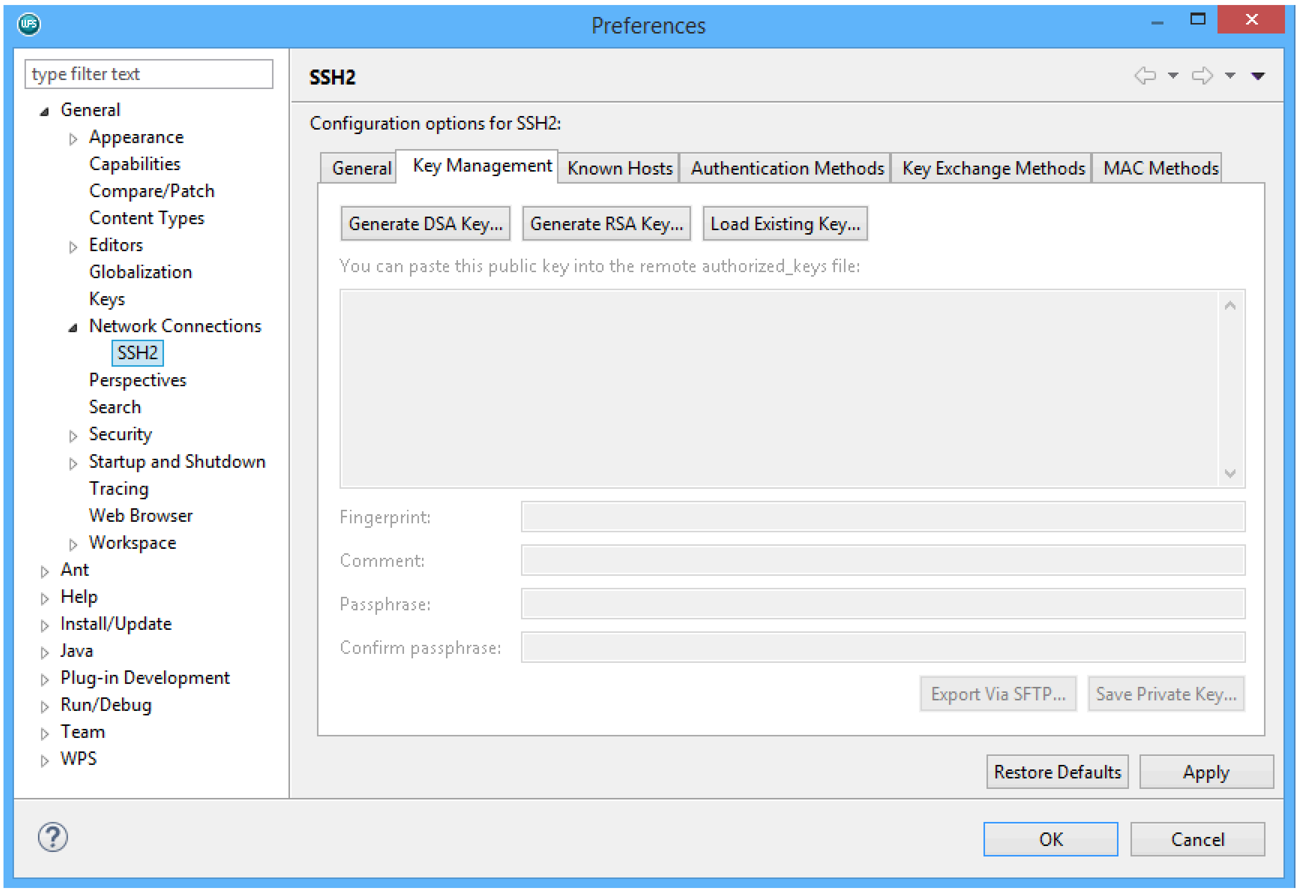
6. The public key that is highlighted in step *3* (page 25) contains the information needed to allow a user to verify that another party is in possession of the corresponding private key. The public key does not need to be kept secure, but you should save it by either copying it to your pasteboard and pasting it to a plain text file, or selecting **Save public key** in order to save it to the `.ssh` folder in your user profile. You should then proceed as in *Deploying public keys on the remote SSH server* (page 28) or *Deploying public keys on Bitvise SSH Server* (page 30).

Note:

If you are going to use Altair SLC Link in conjunction with Altair SLC Communicate, then, in order to avoid the need for two separate key-pairs, you should have a consistent strategy - that is to say, either avoid a **passphrase** in both cases, or else create a single **passphrase** and associate it with a single key-pair via *Passphrase authentication using Pageant* (page 37).

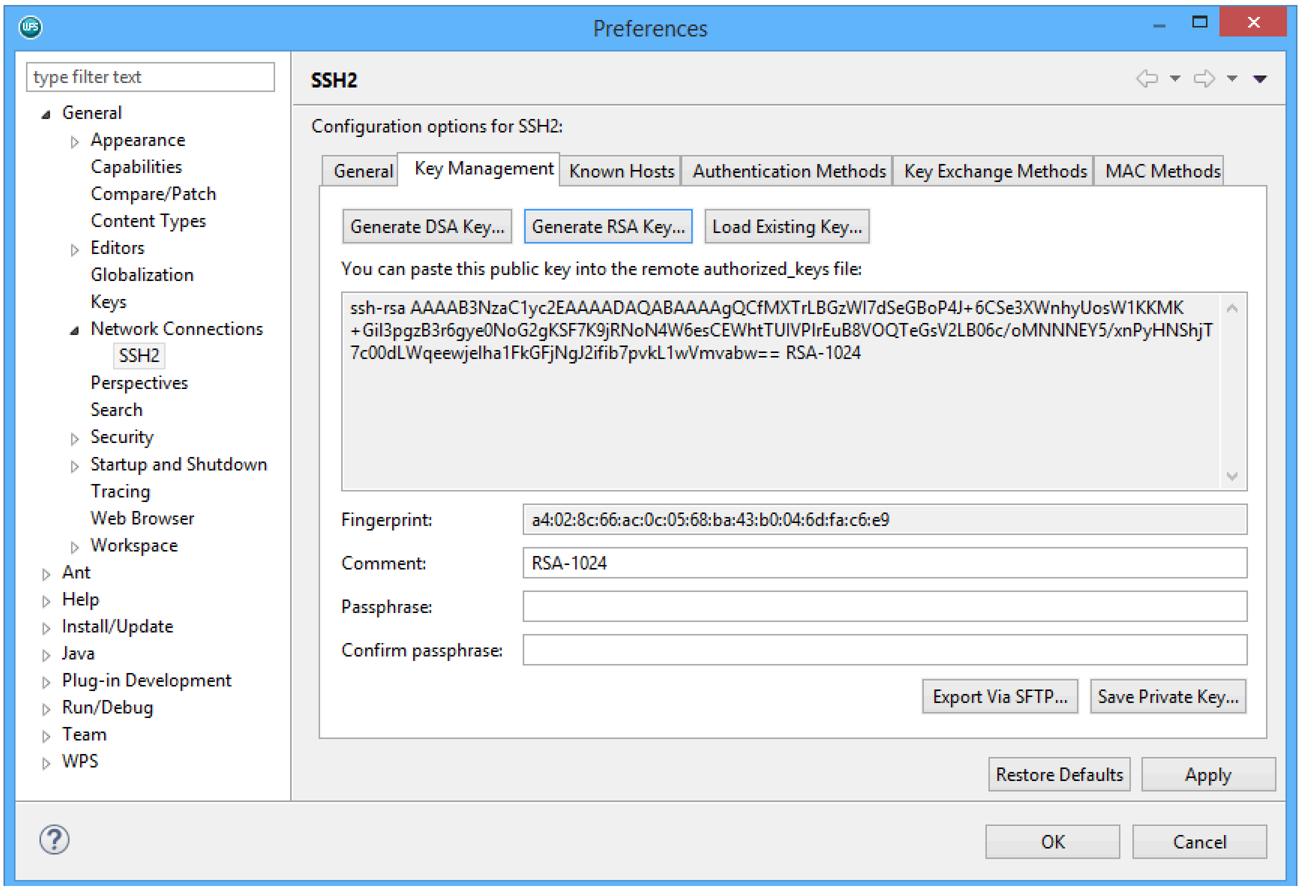
Key generation using Altair Analytics Workbench

1. On the Altair Analytics Workbench main menu, select **Window** ➤ **Preferences** and, in the left-hand pane of the subsequent **Preferences** dialog, expand the **General** ➤ **Network Connections** ➤ **SSH2** nodes.
2. Select the **Key Management** tab of the **Preferences** dialog:



3. Click **Generate RSA Key...**.

   A key pair is generated - the public key is displayed in a text box in the centre of the dialog, for example:



Configuration options for SSH2:

You can paste this public key into the remote authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQCfMXTrLBGzWl7dSeGBoP4J+6CSe3XWnhyUosW1KKMK
+Gil3pgzB3r6gye0NoG2gKSF7K9jRNoN4W6esCEWhtTUIVPIrEuB8VOQTeGsV2LB06c/oMNNNEY5/xnPyHNShjT
7c00dLWqeewjelha1FkGFjNgJ2ifib7pvkL1wVmvabw== RSA-1024
```

Fingerprint: a4:02:8c:66:ac:0c:05:68:ba:43:b0:04:6d:fa:c6:e9

Comment: RSA-1024

Passphrase:

Confirm passphrase:

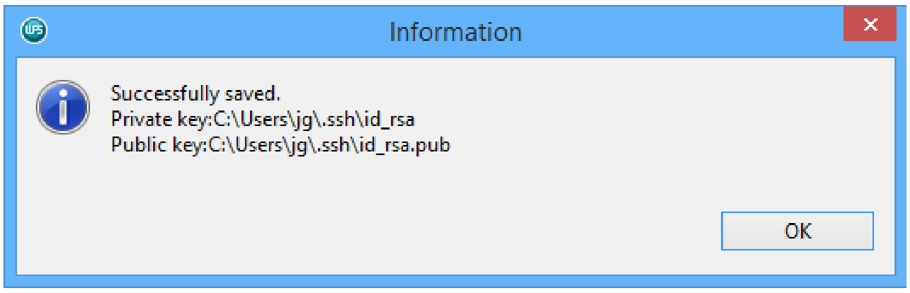
4. If you wish to apply a passphrase to your private key file, complete the **Passphrase** and **Confirm passphrase** fields. You will need to do this if you are going to be using *Passphrase authentication using Pageant* (page 37).

Note:

If you are using Altair SLC Communicate, do not enter a **passphrase** unless you are going to be using *Passphrase authentication using Pageant* (page 37). If you are using Altair SLC Link, you can enter a **passphrase** and use it either with or wihout *Passphrase authentication using Pageant* (page 37).

5. Click **Save Private Key**. If you did not enter a **passphrase**, you will be asked to confirm that you wish to save the key without a **passphrase**. When you are prompted to save the resultant key file to a folder, you should ensure that it is stored in the `.ssh` folder in your user profile. If you do not wish to use the default name of `id_rsa`, give the file a more meaningful name.

   Altair Analytics Workbench displays an information dialog confirming that it has saved your private key file, together with the corresponding public key file. It gives the public key file the same prefix as your private key file, but appends `.pub` to it, for example:

   ```
   Successfully saved.
   Private key:C:\Users\jg\.ssh\id_rsa
   Public key:C:\Users\jg\.ssh\id_rsa.pub
   ```

   Note:

   Ensure that the permissions on your private key file are such that only you can read it. This file is essentially your password, so it is important that no-one else can access the file.

6. Click **OK** to dismiss the **Information** dialog.
7. Click **OK** to save your changes and dismiss the **Preferences** window.
8. The public key that is both displayed on screen (for copying and pasting if required), and saved to a file, contains the information needed to allow a user to verify that another party is in possession of the corresponding private key. You should then proceed as in *Deploying public keys on the remote SSH server* (page 28) or *Deploying public keys on Bitvise SSH Server* (page 30).

   Note:

   If you are going to use Altair SLC Link in conjunction with Altair SLC Communicate, then, in order to avoid the need for two separate key-pairs, you should have a consistent strategy - that is to say, either avoid a **passphrase** in both cases, or else create a single **passphrase** and associate it with a single key-pair via *Passphrase authentication using Pageant* (page 37).

Deploying public keys on the remote SSH server

1. Log into the remote machine.

2. Once logged in, you must configure the server to accept your public key for authentication, so change into the `.ssh` directory and open the file `authorized_keys`.

   If this is the first public key to be put into the file, then you may need to create the directory and file first, by, for example, running the following commands:

   ```
   mkdir -p .ssh
   touch ~/.ssh/authorized_keys
   ```

3. Set the right permissions, for example:

   ```
   chmod 600 ~/.ssh/authorized_keys
   ```

   Note:

   You also need to ensure that your `$HOME` directory and `.ssh` directory have the permissions that are are appropriate to both the server and your particular operation.

4. Now you can add the public key to the `authorized_keys` file, as in the following example:

   ```
   cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
   ```

   If you currently have password-based SSH access configured to your server, and you have the `ssh-copy-id` utility installed, then you can simply transfer your public key by typing:

   ```
   ssh-copy-id username@remote_host
   ```

   You will then be prompted for the user account's password on the remote system. After typing in the password, the contents of your `~/.ssh/id_rsa.pub` key will be appended to the end of the user account's `~/.ssh/authorized_keys` file. You can then log into that account without a password:

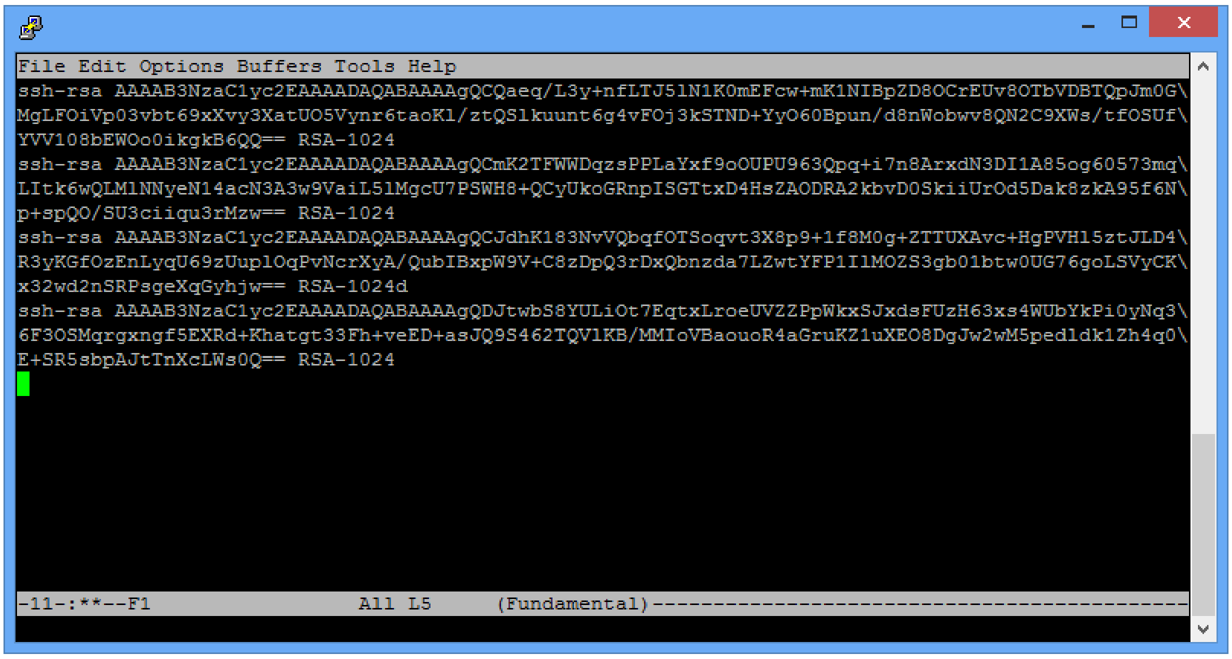
   ```
   ssh username@remote_host
   ```

   Alternatively, you can copy and paste the public key from PuTTYgen or Altair Analytics Workbench into the `authorized_keys` file, ensuring that it ends up on a single line.

5. Verify the contents of `~/.ssh/authorized_keys` to ensure that your public key was added properly, by entering the following on the command line:

   ```
   more ~/.ssh/authorized_keys
   ```

   The contents of a typical `~/.ssh/authorized_keys` file might resemble:



```
File Edit Options Buffers Tools Help
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQCQaeq/L3y+nfLTJ51N1K0mEFcw+mK1NIBpZD8OCrEUv8OTbVDBTQpJm0G\
MgLFOiVp03vbt69xXvy3XatUO5Vynr6taoK1/ztQS1kuunt6g4vFOj3kSTND+YyO60Bpun/d8nWobwv8QN2C9XWs/tfOSUf\
YVV108bEWOo0ikgkB6QQ== RSA-1024
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQCmK2TFWWDqzsPPLaYxf9oOUPU963Qpq+i7n8ArxdN3DI1A85og60573mq\
LItk6wQLM1NNyeN14acN3A3w9VaiL51MgcU7PSWH8+QCyUkoGRnpISGTtxD4HsZAODRA2kbvDOSkiiUrOd5Dak8zkA95f6N\
p+spQO/SU3ciiqu3rMzw== RSA-1024
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQCJdhK183NvVQbqfOTSoqvt3X8p9+1f8M0g+ZTTUXAvc+HgPVH15ztJLD4\
R3yKGfOzEnLyqU69zUup1OqPvNcrXyA/QubIBxpW9V+C8zDpQ3rDxQbnzda7LZwtYFP1I1MOZS3gb01btw0UG76goLSVyCK\
x32wd2nSRPsgeXqGyhjw== RSA-1024d
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDJtwbS8YULiOt7EqtxLroeUVZZPpWkxSJxdsFUzH63xs4WUbYkPi0yNq3\
6F3OSMqrgxngf5EXRd+Khatgt33Fh+veED+asJQ9S462TQV1KB/MMIoVBaouoR4aGruKZ1uXEO8DgJw2wM5pedldk1Zh4q0\
E+SR5sbpAJtTnXcLWs0Q== RSA-1024

-11-:**--F1                All L5      (Fundamental)------------------------------------------
```

Note:

If you look carefully, you can see that the above file contains four public keys - each begins with `ssh-rsa` and ends with a phrase similar to `RSA-1024`.

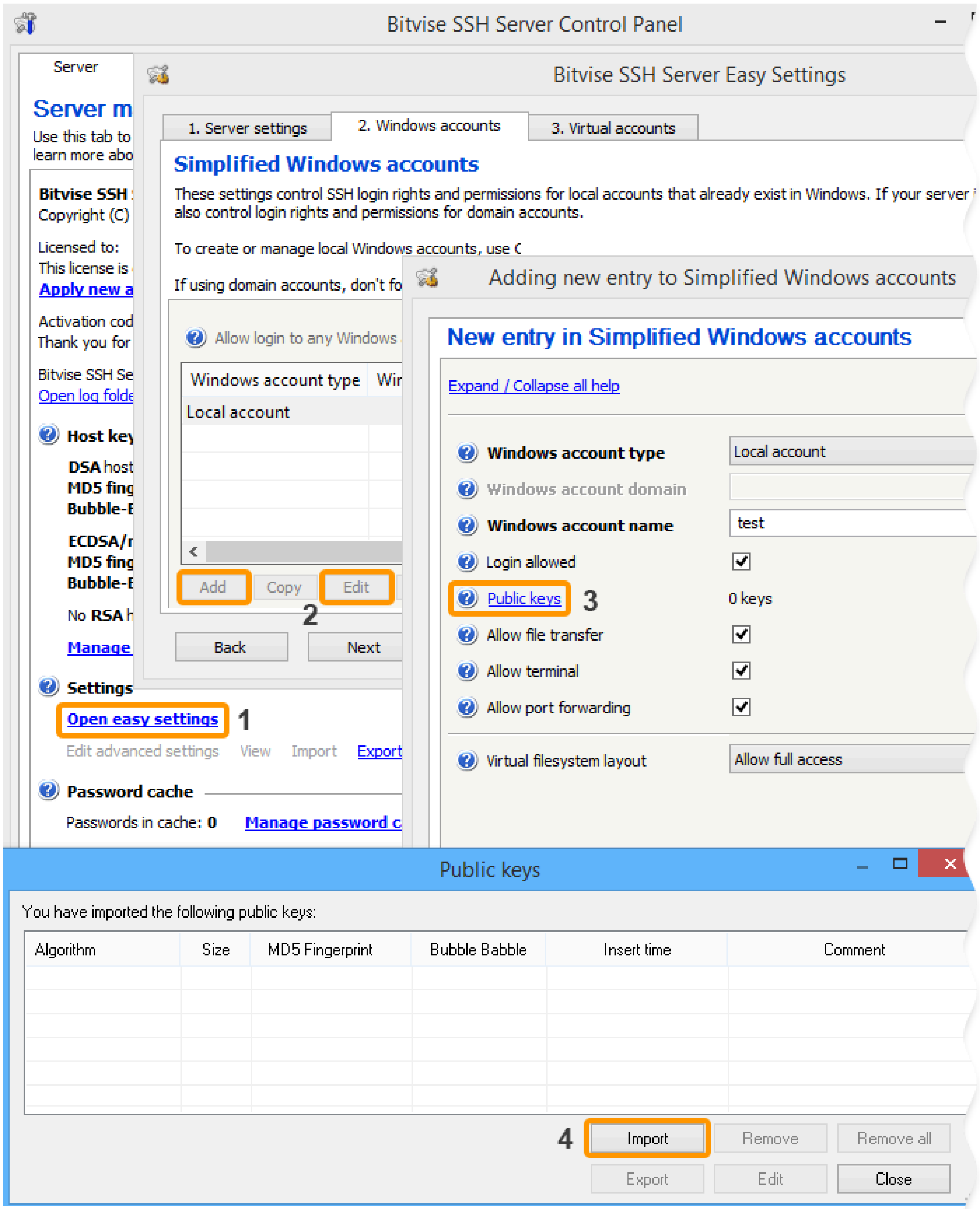
Deploying public keys on Bitvise SSH Server

If you are new to Bitvise SSH Server (refer to the Bitvise documentation for specific configuration details), we highly recommend that you first make sure that you can establish a working SSH connection before you change any settings on the server. If you cannot connect to the SSH server using its default configuration, this is most likely due to a network or firewall problem that you will need to resolve before you are able to connect. In its default configuration, Bitvise SSH Server accepts connections on the often-used port number for SSH servers, 22. This is the only port that you need to open in your firewall in order to connect to the SSH server. If you use port forwarding to tunnel other applications through SSH, you should **not** open any additional ports for the tunnelled connections. All tunnelled connections are forwarded through the SSH session, established through port 22.

1. When connecting to Bitvise SSH Server with an SSH client for the first time, log in with the username and password of a Windows account that exists on the machine where the SSH server is running. To log into a Windows domain account, specify it in the `domain\account` format.

   You can use any SSH client to log into Bitvise SSH Server, as long as it supports SSH protocol version 2.
2. Having ensured that the public key has been saved to a file, transfer it to the machine where Bitvise SSH Server is installed, or to the machine from which you manage the SSH Server remotely using Bitvise SSH Client.

3. Open the **SSH Server Control Panel**, and then, to import the public key into the SSH user's account settings, use either **Open easy settings**:



Bitvise SSH Server Control Panel

Server

Server m

Use this tab to
learn more abo

Bitvise SSH
Copyright (C)

Licensed to:
This license is
Apply new a

Activation cod
Thank you for

Bitvise SSH Se
Open log folde

Host key

DSA host
MD5 fing
Bubble-E

ECDSA/r
MD5 fing
Bubble-E

No RSA h

Manage

Settings

Open easy settings 1

Edit advanced settings View Import Export

Password cache

Passwords in cache: 0 Manage password c

Bitvise SSH Server Easy Settings

1. Server settings | 2. Windows accounts | 3. Virtual accounts

Simplified Windows accounts

These settings control SSH login rights and permissions for local accounts that already exist in Windows. If your server
also control login rights and permissions for domain accounts.

To create or manage local Windows accounts, use C

If using domain accounts, don't fo

Allow login to any Windows

Windows account type	Wir
Local account	

Add | Copy | Edit 2

Back | Next

Adding new entry to Simplified Windows accounts

New entry in Simplified Windows accounts

Expand / Collapse all help

Windows account type	Local account
Windows account domain	
Windows account name	test
Login allowed	✓
Public keys 3	0 keys
Allow file transfer	✓
Allow terminal	✓
Allow port forwarding	✓
Virtual filesystem layout	Allow full access

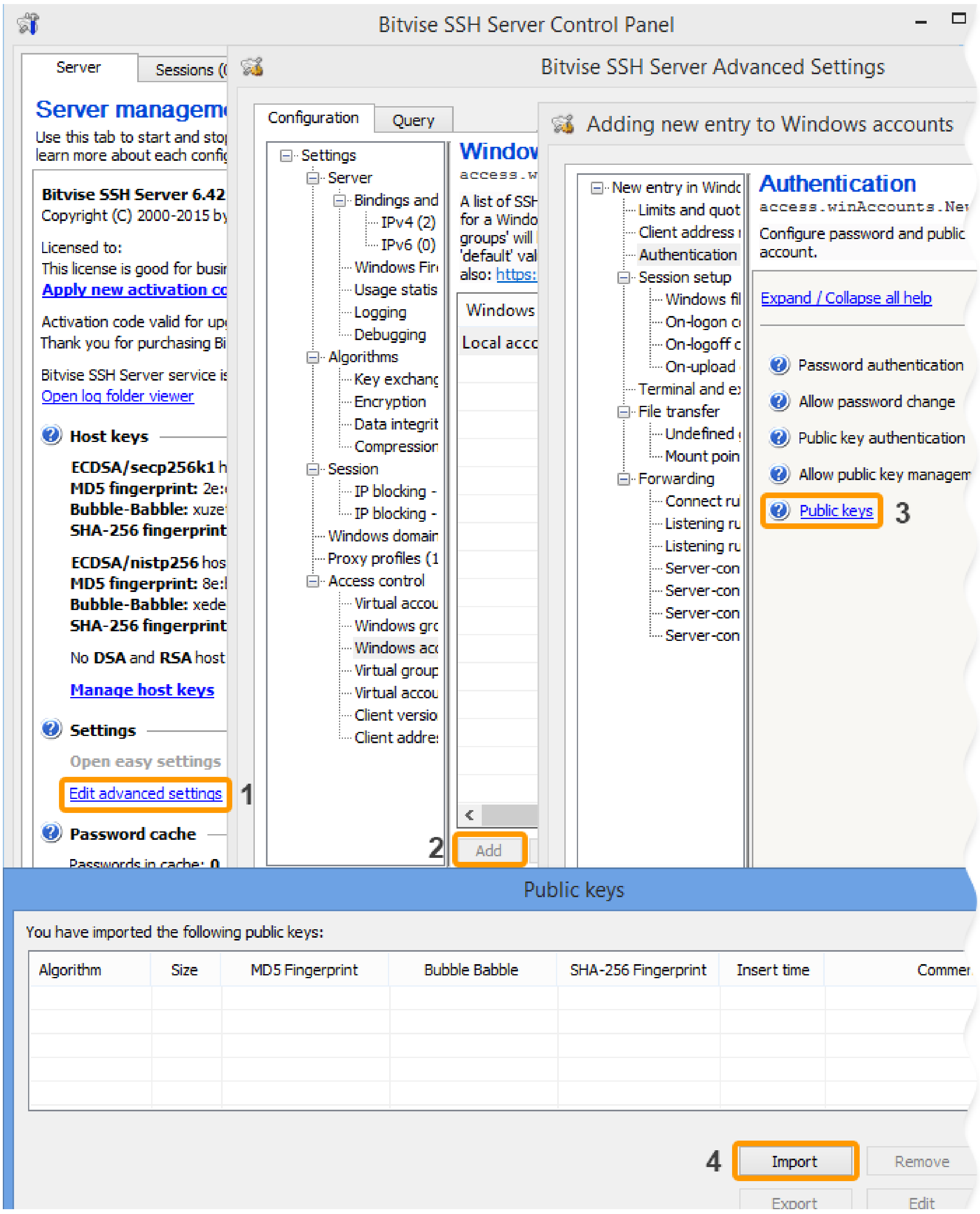
Public keys

You have imported the following public keys:

Algorithm	Size	MD5 Fingerprint	Bubble Babble	Insert time	Comment

4 Import | Remove | Remove all
Export | Edit | Close

or **Edit advanced settings**:



Bitvise SSH Server Control Panel

Server | Sessions (

Server manageme

Use this tab to start and sto
learn more about each confi

Bitvise SSH Server 6.42
Copyright (C) 2000-2015 by

Licensed to:
This license is good for busi
Apply new activation co

Activation code valid for up
Thank you for purchasing Bi

Bitvise SSH Server service i
Open log folder viewer

Host keys

ECDSA/secp256k1 h
MD5 fingerprint: 2e:
Bubble-Babble: xuze
SHA-256 fingerprint

ECDSA/nistp256 hos
MD5 fingerprint: 8e:
Bubble-Babble: xede
SHA-256 fingerprint

No **DSA** and **RSA** host

Manage host keys

Settings

Open easy settings

Edit advanced settings 1

Password cache

Passwords in cache: 0

Bitvise SSH Server Advanced Settings

Configuration | Query

Window
access.w
A list of SSH
for a Windo
groups' will
'default' val
also: https:

Windows
Local acco

2 Add

Adding new entry to Windows accounts

Authentication
access.winAccounts.Ne
Configure password and public
account.

Expand / Collapse all help

Password authentication
Allow password change
Public key authentication
Allow public key managem
Public keys 3

Public keys

You have imported the following public keys:

Algorithm	Size	MD5 Fingerprint	Bubble Babble	SHA-256 Fingerprint	Insert time	Commer

4 Import | Remove
Export | Edit

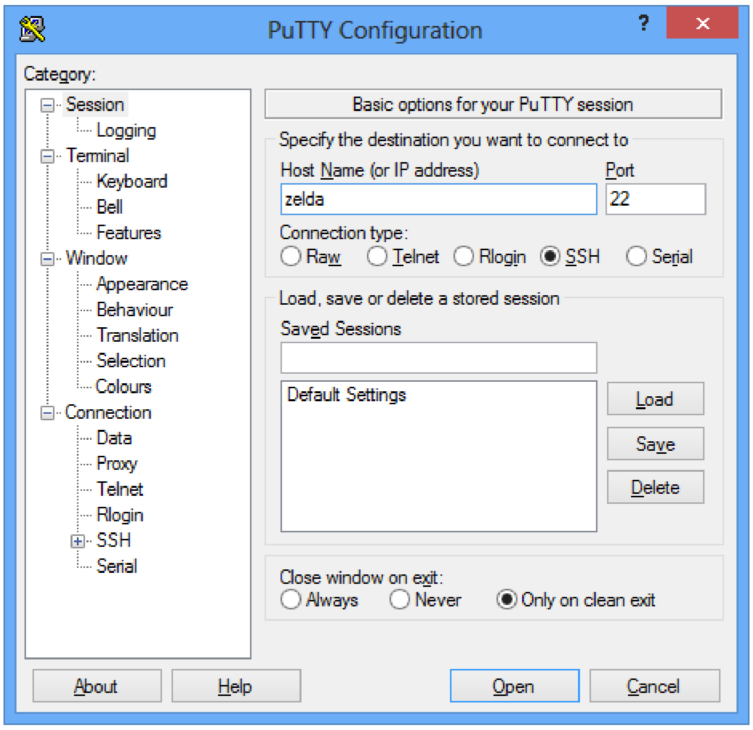
Note:

For Windows accounts, Bitvise SSH Server also supports synchronisation with `~/.ssh/authorized_keys`, provided that this feature is enabled in **Advanced SSH Server settings**, under **Access control**. If this feature is enabled, Bitvise SSH Server will check for the existence of the `authorized_keys` file when the user logs out. If the file exists, Bitvise SSH Server will replace all the public keys configured for the user with the keys found in this file.

Remote host access verification using PuTTY

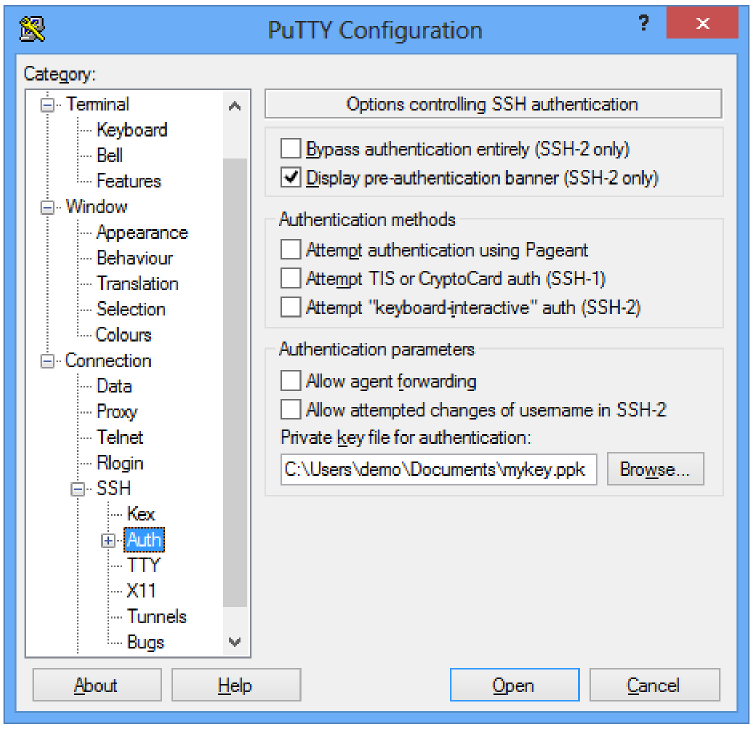
1. Verify authentication using PuTTY, as follows.

   a. Launch the PuTTY client and enter the host name in the **Host Name** field:



Host Name (or IP address): zelda
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

   b. Select the **SSH** ➤ **Auth** configuration page from the category list on the left and ensure that nothing is checked under **Authentication methods** (unless you are verifying *Passphrase authentication using Pageant* (page 37), in which case **Attempt authentication using Pageant** needs to be selected).


PuTTY Configuration

Category:

Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Auth
TTY
X11
Tunnels
Bugs

Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

- [ ] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [ ] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication: C:\Users\demo\Documents\mykey.ppk

c. In the **Private key file for authentication** field, enter the name of the private key file you generated via either *Key generation using PuTTYgen* (page 23) or *Key generation using Altair Analytics Workbench* (page 26).

d. Click **Open** - you will be authenticated via your key-pair combination and a console window will open. The window displays a notification regarding the authentication method.

Note:

If you are prompted for a password, then public key authentication has failed.

2. If public key authentication is successful, then, if you are using Altair SLC Link, create the required host connection and remote host server through Altair Analytics Workbench. If you are using Altair SLC Communicate, sign on using your private key, via the `SIGNON` statement, for which you need to specify either the **IDENTITYFILE** statement option or the **SSH_IDENTITYFILE** system option, for example:

   ```
   SIGNON <servername> SSH
   USERNAME="<username>"
   IDENTITYFILE="C:\Users\techwriter\.ssh\wpscommunicate.ppk"
   LAUNCHCMD="/home/installs/wps-3.2/bin/wps -dmr ";
   ```

   Alternatively:

   ```
   OPTIONS SSH_IDENTITYFILE="C:\Users\techwriter\.ssh\wpscommunicate.ppk";
   SIGNON <servername> SSH
   username="<username>"
   LAUNCHCMD="/home/installs/wps-3.2/bin/wps -dmr";
   ```

   Note:

   You cannot use either **IDENTITYFILE** or **SSH_IDENTITYFILE** if you are using *Passphrase authentication using Pageant* (page 37).

Passphrase authentication using Pageant

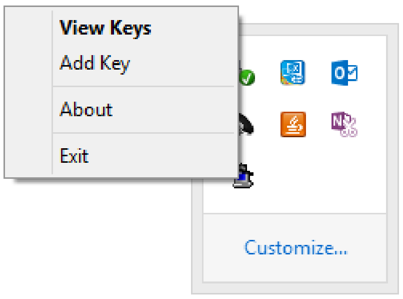
Altair SLC Communicate does not support the reading of private key files that have been saved in encrypted form with a **passphrase**. However, it is still possible to use private key-pairs of this form if you use a keychain agent. We will assume that you are using Pageant on Windows.

A keychain agent is a utility that runs on the client machine and stores the decrypted public key file. The SSH client contacts the agent for a public key to use when connecting to a given host. Although a password is still required to decrypt the private key, this is only required once - when the keychain agent first opens the private key.

This procedure assumes that you have already generated a key-pair with a **passphrase**, and deployed the public key on the remote SSH server (for UNIX/Linux), or on Bitvise SSH Server (for Windows).

Proceed as follows:

1. Run the Pageant tool, right-click the **system tray** icon and choose **View Keys** or **Add Keys** to add your private key file.

   At this point you will be prompted for the **passphrase** for the private key file.
2. Pageant opens and decrypts it, allowing clients to request the loaded identity list for authentication.

SSH (Secure Shell) from a UNIX client

Before you access a remote host via Altair SLC Communicate or Altair SLC Link, it is important to ensure that you can access the remote host manually via SSH. This demonstrates that you can at least connect to the machine using the SSH protocol and that your user ID and password are valid.

Note:

If you intend to use *Public key authentication* (page 39), and keys have not already been generated on the server, then you may wish to use **ssh-keygen** (refer to *Key generation using ssh-keygen* (page 40)). If you intend to use public keys with a **passphrase**, and you are using Altair SLC Communicate, you will also need to download a keychain agent such as **ssh-agent** (refer to *Passphrase authentication using ssh-agent* (page 47)). The use of such an agent for **passphrases** is not necessary with Altair SLC Link, although it may be desirable if you are connecting to multiple servers.

Password authentication and sign-on

Altair SLC will use the OpenSSH host key database stored in the `~/.ssh/known_hosts` file by default. It is necessary to log onto the remote host using the OpenSSH client's command line to validate and accept the host key, and ensure that it is added it to the `known_hosts` file before attempting to make a connection. With OpenSSH, it is also possible for a system administrator to add keys manually to the `/etc/ssh/ssh_known_hosts` file, in preference to the `~/.ssh/known_hosts` file.

To sign on via SSH to a remote host, and ensure that password authentication is employed:

1. Issue the following command:

   ```
   ssh -o PreferredAuthentications=password <hostname>
   ```

2. Verify that you receive a password prompt:

```
<user>@<hostname>'s password:
```

Note:

It is important to ensure that password authentication is being used, and that the host is not, for example, configured to accept only public key sign-on.

3. If you are using Altair SLC Link, create the required host connection and remote host server through Altair Analytics Workbench. If you are using Altair SLC Communicate, sign on via the `SIGNON` statement - you need to specify either the **IDENTITYFILE** statement option or the **SSH_IDENTITYFILE** system option, for example:

```
SIGNON <servername> SSH
USERNAME="<username>"
password="<password>"
LAUNCHCMD="/home/installs/wps/bin/wps -dmr ";

RSUBMIT;
%PUT &SYSHOSTNAME;
ENDRSUBMIT;
SIGNOFF;
```

Alternatively:

```
OPTIONS SSH_IDENTITYFILE="C:\Users\techwriter\.ssh\wpscommunicate.ppk";
SIGNON <servername> SSH
password="<password>"
LAUNCHCMD="/home/installs/wps/bin/wps -dmr";

RSUBMIT;
%PUT &SYSHOSTNAME;
ENDRSUBMIT;
SIGNOFF;
```

Note:

You cannot use either **IDENTITYFILE** or **SSH_IDENTITYFILE** if you are using *Passphrase authentication using ssh-agent* (page 47).

Public key authentication

The method described by which the keys can be generated on a UNIX client, is *Key generation using ssh-keygen* (page 40).

Following the generation of the public keys, they should be placed on the remote server in accordance with *Deploying public keys on the remote SSH server* (page 28), or, in the case of a Windows server, *Deploying public keys on Bitvise SSH Server* (page 30).

If you wish to connect to multiple servers, without having to remember or enter your password for each system, then you should also use *Passphrase authentication using ssh-agent* (page 47).

The validity of the key-pairs should then be checked in accordance with *Remote host access verification and sign-on* (page 46).

Note:

Public key authentication can be used with both Altair SLC Communicate and Altair SLC Link. However, for Altair SLC Communicate, if you are not using a keychain agent such as **ssh-agent**, there cannot be a **passphrase** on the private key file, as there is currently no interactive mechanism to prompt for it during authentication.

Note:

You need to ensure that public key authentication is not disabled on the client machine.

Key generation using ssh-keygen

The **ssh-keygen** program allows you to create RSA keys for use by SSH protocol version 2. The type of key to be generated is specified by the `-t` option. If invoked without any arguments, **ssh-keygen** will generate an RSA key for use in SSH protocol 2 connections.

1. Generate the key-pair. In the following example, we have logged onto `hostA` as `wplusr`:

   ```
   ssh-keygen -t rsa
   Generating public/private rsa key pair.
   Enter file in which to save the key (/home/wplusr/.ssh/id_rsa):
   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   Your identification has been saved in /home/wplusr/.ssh/id_rsa.
   Your public key has been saved in /home/wplusr/.ssh/id_rsa.pub.
   The key fingerprint is:
   f6:61:a8:27:35:cf:4c:6d:13:22:70:cf:4c:c8:a0:23 wplusr@hostA
   ```

   Note:

   Do not enter a **passphrase** if you are using Altair SLC Communicate. You need to use a keychain agent for this (refer to *Passphrase authentication using ssh-agent* (page 47)). In the above example, the private key was saved in `.ssh/id_rsa` (this file is read-only and only for you, and no-one else must see the contents of that file, as it is used to decrypt all correspondence encrypted with the public key), and the public key in `.ssh/id_rsa.pub`. This is the file that needs to be added to the `~/.ssh/authorized_keys` file on the remote machine.

2. To deploy the public key, proceed as in *Deploying public keys on the remote SSH server* (page 28) or *Deploying public keys on Bitvise SSH Server* (page 30).

Deploying public keys on the remote SSH server

1. Log into the remote machine.

2. Once logged in, you must configure the server to accept your public key for authentication, so change into the `.ssh` directory and open the file `authorized_keys`.

   If this is the first public key to be put into the file, then you may need to create the directory and file first, by, for example, running the following commands:

   ```
   mkdir -p .ssh
   touch ~/.ssh/authorized_keys
   ```

3. Set the right permissions, for example:

   ```
   chmod 600 ~/.ssh/authorized_keys
   ```

   Note:

   You also need to ensure that your `$HOME` directory and `.ssh` directory have the permissions that are are appropriate to both the server and your particular operation.

4. Now you can add the public key to the `authorized_keys` file, as in the following example:

   ```
   cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
   ```

   If you currently have password-based SSH access configured to your server, and you have the `ssh-copy-id` utility installed, then you can simply transfer your public key by typing:

   ```
   ssh-copy-id username@remote_host
   ```

   You will then be prompted for the user account's password on the remote system. After typing in the password, the contents of your `~/.ssh/id_rsa.pub` key will be appended to the end of the user account's `~/.ssh/authorized_keys` file. You can then log into that account without a password:

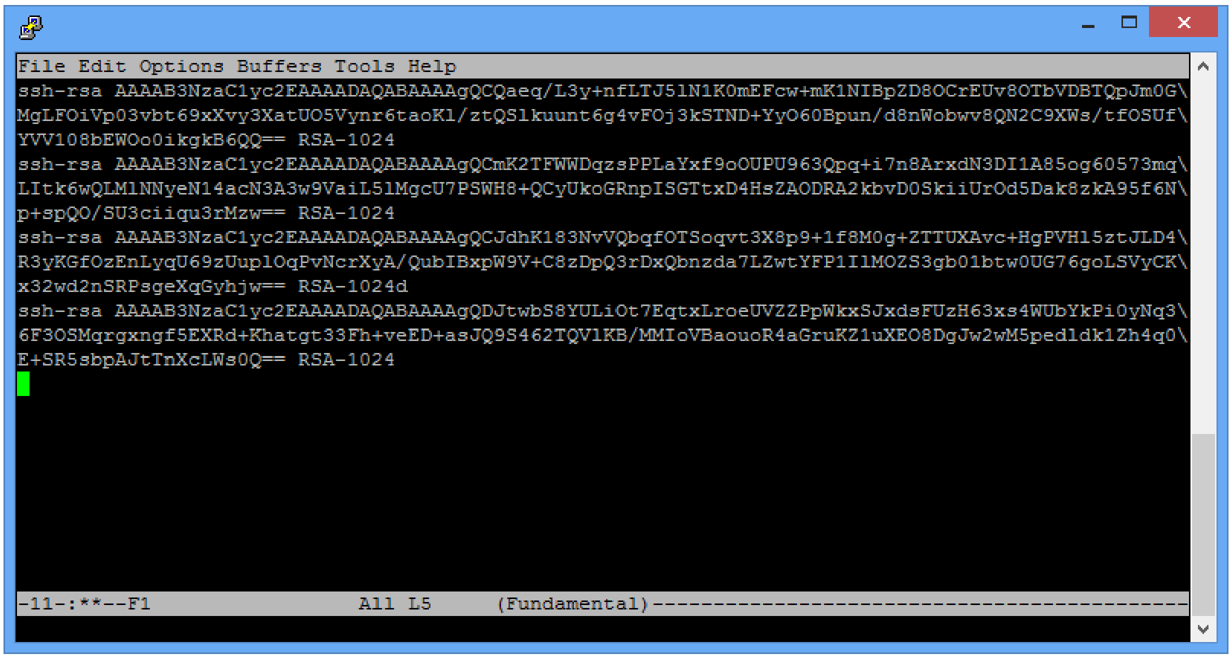
   ```
   ssh username@remote_host
   ```

   Alternatively, you can copy and paste the public key from PuTTYgen or Altair Analytics Workbench into the `authorized_keys` file, ensuring that it ends up on a single line.

5. Verify the contents of `~/.ssh/authorized_keys` to ensure that your public key was added properly, by entering the following on the command line:

   ```
   more ~/.ssh/authorized_keys
   ```

   The contents of a typical `~/.ssh/authorized_keys` file might resemble:



```
File Edit Options Buffers Tools Help
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQCQaeq/L3y+nfLTJ51N1K0mEFcw+mK1NIBpZD8OCrEUv8OTbVDBTQpJm0G\
MgLFOiVp03vbt69xXvy3XatUO5Vynr6taoKl/ztQSlkuunt6g4vFOj3kSTND+YyO60Bpun/d8nWobwv8QN2C9XWs/tfOSUf\
YVV108bEWOo0ikgkB6QQ== RSA-1024
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQCmK2TFWWDqzsPPLaYxf9oOUPU963Qpq+i7n8ArxdN3DI1A85og60573mq\
LItk6wQLM1NNyeN14acN3A3w9VaiL51MgcU7PSWH8+QCyUkoGRnpISGTtxD4HsZAODRA2kbvD0SkiiUrOd5Dak8zkA95f6N\
p+spQO/SU3ciiqu3rMzw== RSA-1024
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQCJdhK183NvVQbqfOTSoqvt3X8p9+1f8M0g+ZTTUXAvc+HgPVH15ztJLD4\
R3yKGfOzEnLyqU69zUup1OqPvNcrXyA/QubIBxpW9V+C8zDpQ3rDxQbnzda7LZwtYFP1I1MOZS3gb01btw0UG76goLSVyCK\
x32wd2nSRPsgeXqGyhjw== RSA-1024d
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDJtwbS8YULiOt7EqtxLroeUVZZPpWkxSJxdsFUzH63xs4WUbYkPi0yNq3\
6F3OSMqrgxngf5EXRd+Khatgt33Fh+veED+asJQ9S462TQV1KB/MMIoVBaouoR4aGruKZ1uXEO8DgJw2wM5pedldk1Zh4q0\
E+SR5sbpAJtTnXcLWs0Q== RSA-1024




-11-:**--F1            All L5      (Fundamental)-----------------------------------------
```

> **Note:**
>
> If you look carefully, you can see that the above file contains four public keys - each begins with `ssh-rsa` and ends with a phrase similar to `RSA-1024`.

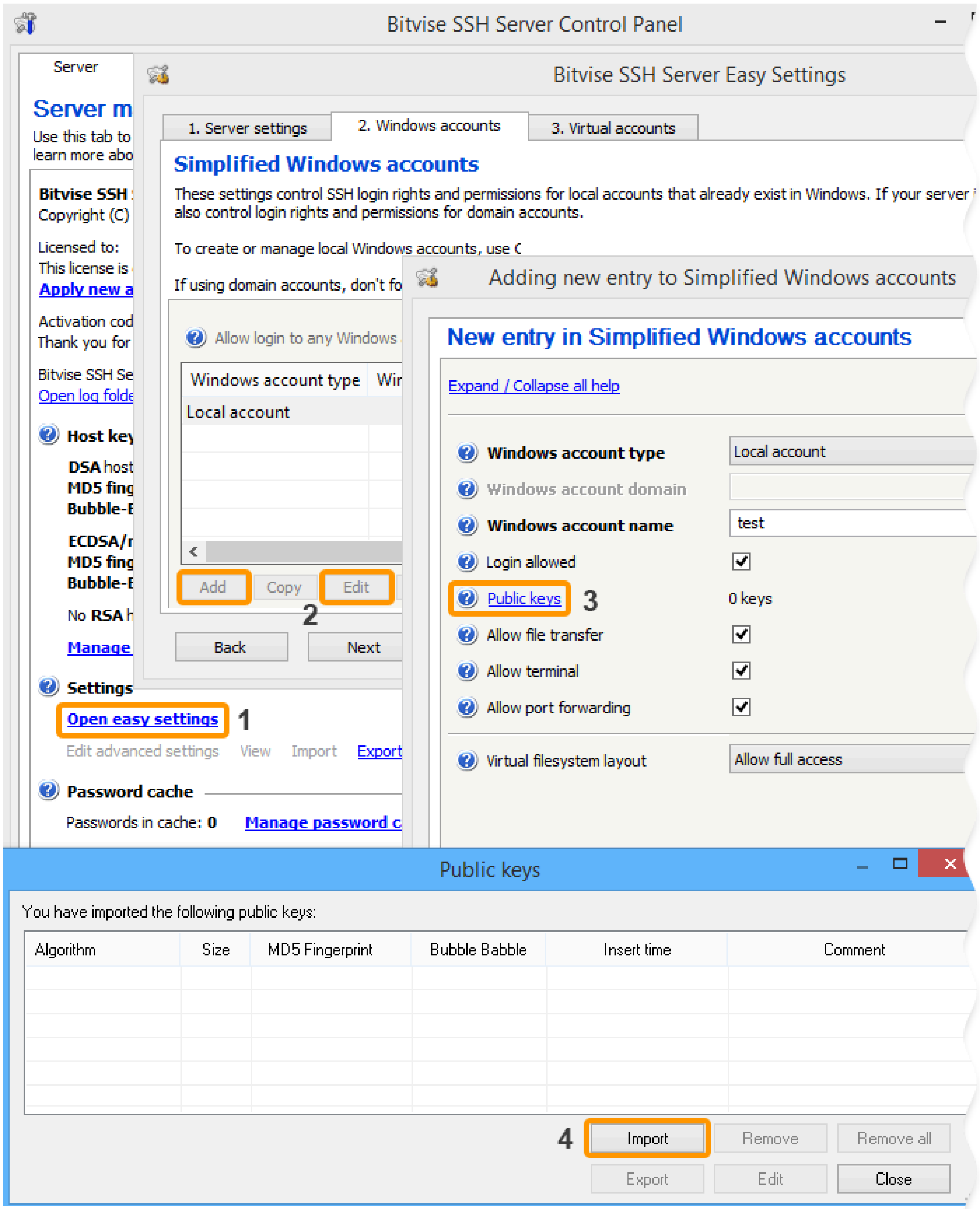
Deploying public keys on Bitvise SSH Server

If you are new to Bitvise SSH Server (refer to the Bitvise documentation for specific configuration details), we highly recommend that you first make sure that you can establish a working SSH connection before you change any settings on the server. If you cannot connect to the SSH server using its default configuration, this is most likely due to a network or firewall problem that you will need to resolve before you are able to connect. In its default configuration, Bitvise SSH Server accepts connections on the often-used port number for SSH servers, 22. This is the only port that you need to open in your firewall in order to connect to the SSH server. If you use port forwarding to tunnel other applications through SSH, you should **not** open any additional ports for the tunnelled connections. All tunnelled connections are forwarded through the SSH session, established through port 22.

1. When connecting to Bitvise SSH Server with an SSH client for the first time, log in with the username and password of a Windows account that exists on the machine where the SSH server is running. To log into a Windows domain account, specify it in the `domain\account` format.

   You can use any SSH client to log into Bitvise SSH Server, as long as it supports SSH protocol version 2.
2. Having ensured that the public key has been saved to a file, transfer it to the machine where Bitvise SSH Server is installed, or to the machine from which you manage the SSH Server remotely using Bitvise SSH Client.

3. Open the **SSH Server Control Panel**, and then, to import the public key into the SSH user's account settings, use either **Open easy settings**:



Bitvise SSH Server Control Panel

Server

Server m
Use this tab to
learn more abo

Bitvise SSH
Copyright (C)

Licensed to:
This license is
Apply new a

Activation cod
Thank you for

Bitvise SSH Se
Open log folde

Host key
DSA host
MD5 fing
Bubble-E

ECDSA/r
MD5 fing
Bubble-E

No RSA h
Manage

Settings
Open easy settings 1
Edit advanced settings View Import Export

Password cache
Passwords in cache: 0 Manage password c

Bitvise SSH Server Easy Settings

1. Server settings | 2. Windows accounts | 3. Virtual accounts

Simplified Windows accounts

These settings control SSH login rights and permissions for local accounts that already exist in Windows. If your server
also control login rights and permissions for domain accounts.

To create or manage local Windows accounts, use C

If using domain accounts, don't fo

Allow login to any Windows

Windows account type	Wir
Local account	

Add Copy Edit 2

Back Next

Adding new entry to Simplified Windows accounts

New entry in Simplified Windows accounts

Expand / Collapse all help

Windows account type	Local account
Windows account domain	
Windows account name	test
Login allowed	✓
Public keys 3	0 keys
Allow file transfer	✓
Allow terminal	✓
Allow port forwarding	✓
Virtual filesystem layout	Allow full access

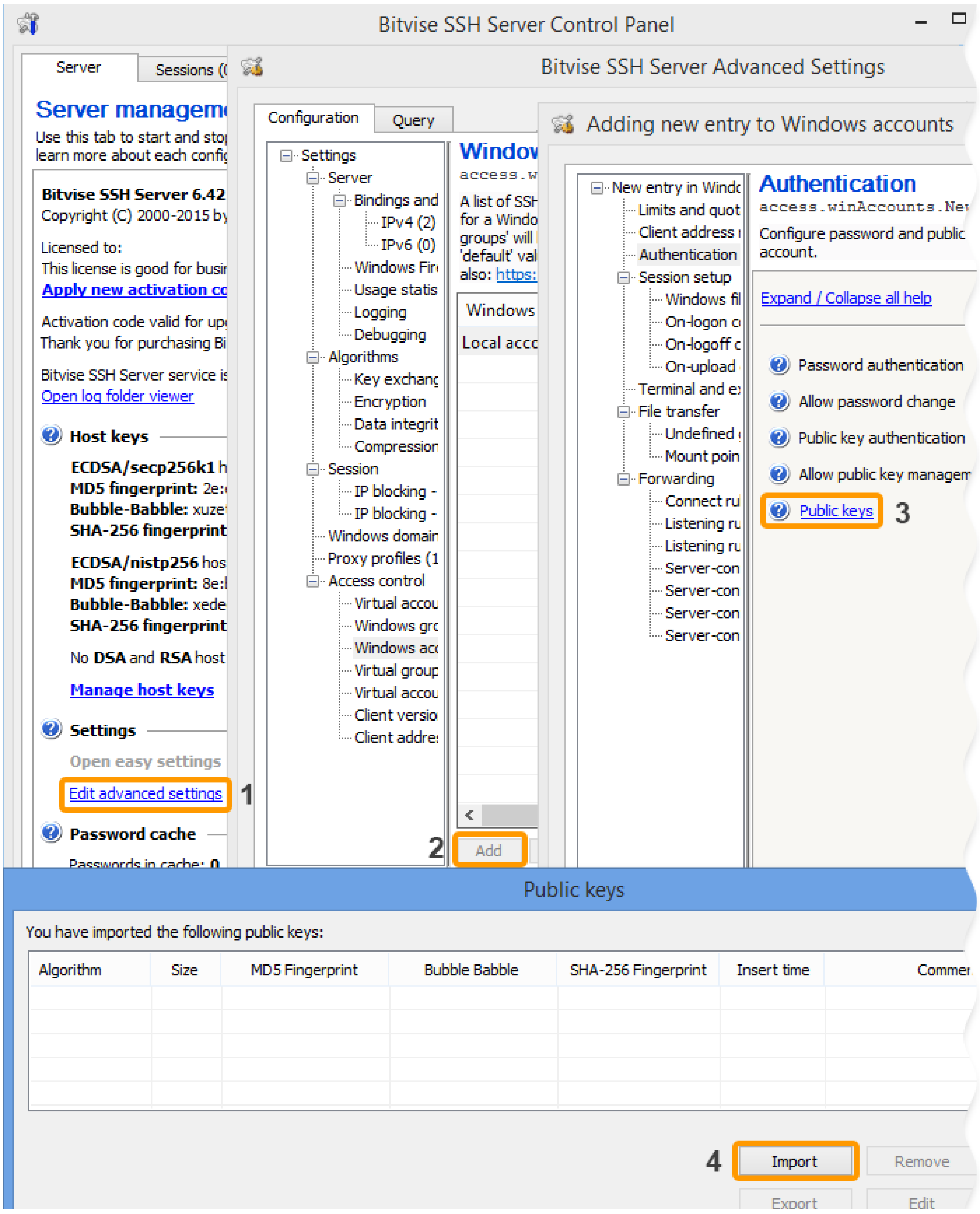
Public keys

You have imported the following public keys:

Algorithm	Size	MD5 Fingerprint	Bubble Babble	Insert time	Comment

4 Import Remove Remove all
Export Edit Close

or **Edit advanced settings**:



Bitvise SSH Server Control Panel

Server | Sessions (

Server managem

Use this tab to start and sto
learn more about each confi

Bitvise SSH Server 6.42
Copyright (C) 2000-2015 b

Licensed to:
This license is good for busi
Apply new activation c

Activation code valid for up
Thank you for purchasing B

Bitvise SSH Server service i
Open log folder viewer

Host keys

ECDSA/secp256k1 h
MD5 fingerprint: 2e:
Bubble-Babble: xuze
SHA-256 fingerprint

ECDSA/nistp256 hos
MD5 fingerprint: 8e:
Bubble-Babble: xede
SHA-256 fingerprint

No **DSA** and **RSA** host

Manage host keys

Settings

Open easy settings

Edit advanced settings 1

Password cache

Passwords in cache: 0

Bitvise SSH Server Advanced Settings

Configuration | Query

- Settings
  - Server
    - Bindings and
      - IPv4 (2)
      - IPv6 (0)
    - Windows Fir
    - Usage statis
    - Logging
    - Debugging
  - Algorithms
    - Key exchang
    - Encryption
    - Data integrit
    - Compression
  - Session
    - IP blocking -
    - IP blocking -
  - Windows domai
  - Proxy profiles (1
  - Access control
    - Virtual accou
    - Windows grc
    - Windows acc
    - Virtual group
    - Virtual accou
    - Client versio
    - Client addre

Windov
access.w

A list of SSH
for a Windo
groups' will
'default' val
also: https:

Windows
Local acc

2 Add

Adding new entry to Windows accounts

- New entry in Windc
  - Limits and quot
  - Client address
  - Authentication
  - Session setup
    - Windows fil
    - On-logon c
    - On-logoff c
    - On-upload
  - Terminal and e
  - File transfer
    - Undefined
    - Mount poin
  - Forwarding
    - Connect ru
    - Listening ru
    - Listening ru
    - Server-con
    - Server-con
    - Server-con
    - Server-con

Authentication
access.winAccounts.Ne

Configure password and public
account.

Expand / Collapse all help

Password authentication
Allow password change
Public key authentication
Allow public key managem
Public keys 3

Public keys

You have imported the following public keys:

Algorithm	Size	MD5 Fingerprint	Bubble Babble	SHA-256 Fingerprint	Insert time	Commer

4 Import | Remove
Export | Edit

Note:

For Windows accounts, Bitvise SSH Server also supports synchronisation with `~/.ssh/authorized_keys`, provided that this feature is enabled in **Advanced SSH Server settings**, under **Access control**. If this feature is enabled, Bitvise SSH Server will check for the existence of the `authorized_keys` file when the user logs out. If the file exists, Bitvise SSH Server will replace all the public keys configured for the user with the keys found in this file.

Remote host access verification and sign-on

1. Verify that login to the remote server can take place, for example:

```
jsmith@local-host$ ssh jsmith@remote-host
Last login: Wed Oct 21 17:22:33 2015 from 192.168.1.2
[Note: SSH did not ask for password.]

jsmith@remote-host$ [Note: You are on remote-host here]
```

Note:

If you are prompted for a password, then public key authentication has failed.

2. If public key authentication is successful, then, if you are using Altair SLC Link, create the required host connection and remote host server through Altair Analytics Workbench. If you are using Altair SLC Communicate, sign on using your private key, via the `SIGNON` statement, for which you need to specify either the **IDENTITYFILE** statement option or the **SSH_IDENTITYFILE** system option, for example:

```
SIGNON <servername> SSH
USERNAME="<username>"
IDENTITYFILE="C:\Users\techwriter\.ssh\wpscommunicate.ppk"
LAUNCHCMD="/home/installs/wps-3.2/bin/wps -dmr ";
```

Alternatively:

```
OPTIONS SSH_IDENTITYFILE="C:\Users\techwriter\.ssh\wpscommunicate.ppk";
SIGNON <servername> SSH
username="<username>"
LAUNCHCMD="/home/installs/wps-3.2/bin/wps -dmr";
```

Note:

You cannot use either **IDENTITYFILE** or **SSH_IDENTITYFILE** if you are using *Passphrase authentication using ssh-agent* (page 47).

Passphrase authentication using ssh-agent

Altair SLC Communicate does not support the reading of private key files that have been saved in encrypted form with a **passphrase**. However, it is still possible to use private key-pairs of this form if you use a keychain agent. We will assume that you are using OpenSSH's **ssh-agent** for UNIX/Linux.

The **ssh-agent** program runs on the client system, acting as a temporary store of private keys in decrypted form. When the SSH client authenticates with a remote host, it can fetch the private key from the agent without needing to prompt the user.

On startup, **ssh-agent** does not hold any keys. These are loaded from disk using the **ssh-add** command, at which point the user enters each key's passphrase to decrypt it. The agent can store multiple keys simultaneously, from which the system will automatically choose the correct key for the remote server.

The following procedure assumes that you have already generated a key-pair with a **passphrase**, and deployed the public key on the remote SSH server (for UNIX/Linux), or on Bitvise SSH Server (for Windows).

Proceed as follows:

1. Start **ssh-agent** by typing the following into your local terminal session:

   ```
   eval $(ssh-agent)
   ```

   This will start the agent program and place it into the background.

   Note:

   The `eval` command tells the shell to run the output of **ssh-agent** as shell commands. Thereafter, processes run by this shell inherit its environment variables and have access to the agent.

2. Now, you need to add your private key to the agent, so that it can manage your key:

   ```
   ssh-add
   ```

   Note:

   When run without arguments, **ssh-add** automatically adds the files `~/.ssh/id_rsa`, `~/.ssh/id_dsa`, `~/.ssh/id_ecdsa`, `~/.ssh/id_ed25519` and `~/.ssh/identity`.

3. You are prompted to enter your **passphrase**:

```
Enter passphrase for /home/demo/.ssh/id_rsa:
Identity added: /home/demo/.ssh/id_rsa (/home/demo/.ssh/id_rsa)
```

Note:

If you wish to be able to connect without a password to one server from within another server, you will need to forward your SSH key information. This will allow you to authenticate to another server through the server to which you are connected, using the credentials on your local host. To start this, **ssh-agent** must be running, and your private key must have been added to the agent (see above). You then need to connect to your first server using the `-A` option. This forwards your credentials to the server for this session:

```
ssh -A username@remote_host
```

From here, you can SSH into any other host that your SSH key is authorised to access. You will connect as if your private SSH key was located on this server.

Kerberos single sign-on

You can set up Kerberos single sign-on so that you can sign on to a remote server without directly providing a password or SSH identity file.

This typically requires modifications to the configuration of the SSH daemon to accommodate Kerberos, details of which are outside the scope of this document. You will need to discuss the required modifications with the system administrator of the remote host, as setting up Kerberos authentication is a job for an experienced system administrator.

Kerberos configuration is also required on the client machine.

Before attempting to perform a Kerberos sign-on using Altair SLC, you must be able to perform a Kerberos sign-on to a remote host using a conventional SSH client.

Once you can perform a Kerberos sign-on to a remote host using an external SSH client, you can perform the same sign-on from a SAS language program. You do not need to specify any authentication information with the `SIGNON` statement, but you should ensure that neither the **IDENTITYFILE** statement option nor the **SSH_IDENTITYFILE** system option is used.

Example sign-on code might resemble:

```
SIGNON <servername> SSH
LAUNCHCMD="/home/installs/wps/bin/wps -dmr";
```

Linux clients

The `kinit` command will perform a Kerberos sign-on and prompt you for your Kerberos password. This action acquires a ticket which is cached for a period during which the ticket can be securely passed to other hosts as proof of identity, therefore allowing for authentication to those hosts without the need to provide further authentication information:

```
kinit
ssh -o PreferredAuthentications=gssapi-with-mic <hostname>
```

Specifying the `PreferredAuthentications=gssapi-with-mic` option ensures that SSH only attempts GSSAPI authentication and does not, for example, attempt to use public key authentication which might otherwise proceed without prompting you for a password.

Windows clients

It is possible to configure Kerberos single sign-on on Windows, but there are some restrictions:

- The first is that you cannot be a local administrator on your machine. If you are, Windows will not issue the Kerberos Ticket-Granting Ticket necessary for Altair SLC to perform the Kerberos authentication.
- The second is that you need to set a registry key to allow Windows to give out the Ticket-Granting Ticket, even if you are not in the local administrators group. The value `allowtgtsessionkey` under the following key needs to be set to 1:

  ```
  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\Kerberos\Parameters
  ```

  You may need to add this value to the registry if it is not already present.The above restrictions do not apply to PuTTY, so, if you can log on using PuTTY but not Altair SLC, then one of the above restrictions is likely to be the cause.

Environment variables

Environment variables may need to be configured to provide access to third-party software being used with a particular Altair SLC installation.

Using Altair SLC with third-party applications such as database servers, requires that the environment information is available at the server start up, such as:

- `LD_LIBRARY_PATH` or `LIBPATH` on Linux or Unix systems pointing to client libraries
- `ODBCSYSINI` pointing to the unixODBC client libraries
- `PATH`, for example pointing to the client libraries on Windows, or application directories on Linux or Unix systems.

The recommended method for setting these environment variables is described in the `install-EN`.

Once the required environment variables have been set up by your system administrator, you can test the link server connection using `PROC OPTIONS`:

```
PROC OPTIONS;
RUN;
```

Legal Notices

Trademarks

General Notices

Altair Engineering, inc. is not associated in any way with the SAS Institute.

Altair SLC is not the SAS System.

The phrases "SAS", "SAS language", and "language of SAS" used in this document are used to refer to the computer programming language often referred to in any of these ways.

The phrases "program", "SAS program", and "SAS language program" used in this document are used to refer to programs written in the SAS language. These may also be referred to as "scripts", "SAS scripts", or "SAS language scripts".

The phrases "IML", "IML language", "IML syntax", "Interactive Matrix Language", and "language of IML" used in this document are used to refer to the computer programming language often referred to in any of these ways.

Altair SLC includes software developed by third parties. More information can be found in the THANKS or acknowledgments.txt file included in the Altair SLC installation.