SSH Protocol

Teldat-Dm 787-I

Legal Notice

Table of Contents

I Related Documents

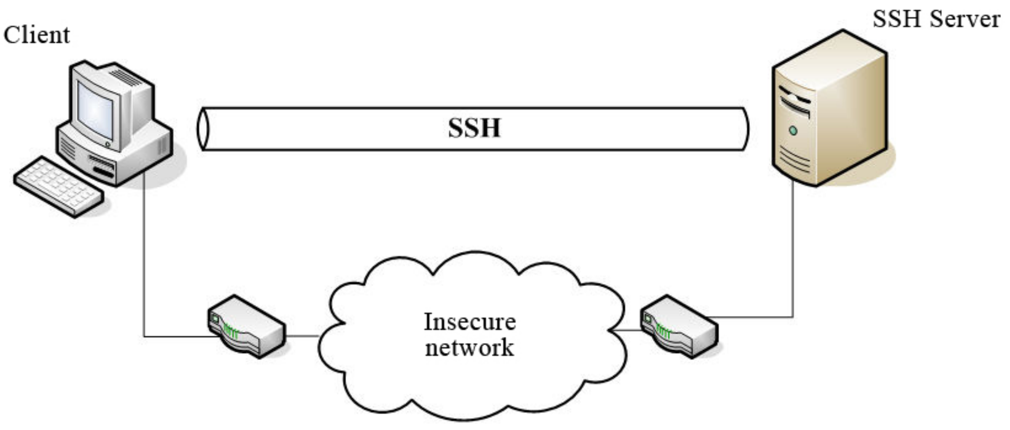
Teldat-Dm 704-I Configuration and Monitoring

Teldat-Dm 800-I AAA Feature

Chapter 1 Introduction

1.1 Introduction

SSH (*Secure Shell*) is a protocol used to obtain secure remote access (login), and other secure network services, over an insecure network.



The SSH architecture has three components:

- Transport layer protocol. Provides server authentication so clients can verify the authenticity of the server they are connecting to. This also provides confidentiality and integrity when *Perfect Forward Secrecy* is used for the connection. This property indicates the keys used in each session are generated from unique material and obtained via a Diffie-Hellman key exchange.
- User authentication protocol: The client authenticates with the server. As a result, connection is only established with the users admitted by the server in its configuration.
- Connection protocol: Multiplexes the encrypted connection and authentication in a group of logical channels. Provides a series of services, including remote terminal access.

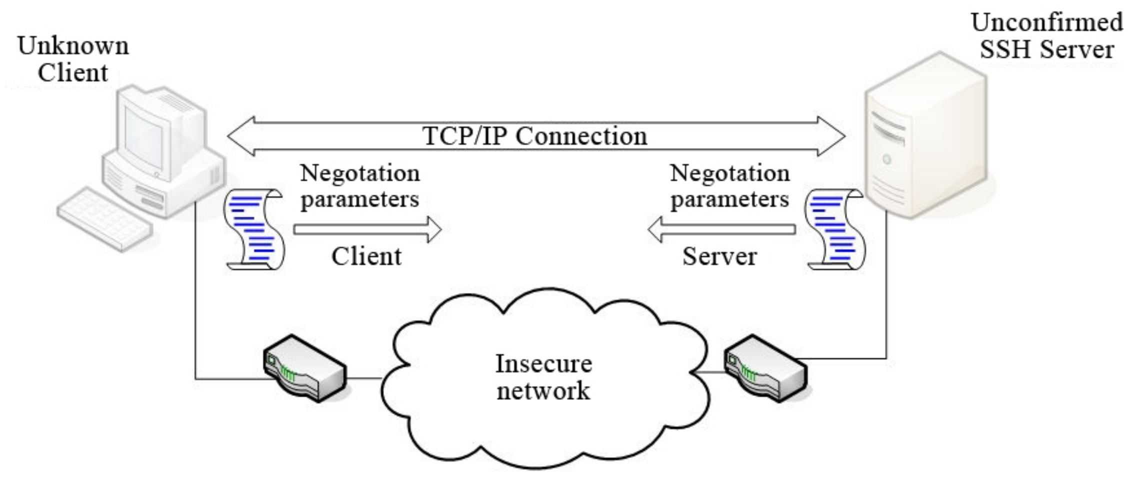
SSH traffic is typically exchanged over a TCP/IP connection. The SSH server listening port assigned by IANA (*Internet Assigned Numbers Authority*) is 22.

The SSH protocol is currently available in two versions:

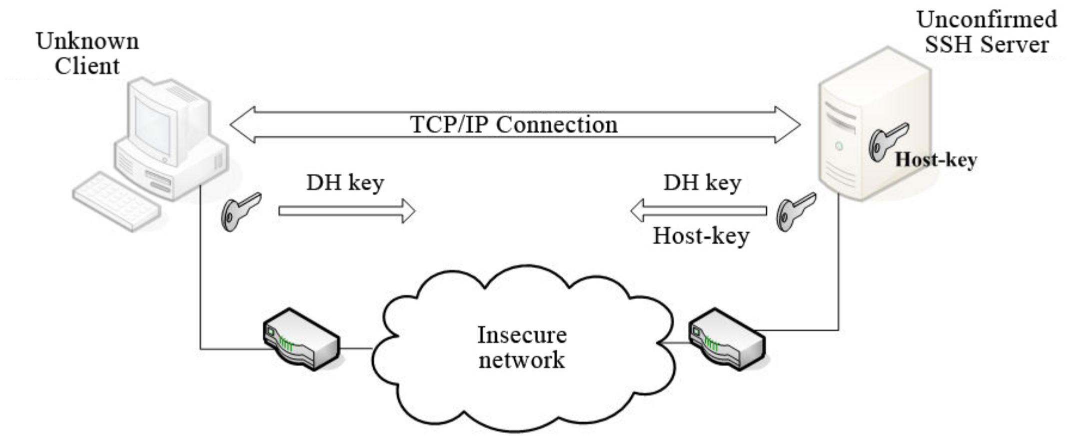
- The first version has not been standardized by the IETF (*Internet Engineering Task Force*). Initially it was distributed with a free license, but later developments were no longer free (despite being related to this first version). They were also subject to patents and governmental restrictions, which allowed them to be used only for non-commercial ends. A series of weaknesses have also been discovered, meaning the use of this version is not recommended.
- The second version, known as SSH, has been standardized by the IETF and is supported by the OpenSSH project, whose aim was to achieve free and portable protocol implementation. The majority of SSH distributions present on the market, both for clients and servers, use this version (which resolves the problems found in the first one and uses more robust encryption mechanisms).

1.1.1 SSH connection phases

Once the TCP/IP connection has been established, client and server exchange a string of characters where they report their SSH versions. Where the server supports both versions, version “SSH-1.99” is implemented. If both versions are compatible, parameter negotiation begins. The rest of the SSHv2 process is described further on.

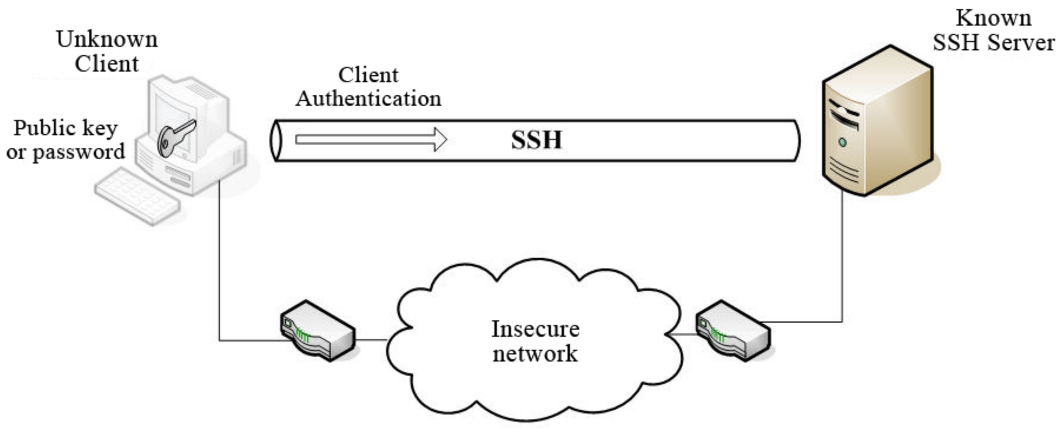


The negotiated parameters include KEX (*Key Exchange*) algorithms, the server key type, possible encryption algorithms, algorithms available for MAC error detection (*Message Authentication Code*), compression methods, etc. If both sides reach an agreement, key exchange begins. Here, the server is authenticated and generates an encryption key to be used for this session only.

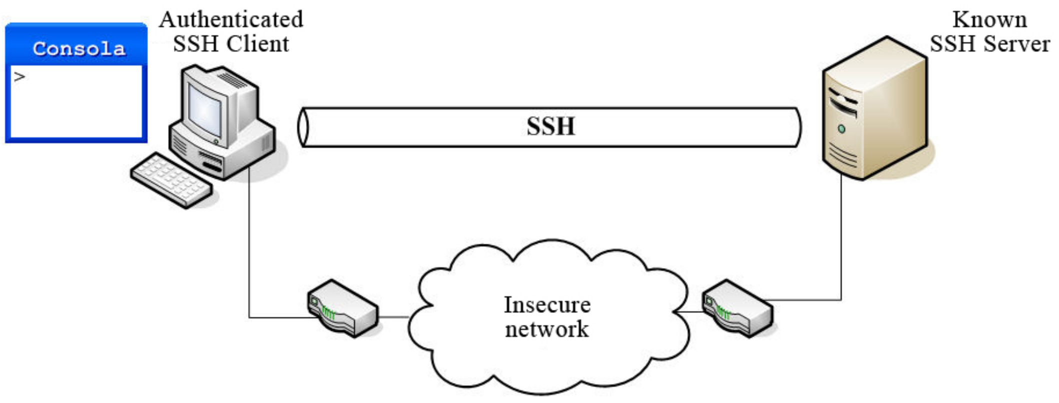


This key is used to encrypt all communications until a new key exchange is considered appropriate (for reasons related to time out or sent traffic). From this point on, the connection between client and server is deemed to be complete and confidential.

Once the connection is secure, the client tries to authenticate with the server using one of the methods permitted by the latter. To prevent unwanted client authentication, the server limits the authentication attempts and their execution time.



Lastly, when client identity has been checked, the client tries to access the group of services provided by the server over the secure connection. The server is responsible for giving or denying access to these services (depending on the client's access privileges or on the desired functions).



1.2 SSH Server

An SSH server provides a secure network connection for an SSH client. At the very least, the server provides client access to the server via a remote terminal or console (known as Secure Console). The protocol specification currently offers a wider range of optional services, such as file transfer over SSH (SFTP) or *Port-Forwarding* (which provides a secure tunnel to exchange data with other applications).

As for protocol versions, the server can be compatible with SSHv1, SSHv2 or both. Even though we recommend using SSHv2 due to the security risks detected in the first version, there are clients who only support SSHv1, forcing backward compatibility in many SSH services.

1.2.1 Server Authentication

To authenticate the server, you need a key (*host-key*) that uniquely identifies it in the network. Depending on the supported SSH, this key must be of a specific type. For the first version, the server needs to have an RSA key, known as RSA1 in this case. For the second version, the key must be RSA or DSA (*Digital Signature Algorithm*), and you can negotiate with one or the other. The server can have a key of each type to grant access to clients who only have one specific key type.

These keys consist of a public and a private part. The public part must be distributed among all clients wishing to start an SSH connection so the server they want to connect to is verified as the correct one. The different client distributions found on the market store this information after a connection with the server has been initialized, recording its IP address and port. At this point, they report the server is unknown. The client is responsible for verifying that the received public key is associated to the required server. In subsequent SSH connections with the server, the client checks that the public key received is the same as the one stored. If it isn't, the application reports a possible security breach. Once again, the client is responsible for deciding whether to trust the new key or not (recognizable by its *fingerprint*).

Regarding the private part, the server must zealously protect this to prevent another “malicious” server from taking on its identity. Although the method to store the private key depends on the implementation, the information on the public key is commonly stored with the private. The key is stored encrypted to increase confidentiality.

1.2.2 Client authentication

Once a secure communication has been established, the client must authenticate with the server. As a result, the latter must register access accounts for potentially permitted clients, together with their restrictions or privileges. SSH is not prone to granting server access to unregistered users (or those without a password). This is because, although the channel is encrypted, any client would then have access to the server and control would be lost over permitted users.

The server can set a maximum number of authentication attempts and limit the time available for these purposes. There are multiple authentication methods, all made up of a user name and another component (explained further on). Often, server implementation does not permit you to change the user name for each new attempt. Consequently, if you enter an incorrect username, you will need to restart the connection.

The following mechanisms can be found here:

- Password. The server checks the user and password fields provided by the client, which are encrypted, correspond to those registered; the associated permissions are assigned to the user.
- Authentication through the RSA public key for SSHv1. The public part of the client’s RSA1 is stored in the server’s configuration, so this can be identified once it starts to use the private part to create a signature.
- Authentication through the public key for SSHv2. This key can be either RSA or DSA, and the mechanism is similar

to the one above although the way the key is saved is different.

The client has the option of choosing which authentication method to use among those the server offers, and may need to use more than one.

1.3 SSH Client

The SSH client configuration is, depending on the implementation, very significant. While the server restricts the possible options for the negotiation parameters (encryption, authentication methods, etc.), the client can select from these options based on a priorities system. E.g. a server permits compression, but the client is configured so it doesn't use it and the client has preference over the server. The main parameter to be configured is the SSH version used to establish the connection.

For server authentication, the client saves a register with the public keys from all SSH servers it connects to. The first time, the user is warned the key is unknown. If, the value received the next time connection is attempted does not match the one stored, a potential security risk is reported.

When it comes to client authentication, any mechanisms allowed by the server can be selected. In the section focusing on the SSH server, some of these standardized authentication methods have been mentioned. A user name is required first, and this is normally not modifiable on successive authentication attempts. For the given methods, the client perspective is:

- Password. The client application stores the password used to authenticate with the server or asks the user each time authentication is executed.
- Authentication through the RSA public key for SSHv1. The client saves the route to the file with the RSA1 private key in its configuration. In order to use this key to authenticate, the client must have previously configured the server by registering its public key. During this process, if the key is password protected, the user will be asked to enter it if required. SSH clients however, usually have an associated application to generate keys. Moreover, thanks to agents, keys are often already downloaded and *passphrases* do not have to be entered.
- Authentication through the *public key* for SSHv2. As with the RSA1 key, the client must have the route to the file with the private key stored (although the format varies depending on the client). The most common formats are: OpenSSH (OpenSSL PEM format), ssh.com (commercial brand) and Putty. It's still necessary to configure the public part of the key in the server so that it is accepted, as well as entering the password at the execution stage (where required).

Finally, if the aim of the connection is to open a remote console, the client must create a terminal so that the user and the server can interact.

Chapter 2 Configuration

2.1 Accessing the SSH configuration menu

This chapter describes the steps that need to be followed to configure the SSH protocol, starting with how to access the protocol menu. The following sections describe the configuration process in more depth.

Firstly, enter the device configuration menu. For cases where you wish to modify the static configuration:

```
*config
Config>
```

To access the SSH configuration from the *Config>* (static configuration) or *Config$* (dynamic configuration) prompts, enter the following command:

```
Config>feature ssh
-- SSH protocol configuration --
SSH Config>
```

2.2 SSH protocol: Main menu

This section lists and describes the SSH configuration menu commands. Currently, we only have an SSH server that requires some keys from the device (*host-keys*). Since these keys identify all the equipment (and not just the server), they can be found in a separate command. Below, you can see the list of commands available in the main protocol menu:

Command	**Function**
?(HELP)	Lists the available commands or the options associated to a specific command.
HOST-KEY	Allows you to generate or insert device keys.
LIST	Displays the values for each of the configuration parameters, including the device's private keys.
NO	Clears the value configured for a parameter, leaving it with its default value. In cases involving keys, these are completely eliminated.
SERVER	Enters the SSH server configuration menu.
EXIT	Returns to the *Config>* prompt.

2.2.1 ? (HELP)

Lists the commands that are available at the layer where the router is programmed. You can also use this following a specific command to list the options. This option is available for all submenus and commands.

Syntax:

```
SSH config>?
```

Example:

```
SSH Config>?
  host-key    Host key configuration
  list        Display protocol configuration
  no          Negate a command or set its defaults
  server      Server configuration
  exit
SSH Config>
```

2.2.2 HOST-KEY

Through the **HOST-KEY** command, you can generate new keys or insert previously generated keys. The following subcommands here are:

Command	**Function**
DSA	DSA *Host-key* (for version 2).
RSA	RSA *Host-key* (for version 2).
RSA1	RSA1 *Host-key* (for version 1).

The device can only have one *host-key* of each type, although its existence is not always mandatory. If the server only needs to be compatible with SSHv2, it won't need an RSA1 key. Likewise, if it only needs to be compatible with SSHv1 (not recommended), it won't need DSA or RSA keys. When the server version includes SSHv2, you don't need both version keys (RSA and DSA); consequently, in the majority of cases, only one *host-key*, RSA or DSA, will suffice. The reason for having both is to permit interoperability with an SSH client that needs the server to use a specific key.

By default, the device does not have a *host-key*, however when the SSH server is enabled, the minimum keys needed, depending on the SSH version, are automatically generated. If the server is configured to operate in both SSHv1 and SSHv2, keys RSA1 and RSA with 1024 bits are generated. If only one version is enabled, the corresponding key is generated. By default, key DSA is not generated. In cases where the user deletes the configured *host-keys* while the server is enabled, when the server starts up (on device reboot) it tries to load the keys. If it doesn't find any of the required keys, it generates them automatically. At this point, the user is responsible for saving the dynamic configuration if he wishes to keep the keys.

A *host-key* is made up of a public part and a private part. All the information is saved in the *host-key* so that the public part is found together with the private part. Although the public part is freely distributed, since any client can initiate an SSH connection with a public key, the private part must be protected. Consequently, the *host-key* content must not be divulged and has to be protected through encryption.

Host-keys are saved in the configuration file so the device always keeps the same ones. If one of them is modified, the clients who registered the public key from the server report this indicating the key doesn't match the one saved. Although the keys are saved in the configuration, when the user uses the **SHOW CONFIG** command they are not displayed as they don't contain any relevant information and should not be moved to different devices. Keys must be unique for each device (so that they can be clearly identified). Even so, mechanisms exist to store the keys from other configurations.

To display the content of the *host-keys*, use the **LIST HOST-KEY [TYPE]** command. To enter a previous key in the device, use the **HOST-KEY [TYPE] INSERT**. Finally, as already mentioned, *host-keys* are also saved in the configuration file. Consequently, if this is copied in various devices, you must generate individual keys for each of them.

In every subcommand of each key type, the following options can be found:

Command	Function
GENERATE	Generates a *host-key* for the indicated type.
INSERT	Allows you to insert a previously generated *host-key* by pasting text.

The process of generating keys requires a time period that varies depending on the number of bits, the algorithm used and the device. In the worst case scenario, it can take several minutes. However, key generation should not be executed more than once per device. Once the *host-key* has been generated, the public key and the *fingerprint* are displayed on the console. The *fingerprint*, which results from executing MD5 over the public key, should at least be given to future SSH clients so they can visually verify server authenticity.

2.2.2.1 HOST-KEY DSA GENERATE [ENC_ALG]

Generates the DSA *host-key* (also known as DSS) with 1024 bits for SSHv2. The length is set and must have 1024 bits. If the device already has a key, the old one is overwritten.

Syntax:

```
SSH config>host-key dsa generate ?
  des-ede3-cbc    Select this encryption algorithm
  aes-128-ctr     Select this encryption algorithm
  aes-192-ctr     Select this encryption algorithm
  aes-256-ctr     Select this encryption algorithm
  <cr>
```

As already mentioned, the *host-key* is encrypted to increase confidentiality. By default, the *host-key* is encrypted using the *AES-256-CTR* algorithm (but we can select another one when we generate it). The algorithms available are: *DES-EDE3-CBC*, *AES-128-CTR*, *AES-192-CTR* and *AES-256-CTR*.

Example:

```
SSH Config>host-key dsa generate aes-128-ctr

Generating public/private dsa key pair...
Please wait for a few seconds.
Key generation done.

        Public key:
        ssh-dss AAAAB3NzaC1kc3MAAACBAIJZ7oaCXoGwWpRQlswnui+5V0B6VyfS5/xm
        XfZNhFZn4wf8Gl0w/xdW10v4D7jbZn4uuuNxlrL1Eo2oNdxf+JDvBU4oG+ho/G7m
        CS5YIVuvo/k3GEf0pupyP5T5Fn+xar4Z4nk20BQpX/kmD005RJd+dax/7GIWuGyI
        9QR2BocFAAAAFQDDrFReLdnYcW9lTm8aW7AtubvEEwAAAIAlk0YK77cl7KeEQ9eA
        PNyraj/v/N/gv1RelmFs9SwFguAB6jloKcjo+0P4Vf2mk/L9etymiWcFoWRf1hnK
        aIRw5Y/23Qps0n+hPyRI1qN6PVRNnBJ2Wm7TJku1RuCgqz28Q0BpKrW3ZbQiO8Vz
        2AboXpbHQlD18P7GjI9e1o1DEQAAAIA/rXRWWyiDUbcnc643mGEmL6PPp/nHlrll
        DTIZwuTrJ4uyazS90tSc7GSI5hoT88k4ErvCKPYHjkNC79oYD/hZeaDEib2aYyze
        jIeOfyw/nLv2xmYN2mEV2al0cKnby1iChNUKg5ylzYkfsPHwM+s/xYMS9Y8dQ085
        VmqPLF4EYQ==

        The key fingerprint is:
        4c:79:78:1d:c9:46:8a:61:3d:ed:bd:b3:07:17:18:5f

SSH Config>
```

For further information on the private key, please see the **LIST HOST-KEY DSA** command.

Command history:

Release	Modification
11.01.09	The possibility of choosing the encryption algorithm for the *host-key* has been added. The algorithms available are: *DES-EDE3-CBC*, *AES-128-CTR*, *AES-192-CTR* and *AES-256-CTR*.
11.01.05.40.07	The possibility of choosing the encryption algorithm for the *host-key* has been added. The algorithms available are: *DES-EDE3-CBC*, *AES-128-CTR*, *AES-192-CTR* and *AES-256-CTR*.
11.01.05.70.07	The possibility of choosing the encryption algorithm for the *host-key* has been added. The algorithms available are: *DES-EDE3-CBC*, *AES-128-CTR*, *AES-192-CTR* and *AES-256-CTR*.
11.01.06.53.01	The possibility of choosing the encryption algorithm for the *host-key* has been added. The algorithms available are: *DES-EDE3-CBC*, *AES-128-CTR*, *AES-192-CTR* and *AES-256-CTR*.

2.2.2.2 HOST-KEY DSA INSERT

Allows a known DSA *host-key* to be inserted in the device. Our device must have generated this. Otherwise, while no error message is displayed for the user, when the server tries to load it on start up, the key will be considered invalid. At this point, a new key is generated and the user notified through an event.

Syntax:

```
SSH config>host-key dsa insert
```

Example:

```
SSH Config>host-key dsa insert
Enter the host key (PEM format)
<cr> to escape
        -----BEGIN DSA PRIVATE KEY-----
        Proc-Type: 4,ENCRYPTED
        DEK-Info: DES-EDE3-CBC,82AECC6A19793195
        hU6tMIxW/GDAuugsnC++ROI3NDZcGNIlJFaoXhrhYuWHINXSXwihfKT0UgBtG7F2
        FutmdkEYAytkog5MLDK0+DCEmWs/9M/UkRmcYlYkJachv+UWvgDpMvfbVk50zRo/
        X3Oq6TAuT4voArLcb71GOtqHQKEeGi1lhUy3UffkN0WJYbqoImFzMa8cQTVaDDvH
        4DwnHSOlLWUIy/dCJOJiz0FuGNVqmp0Y5mZthcG7LdFWoHoQrlsa3+El0191/zAM
        Yq3yMkvUkp4KrxDkOA5jvHJGu/futCXq2dvGatQog1041DAqVZKiRT8RHVr8xK0m
        oSglm3KpW13rxpwfCAFbgHJMXPX447zr/tXOKS9P0Srl/YW0msbwRdw1uRnHKbWH
        bO6XtfJld98Z2QdUcxrMdc9dvxtDqAF9q84TjxqofrX+ZB/q83iYtcBR55IFh3eh
        NrBZgivEw5ZOJxIhZ0stBafsxn8hBloDdReVzxbyODfChLUKKtID8njuV1AH/O1j
        3rWrer7nZnaquvRDGvYw+UBGrZL6X/0el0lbMjjJvHegqg1KLRT/Vj3lNqiPWnMz
        gRBMIws4n1IkH138mm83YrKEOq3Agaa4
        -----END DSA PRIVATE KEY-----
SSH Config>
```

For further information on the private key, please see the **LIST HOST-KEY DSA** command.

2.2.2.3 HOST-KEY RSA GENERATE [NUM_BITS] [ENC_ALG]

Generates the RSA *host-key* with the specified number of bits, between 768 and 2048, for SSHv2. In most cases, a length of 1024 bits should be enough. If the device already has the key, the old one is overwritten.

As already mentioned, the *host-key* is encrypted to increase confidentiality. By default, the *host-key* is encrypted using the *AES-256-CTR* algorithm (but we can select another one when we generate it). The algorithms available are: *DES-EDE3-CBC*, *AES-128-CTR*, *AES-192-CTR* and *AES-256-CTR*.

Syntax:

```
SSH config>host-key rsa generate <num_bits> <enc_alg>
```

Example:

```
SSH Config>host-key rsa generate 2048 des-ede3-cbc

Generating public/private rsa key pair...
Please wait for a few seconds.
Key generation done.

        Public key:
        ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAxmh+MS0PEDcy+56ej6KQm+pPoXyD
        aNO7cuwnOJlhcmNXZdqwoPIhXx4OgVSuEEQVnWHZT0fEVflFzZUpSOdk8MkwvHjO
        u+YOjN8EO/azyPhfru7SLYuFdm6ZIyES4GahkoBQIFiQ5hMSsAQEuqyXVpuKGkWg
        gy/OFA63Ft3VAc7+VDNJZoJZpsY2wJb2hxqSEcTL4ZwCMQr9Q3515bQzohutKHX5
        3E7vWnbm1VmSLq8HLjSedUvds3u5pWvQ/JHBQAs+YGxIHaP+lYxFhW3Tk/UXwzXu
        c8PPv2bLZwnT+szMfa4q4x7qCs0ozwTwNzTb9K6Bb8H31YbD5pOslBhsVQ==

        The key fingerprint is:
        ba:4a:de:ff:40:9f:a2:3a:15:08:62:ef:f8:ee:73:39

SSH Config>
```

For further information on the private key, please see the **LIST HOST-KEY RSA** command.

Command history:

Release	Modification
11.01.09	The possibility of choosing the encryption algorithm for the *host-key* has been added. The algorithms available are: *DES-EDE3-CBC*, *AES-128-CTR*, *AES-192-CTR* and *AES-256-CTR*.
11.01.05.40.07	The possibility of choosing the encryption algorithm for the *host-key* has been added. The algorithms available are: *DES-EDE3-CBC*, *AES-128-CTR*, *AES-192-CTR* and *AES-256-CTR*.
11.01.05.70.07	The possibility of choosing the encryption algorithm for the *host-key* has been added. The algorithms available are: *DES-EDE3-CBC*, *AES-128-CTR*, *AES-192-CTR* and *AES-256-CTR*.
11.01.06.53.01	The possibility of choosing the encryption algorithm for the *host-key* has been added. The algorithms available are: *DES-EDE3-CBC*, *AES-128-CTR*, *AES-192-CTR* and *AES-256-CTR*.

2.2.2.4 HOST-KEY RSA INSERT

Allows a known RSA *host-key* to be inserted in the device. Our device must have generated this. Otherwise, while no error message is displayed for the user, when the server tries to load it on start up, the key will be considered invalid. At this point, a new key is generated and the user notified through an event.

Syntax:

```
SSH config>host-key rsa insert
```

Example:

```
SSH Config>host-key rsa insert
Enter the host key (PEM format)
<cr> to escape
        -----BEGIN RSA PRIVATE KEY-----
        Proc-Type: 4,ENCRYPTED
        DEK-Info: DES-EDE3-CBC,6A1846834A4DA3FC
        ++QKlzwaKAL8u9G/eYkutMjQPpBIZoV5eQMxNfFzWRlE47T1myBRqFrCFvDXiI4G
        1ZMrmvbua0S1GUBZCcGkNpJHk0jP7/AYNMIW1XCHnKPafZ2DkquAZ5BU/IhY4ztz
        /GZLyVB+AULhG1rQil9P82cqqGp5Cv4VkZdwRJHnLLslij5B5ToCO/ap6krtsCUH
        rEXfRXf3kXkBvwsNnAve7DMU3EXWzq3TS//u4NRPeVNEVAHp6zooPZhp8LzHPpxh
        E2ki/h6yBW0+P8/Rzg1jWV8oreaDJfFzPvzhrOdWN4aNJSXZbSXL1hwYdWgouJ+u
        doaCX1YZzrLLdGcasY3kmSC3N9OwalFnc4i2xyN+lgM/HbYmoljJPJfMV0RCK5S0
        KljWhlk4oiRLcmEzIM4+1UjCPP1VWTanxCMmkCMfgmtjun+Dcfq3JGz34xN3GNap
        e2voWH3vwG196yXmOlHLF0/WsJgQuFkBu1/rrDx65ztNsKumoAAbCEAz5fkHf1T1
        2UlZu347I+A4FDUw7/VOluJu9Ey1ffBfQPvc+7Y0IDvxVTkiAHk5ChtUjoWMFCsz
        iU3I794EJ/YgFoldkg1DT+07pmQTvzTldv078hXKzkWa8JKc+GbE7u08bHVWYV/E
        7P8wrwP4tXjH22zRyycS7uRvsX3859A29KwQxFU1fX264SjmLBwdFY6GaLURBIvV
        OLUxdjnFaDtcsTFjbvxT6/FgWk2dDVrxJJOl+bfxGLgNpFQk16SYJsQwM7LtFQsW

        7LkXuuZIwOmTxI1mX7jdGW4VTp7sm8anmRCGCNpEz7n7yyNHKf9KR0rsWZw5sfiN
        I8opvHfl6ZFpBBweLLtByl5og0WWrbjPcNMNRYPCnKLCP1F9ph+271Mf3PBXMTaj
        RshZboZLKJqfOE/LYSJjjC3IgHngC4goSd0+4PT0nlCmAW89lrmMKmuUAG2utELy
        40IMg8VYAWjZ+sdz+XDgLOKOqud2L7RnsMED67V7ZgrFTWF8AsLwMNmcouLzHNE3
        d5DUD7c9MVSISYQvM+XaXo2BE9TzeT7T4xvD03fiQxrv9ncrkwUHutwEENbjOmLL
        3fW8bZeNTBudy4j/cJKtLuqHtbrNJNasTcgEicR5G80+Vdjoy7qpd6+zYn5zlSZt
        fxDoIC3AFX+4BG+iS782dx1moA6IZJjnrn4BxPjPu+iP/Db0XTvn5Q41Z8LRx6Dq
        XtEFG6i3NTLy84VDKL4V+3wpazrV+tSrLd9KNypfDGJHcBF8gSeJnoq0qP9RaH9U
        b14qVHCfOP/CMOJeBxMPvz6WHZklpXpaHhCudkHnbABrBYGZ1L1QaPL4hdVtin0b
        T/l+nPV2HIeYm7nWaZJHp5wtQ8x6hwaW9++AQCpVsWoufbfamipAwfJpet3kkBUI
        x0Ms5Dg9jzFluR/U2YoQuK2KnXsjmKtmt3f2UuvWrzNSWOAJlGn/H9n1A9UNmjcy
        +P+ZYO/TirddJockBb/Yy0xDb9nk9zTn5Fyl98H7/UPRPwF2wXgjGHsCR06Y81UV
        0Pu0n5lA8d31UXNW00hmC/o9I5xOkn9yz2RBIZ6tH1g0VaDUx2+ZdQ==
        -----END RSA PRIVATE KEY-----
SSH Config>
```

For further information on the private key, please see the **LIST HOST-KEY RSA** command.

2.2.2.5 HOST-KEY RSA1 GENERATE [NUM_BITS]

Generates the RSA1 *host-key* with the specified number of bits, between 768 and 2048, for SSHv1. A length of 1024 bits should be enough in the majority of cases. If the device already has the key, the old one is overwritten. This is known as RSA1, even though it is still an RSA key, as the format of the *host-key* varies both when saving it and when extracting its public part.

Syntax:

```
SSH config>host-key rsa1 generate <num_bits>
```

Example:

```
SSH Config>host-key rsa1 generate 1024
Generating public/private rsa1 key pair...
Please wait for a few seconds.
Key generation done.
        Public key:
        1024 35 160578883652656818657287903682622114743386532612860086 75
        36281608085125521320510295982789525407462525286298156177349029 17
        88215443088250014201492858699431905211166084332383768689534537 49
        70510771467680555253701428302516535208884645913019269801734049 70
        0790695642285984523750486329852429527057770945907085394177099
        The key fingerprint is:
        a2:94:6a:b9:39:d5:75:73:5a:f4:d8:cd:35:3c:3e:08

SSH Config>
```

You can see that the format of the public part is different from the RSA part (SSHv2). In this case, this is made up of the module length, followed by the exponent and module, all in decimal. For further information on the private key, please see the **LIST HOST-KEY RSA1** command.

2.2.2.6 HOST-KEY RSA1 INSERT

Allows a known RSA1 *host-key* to be inserted in the device after it has been generated. Otherwise, while no error message is displayed for the user, when the server tries to load it on start up, the key will be considered invalid. At this point, a new key is generated and the user notified through an event.

Syntax:

```
SSH config>host-key rsa1 insert
```

Example:

```
SSH Config>host-key rsa1 insert
Enter the host key (base64)
<cr> to escape
        U1NIIFBSSVZBVEUgS0VZIEZJTEUgRk9STUFUIDEuMQoAAwAAAAAAAAAQABADkrAMA
        VHLs6btQpuo/JU3e0t4ZGqaxdeUmNk23hKFzfXGYVOMeA1qvSBvHlnvTHKg7oIah
        Of6zkTofLnUW3i+okfnAqK2XRJ98woh/enW8Fj6DKk9Pd4Ek1KnuKn8BWPEOU+dJ
        7caGmgbfQj1t2crgcqiAYdjGN6LsToT9ytnsSwAGIwAAAA11bmtub3duX3VzZXJA
        h18uq51TTPQG+P9u7O/fU69F2EGdS5k28PW5wPExDPyiHWGz+D1Z/uHV51XBT+3K
        CMjnrWPRaiXvnDwAJ1B+isEtUuke1kykQZ6pSTppivQj6oynNPxGSQPcCJqaxLy4
        0/9g+tRZW0Kfz+/ELXtuzXl7cc1VZi9uVgFdjCqi+qRbUxmcS/QpgpbJTph895MZ
        zxVNGSOoR6iA32tyNcJlanpRauN7xAipbrNEQflVZu+H0k67EUb3T1JQ+jJmrZC+
        nsB0iBOgXo6s2KM+TdYxU+NQwa7DH0Ag6UJczAvbj+KhofTGIOskztC8Z+k4aVuz
        8YvMprTjkoJ+dcjAatCttyGZDMeM9xdYbNQyHAsyXLCvRojB7laS1yG6lZKyUgC1
        E3rm52XicGkN5+Yl4pWUZcKdeEUO7GOVn/erkXAcnNaEYUGyKkxLc2Su8Hdhn7tT
SSH Config>
```

For further information on the private key, please see the **LIST HOST-KEY RSA1** command.

2.2.3 LIST

Displays the whole SSH configuration. It has various subcommands, allowing you to focus on specific configuration sections. The "*list*" command does not show the changes made in the default configuration (use the **SHOW CONFIG** command instead). However, this command lists the protocol configuration parameter values (be they default values or values modified by the user). The following subcommands are available:

Command	Function
ALL	Lists the whole protocol configuration.
HOST-KEY	Displays all the *host-keys* in the device.
SERVER	Displays the SSH server configuration.

You should see that, depending on which configuration menu you execute the command in, either the static or dynamic configuration is shown. On starting the router, both configurations match. The exception comes up in cases where the server is enabled, there are no *host-keys* saved in the configuration and (on not finding them) the server generates them and saves them in the dynamic configuration. The executing server uses these keys and must save the configuration in order to keep them (and also to prevent the device from having to generate them again each time it starts up).

2.2.3.1 LIST ALL

Displays the information on protocol configuration. This is similar to executing a **LIST SERVER ALL** command followed by **LIST HOST-KEY ALL.**

Syntax:

```
SSH config>list all
```

Example:

```
SSH Config>list all

SSH Server configuration:
        Server status: enabled


        Version compatibility: SSHv1 and SSHv2


        Listening port:    22


        Payload compression: enabled


        Ciphers:
                    3des-cbc : available
                  aes128-cbc : available
                  aes192-cbc : available
                  aes256-cbc : available
                  aes128-ctr : available
                  aes192-ctr : available
                  aes256-ctr : available
                  arcfour128 : available
                  arcfour256 : available
                     arcfour : available
                blowfish-cbc : available
                 cast128-cbc : available

        Message Authentication Codes:
                    hmac-md5 : available
                   hmac-sha1 : available
              hmac-ripemd160 : available
                hmac-sha1-96 : available
                 hmac-md5-96 : available
               hmac-sha2-256 : available
               hmac-sha2-512 : available

        KEX Algorithms:
                     diffie-hellman-group1-sha1 : available
                    diffie-hellman-group14-sha1 : available
             diffie-hellman-group-exchange-sha1 : available
           diffie-hellman-group-exchange-sha256 : available

        Authentication methods:
          password : available
          public-key : available
          rsa : available

        Maximum number of authentication attempts:  6
        Maximum time to complete authentication:   2m0s
        Maximum number of SSH connections:  4

        Keep-alive activated: yes

        Client-alive message parameters:
                Maximum number of messages sent without response:  3
                Interval between messages:    15s

        Ephemeral server key parameters:
                Number of bits:   768
                Interval to regenerate server key: 1h0m0s


SSH Host keys:

    RSA1 (only for SSHv1)
        Host key:
        Host key not found

    RSA (only for SSHv2)
        Host key:

        -----BEGIN RSA PRIVATE KEY-----
        Proc-Type: 4,ENCRYPTED
        DEK-Info: DES-EDE3-CBC,38D93C62DCD02657

        VCBNgLYGsHsAPI8r3HD12lcE3cUSOgQAf8CKCpOG8Hhu3S+ziCDPmvLrYQdjXk0i
        3spQeBW+MvuoS4EA9XiMU8/NoSLjVS/k8aYSArHF8g5jKT+aRjTsO+akm+6yWILa
        ylUTh29H0xTLf1sYg9oBR//7zhuz6vEEjFtqDXzOLO30Izn2Hpdzh2v4FAJaXO40
        /Web1IUDWNzdcKQU78wHy+M+3gmgfj/O75X68UGphLJRnHvf2tJcAB5l3gALdMuq
        ffpT/WMMXS9TgE1IWJMEI7m7L+fUWtlXN6FAxQ4rG5B64PKybNFRAjTRnz3G0mkn
        zvi+4UoB5kG+emO2Uohxs5/kTHM4cSSE0qghCV3m5VYgkdnj0lRaBCm3s8qiVJhi
        8WE9sMxsPKcurxQTLppeXpDHxOXGAJRAUR4yP9U2qnODz62aVKM3h/anC5NfT41C
        oZMUIteFB8HUkRHLeSdZkCx4ePv6GDzd+n2dh/iiThG1E+a15PICMJIwBzowJtvb
        kZ8OoAWm7zpMlcBxYnIPrKUtFeOoXU6irHrlCAw+NN0WVr+LU9SeLhQx42F22YfV
        KIxBQoluwyserVG2lWdd6bGbm7Wj70PvXyYHzsu8ka2KdOVjAsBpJbDABf41GBbv
        6Tzd+71GeL96l56he1fm5UojOQK+G1fNvHgnxfA0WJVfMnvzUwlkamJ5KwYwdc8q
        pc2MkWnJS0DyHuQ7GYOMV8mxT3njJli4yHoZ1fTbhqYfg+RoZUi3FdM6+bVHtBMM
        ZZ/YcodRqQwlFSW/MR3N2PiD6uFbOQLV/0stPc+tw5KDLJ4bqHm4cPyRhgdSDdgj
        pT1q6ydpJZywBU6rntwwbE94zzkOR1m9DE1g5tV2v1v7rX4OlBYXTJAdM16I/6XS
        4qIyLwuCGVA4/+qm40TlWwjxnZlk8mNLgUId6z/GkLNyaoqL251BEzAjV7DNVfFQ
        2KSF4GVuc59gd6op6HlNIgl3nbwUqGVtrdpFjnOhdeK+/pYGj0GQQ0naM+rw3qka
        eLk0yH9LENu3LcdjDoMkON+ULzoByZHVgrsFC04PbjXDFq4ktoOYf68PWLs1J5xa
        IwhLzjPZUDxnGn46Ov92AZFudD40wK6nM83jYoTA8PggI4MaTfS4PeaN5CTUdUBY
        3rxqe7rtNRLUIYWzp2FJJIP41QxqP5z10ERDd27S7rkDDWzsY9cJJ0TllcXhX3U7
        wy4Vrc17z+WByE6BCuIUC++xSdXXfRXXa0crg/vxMAU+HnnZe/raBHe17bai0XbW
        rUlCap0O8ZnzQFWlJZUJwhOWn+XnQfzb7wl4fXobmWIyPFxtpDGcH3iEnK6p5c7G
        qQhE3zs569C4PUllrfsNF5rrVXidpez4UGwTnKfKe/b3sYo6LH0faBx0nv+DM5Kq
        3DSrhnjFcwjXFwjRl6HaoCaR8Jz5k6IZ6m7Klls/dkYH4BYyVlrgNXimB9c6XaPs
        wuRfGmt7p8UTxNTUp84iI50hpeBB5Skz3jHi1BIg2N1gTfrNfp7jc7zVaBH+FKDy
        5PfeRwTTi3xnFBbSrZdJGJL3hFeM3Ua5HuLdz+LjCqlMZpmbSnxz7w==
        -----END RSA PRIVATE KEY-----

    DSA (only for SSHv2)
        Host key:

        -----BEGIN DSA PRIVATE KEY-----
        Proc-Type: 4,ENCRYPTED
        DEK-Info: AES-128-CTR,FB40AE9B5A4C0D1D2227D2250710AC30

        Qpsf7Q8NNjLVS8DkFo06PY9nxo4U+r+yqHXdz9M8Gm1d3SAnYoQqgeBn9kYVIWKR
        qgNKXlUTXlcD+malHXMXXe0De4DPoW25EZtDrZjv/7e046am+maKg2OVjaVz/QRm
        Xml+fA+tON1ArxsynOAASw2q82ONPA12/3eEPa2D2vziRbzA9FGKcfvP7nJBWt4s
        RsuZmC3LbKEX5R6jiXJZ4uePl6K4R+HfmNN+WzPdzrivBLdvJJdJRvA6VbFuKxMP
        z6fc3M3Nh9iBSRCJJwSSCGU5Sl4IR3Ip4PwjWQdu6tS44GKfeaEgntC/9pBvZFej
        ceEWSf1+YH/lFCznh30bFXmJ3IxB5+pl07I1mNsej0E2vUr+7fY93Wx4Tfn8YZ87
        uMOk+4KgebqMkz/r7g8bNdlrvROKMBszXfAkVVl0HrMt/7+gz1tKtCZIFqtKFV87
        ba4sepryIiMVqDxlbu6hJdwZtcGc3ZjjXA4xMV2tKSFACVIxJMFsZ4Vyl5dkwqV9
        gis3ZhnyoQhDRsNPfe7fCT8vkEU8lvbBkg0p5FNmtsJef2ditIlIbNv17BoM7+Ms
        NWaiIepVBie1ixNWly4Lyw==
        -----END DSA PRIVATE KEY-----

SSH Config>
```

Each outcome presented using every **LIST** command available is described.

Command history:

Release	Modification
11.01.09	The command output has changed as of version 11.01.09. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.05.70.06	The command output has changed as of version 11.01.05.70.06. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.05.40.07	The command output has changed as of version 11.01.05.40.07. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.06.53.01	The command output has changed as of version 11.01.06.53.01. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.09	The command output has changed as of version 11.01.09. The possibility of selecting the key exchange algorithms to generate per-connection keys in SSHv2 has been added on the SSH server and this command shows their availability.
11.01.05.40.07	The command output has changed as of version 11.01.05.40.07. The possibility of selecting the key exchange algorithms to generate per-connection keys in SSHv2 has been added on the SSH server and this command shows their availability.
11.01.05.70.07	The command output has changed as of version 11.01.05.70.07. The possibility of selecting the key exchange algorithms to generate per-connection keys in SSHv2 has been added on the SSH server and this command shows their availability.
11.01.06.53.01	The command output has changed as of version 11.01.06.53.01. The possibility of select-

Release	Modification
	ing the key exchange algorithms to generate per-connection keys in SSHv2 has been added on the SSH server and this command shows their availability.

2.2.3.2 LIST HOST-KEY

Shows the device's keys on the console. The following options are available:

Command	Function
ALL	Lists all the *host-keys* in the device.
DSA	DSA *Host-key* (for version 2).
RSA	RSA *Host-key* (for version 2).
RSA1	RSA1 *Host-key* (for version 1).

The difference between the static and dynamic configuration is more obvious in this subcommand. In the former, only the *host-key* is displayed (i.e., the one saved in the configuration). In the latter, in addition to the *host-key*, the public key is shown with its *fingerprint* (which is extracted from the *host-key*). This is only executed for the dynamic configuration, based on the keys loaded by the SSH server. This list shows the public key belonging to the *host-key*, which currently identifies the device.

2.2.3.2.1 LIST HOST-KEY ALL

Displays device key configuration. If one is missing, it appears as not found. This is similar to executing the **LIST HOST-KEY RSA1**, **LIST HOST-KEY RSA** and **LIST HOST-KEY DSA** commands one after another. In this example, the command is executed in the dynamic configuration (prompt ends in "$").

Syntax:

```
SSH config$list host-key all
```

Example:

```
SSH Config$list host-key all
SSH Host keys:
    RSA1 (only for SSHv1)
        Host key:
       U1NIIFBSSVZBVEUgS0VZIEZJTEUgRk9STUFUIDEuMQoAAwAAAAAAAAQABADkrAMA
        VHLs6btQpuo/JU3e0t4ZGqaxdeUmNk23hKFzfXGYVOMeA1qvSBvHlnvTHKg7oIah
        Of6zkTofLnUW3i+okfnAqK2XRJ98woh/enW8Fj6DKk9Pd4Ek1KnuKn8BWPEOU+dJ
        7caGmgbfQj1t2crgcqiAYdjGN6LsToT9ytnsSwAGIwAAAA11bmtub3duX3VzZXJA
        h18uq51TTPQG+P9u7O/fU69F2EGdS5k28PW5wPExDPyiHWGz+D1Z/uHV51XBT+3K
        CMjnrWPRaiXvnDwAJ1B+isEtUuke1kykQZ6pSTppivQj6oynNPxGSQPcCJqaxLy4
        0/9g+tRZW0Kfz+/ELXtuzXl7cc1VZi9uVgFdjCqi+qRbUxmcS/QpgpbJTph895MZ
        zxVNGSOoR6iA32tyNcJlanpRauN7xAipbrNEQflVZu+H0k67EUb3T1JQ+jJmrZC+
        nsB0iBOgXo6s2KM+TdYxU+NQwa7DH0Ag6UJczAvbj+KhofTGIOskztC8Z+k4aVuz
        8YvMprTjkoJ+dcjAatCttyGZDMeM9xdYbNQyHAsyXLCvRojB7laS1yG6lZKyUgC1
        E3rm52XicGkN5+Yl4pWUZcKdeEUO7GOVn/erkXAcnNaEYUGyKkxLc2Su8Hdhn7tT

        Public key:
        1024 35 160578883652656818657287903682622114743386532612860086755
        3628160808512552132051029598278952540746252528629815617734902917
        8821544308825001420149285869943190521116608433238376868953453749
        7051077146768055525370142830251653520888464591301926980173404970
        0790695642285984523750486329852429527057770945907085394177099

        Key fingerprint:
        a2:94:6a:b9:39:d5:75:73:5a:f4:d8:cd:35:3c:3e:08

    RSA (only for SSHv2)
        Host key:
        Host key not found

        Public key not found

    DSA (only for SSHv2)
        Host key:

        -----BEGIN DSA PRIVATE KEY-----
        Proc-Type: 4,ENCRYPTED
        DEK-Info: DES-EDE3-CBC,82AECC6A19793195

        hU6tMIxW/GDAuugsnC++ROI3NDZcGNIlJFaoXhrhYuWHINXSXwihfKT0UgBtG7F2
        FutmdkEYAytkog5MLDK0+DCEmWs/9M/UkRmcYlYkJachv+UWvgDpMvfbVk50zRo/
        X3Oq6TAuT4voArLcb71GOtqHQKEeGi1lhUy3UffkN0WJYbqoImFzMa8cQTVaDDvH
        4DwnHSOlLWUIy/dCJOJiz0FuGNVqmp0Y5mZthcG7LdFWoHoQrlsa3+El0191/zAM
        Yq3yMkvUkp4KrxDkOA5jvHJGu/futCXq2dvGatQog1041DAqVZKiRT8RHVr8xK0m
        oSglm3KpW13rxpwfCAFbgHJMXPX447zr/tXOKS9P0Srl/YW0msbwRdw1uRnHKbWH
        bO6XtfJld98Z2QdUcxrMdc9dvxtDqAF9q84TjxqofrX+ZB/q83iYtcBR55IFh3eh
        NrBZgivEw5ZOJxIhZ0stBafsxn8hBloDdReVzxbyODfChLUKKtID8njuV1AH/O1j
        3rWrer7nZnaquvRDGvYw+UBGrZL6X/0el0lbMjjJvHegqg1KLRT/Vj3lNqiPWnMz
        gRBMIws4n1IkH138mm83YrKEOq3Agaa4
        -----END DSA PRIVATE KEY-----

        Public key:
        ssh-dss AAAAB3NzaC1kc3MAAACBAMBl0OdknSG228cLtQ+6z/BwizJo2ijElXRI
        JkoLAFO0q+ACbA6fe8wo+9Hy4RjAyoO6HWtUXbuuO7fDFoIqOWLYLM0t5jNfI1g+
        yiezyrNyRFcffwslKezZ6XjV6CQESGX2zj+SQxWKeVx++FdsIx2NG4zaCRNmtTdX
        MDfLIRrFAAAAFQDod41UuyTQ9tIvdOF+tXvf5ZyzoQAAAIEAvYPpolQj6lbrhIhp
        q7U+b1SJTBQIruXRco11bym2O91kHM0EVIZm0ZyHaBvwkyEeCno8WCE6KI9X52XK
        ROow2Es45FqsvGR1sJleDyVxsjbLU4eRwHLLSgQ1ORdzTH1ic+oFpplaPZOmvZeu
        uCHoRnUne/jgJHaF4rSquBrn+aYAAACBAJu1xPY74hvhi1fy9L6HP5v5bu3lAxQK
        W2eH+zYONKhNHOrC2Xs8Mt7adhPDQRabXxtPA4PLwo0uVKuBcufvqaKLs/llzNaJ
        ghxwS1wR9W3IdAX4FQkJ6HZ7plQx1lT+ZwIA7JpZcKNChDwMPNM/iDHcoRO0+Kga
        9X12KiA5Bcew
        Key fingerprint:
        97:53:9d:25:21:8b:76:49:09:5d:e9:6c:4c:f6:8e:56

SSH Config$
```

In this example, the device doesn't have an RSA key (indicated as not found). However, the structure of the information shown is more relevant. Each key appears with the title "Host-key". The *host-key* is the content displayed in the lines up until "Public-key" (for dynamic configuration), without taking into account the blank lines at the beginning and at the end.

2.2.3.2.2 LIST HOST-KEY DSA

Describes the DSA key, where available. This example shows the command executed in the static configuration. In the dynamic configuration, it also shows the public key and the *fingerprint* (as seen in the **LIST HOST-KEY ALL** example).

Syntax:

```
SSH config>list host-key dsa
```

Example:

```
SSH Config>list host-key dsa
    DSA (only for SSHv2)
        Host key:

        -----BEGIN DSA PRIVATE KEY-----
        Proc-Type: 4,ENCRYPTED
        DEK-Info: AES-128-CTR,FB40AE9B5A4C0D1D2227D2250710AC30

        Qpsf7Q8NNjLVS8DkFo06PY9nxo4U+r+yqHXdz9M8Gm1d3SAnYoQqgeBn9kYVIWKR
        qgNKXlUTXlcD+malHXMXXe0De4DPoW25EZtDrZjv/7e046am+maKg2OVjaVz/QRm
        Xml+fA+tON1ArxsynOAASw2q82ONPA12/3eEPa2D2vziRbzA9FGKcfvP7nJBWt4s
        RsuZmC3LbKEX5R6jiXJZ4uePl6K4R+HfmNN+WzPdzrivBLdvJJdJRvA6VbFuKxMP
        z6fc3M3Nh9iBSRCJJwSSCGU5Sl4IR3Ip4PwjWQdu6tS44GKfeaEgntC/9pBvZFej
        ceEWSf1+YH/lFCznh30bFXmJ3IxB5+pl07I1mNsej0E2vUr+7fY93Wx4Tfn8YZ87
        uMOk+4KgebqMkz/r7g8bNdlrvROKMBszXfAkVVl0HrMt/7+gz1tKtCZIFqtKFV87
        ba4sepryIiMVqDxlbu6hJdwZtcGc3ZjjXA4xMV2tKSFACVIxJMFsZ4Vyl5dkwqV9
        gis3ZhnyoQhDRsNPfe7fCT8vkEU8lvbBkg0p5FNmtsJef2ditIlIbNv17BoM7+Ms
        NWaiIepVBie1ixNWly4Lyw==
        -----END DSA PRIVATE KEY-----

SSH Config>
```

The *host-key* has the OpenSSL PEM format, which forces it to follow certain guidelines. It must be delimited by the lines (header and footer):

-----BEGIN DSA PRIVATE KEY-----

and

-----END DSA PRIVATE KEY-----

The *host-key* is encrypted to increase confidentiality. This also forces the first lines to display:

Proc-Type: 4,ENCRYPTED

DEK-Info: AES-128-CTR, ...

This must be followed by a blank line and the context of the key itself, which is encoded in Base 64 as well as being encrypted (in this example) with the AES-128-CTR algorithm.

If you need to copy the *host-key* to another configuration, make sure you copy the whole thing including the delimiters (header and footer), and paste it in the destination configuration using the **HOST-KEY DSA INSERT** command. The presence of the blank spaces at the beginning of the line does not pose a problem, but you must be careful not to include tabulations.

2.2.3.2.3 LIST HOST-KEY RSA

Describes the RSA key, where applicable. This example shows the command executed in the dynamic configuration. In the static configuration, the public key or the *fingerprint* are not shown.

Syntax:

```
SSH config$list host-key rsa
```

Example:

```
SSH Config$list host-key rsa
    RSA (only for SSHv2)
        Host key:
        -----BEGIN RSA PRIVATE KEY-----
        Proc-Type: 4,ENCRYPTED
        DEK-Info: DES-EDE3-CBC,6A1846834A4DA3FC
        ++QKlzwaKAL8u9G/eYkutMjQPpBIZoV5eQMxNfFzWRlE47T1myBRqFrCFvDXiI4G
        1ZMrmvbua0S1GUBZCcGkNpJHk0jP7/AYNMIW1XCHnKPafZ2DkquAZ5BU/IhY4ztz
        /GZLyVB+AULhG1rQil9P82cqqGp5Cv4VkZdwRJHnLLslij5B5ToCO/ap6krtsCUH
        rEXfRXf3kXkBvwsNnAve7DMU3EXWzq3TS//u4NRPeVNEVAHp6zooPZhp8LzHPpxh
        E2ki/h6yBW0+P8/Rzg1jWV8oreaDJfFzPvzhrOdWN4aNJSXZbSXL1hwYdWgouJ+u
        doaCX1YZzrLLdGcasY3kmSC3N9OwalFnc4i2xyN+lgM/HbYmoljJPJfMV0RCK5S0
        KljWhlk4oiRLcmEzIM4+1UjCPP1VWTanxCMmkCMfgmtjun+Dcfq3JGz34xN3GNap
        e2voWH3vwG196yXmOlHLF0/WsJgQuFkBu1/rrDx65ztNsKumoAAbCEAz5fkHf1T1
        2UlZu347I+A4FDUw7/VOluJu9Ey1ffBfQPvc+7Y0IDvxVTkiAHk5ChtUjoWMFCsz
        iU3I794EJ/YgFoldkg1DT+07pmQTvzTldv078hXKzkWa8JKc+GbE7u08bHVWYV/E
        7P8wrwP4tXjH22zRyycS7uRvsX3859A29KwQxFU1fX264SjmLBwdFY6GaLURBIvV
        OLUxdjnFaDtcsTFjbvxT6/FgWk2dDVrxJJOl+bfxGLgNpFQk16SYJsQwM7LtFQsW
        7LkXuuZIwOmTxI1mX7jdGW4VTp7sm8anmRCGCNpEz7n7yyNHKf9KR0rsWZw5sfiN
        I8opvHfl6ZFpBBweLLtByl5og0WWrbjPcNMNRYPCnKLCP1F9ph+271Mf3PBXMTaj
        RshZboZLKJqfOE/LYSJjjC3IgHngC4goSd0+4PT0nlCmAW89lrmMKmuUAG2utELy
        40IMg8VYAWjZ+sdz+XDgLOKOqud2L7RnsMED67V7ZgrFTWF8AsLwMNmcouLzHNE3
        d5DUD7c9MVSISYQvM+XaXo2BE9TzeT7T4xvD03fiQxrv9ncrkwUHutwEENbjOmLL
        3fW8bZeNTBudy4j/cJKtLuqHtbrNJNasTcgEicR5G80+Vdjoy7qpd6+zYn5zlSZt
        fxDoIC3AFX+4BG+iS782dx1moA6IZJjnrn4BxPjPu+iP/Db0XTvn5Q41Z8LRx6Dq
        XtEFG6i3NTLy84VDKL4V+3wpazrV+tSrLd9KNypfDGJHcBF8gSeJnoq0qP9RaH9U
        b14qVHCfOP/CMOJeBxMPvz6WHZklpXpaHhCudkHnbABrBYGZ1L1QaPL4hdVtin0b
        T/l+nPV2HIeYm7nWaZJHp5wtQ8x6hwaW9++AQCpVsWoufbfamipAwfJpet3kkBUI
        x0Ms5Dg9jzFluR/U2YoQuK2KnXsjmKtmt3f2UuvWrzNSWOAJlGn/H9n1A9UNmjcy
        +P+ZYO/TirddJockBb/Yy0xDb9nk9zTn5Fyl98H7/UPRPwF2wXgjGHsCR06Y81UV
        0Pu0n5lA8d31UXNW00hmC/o9I5xOkn9yz2RBIZ6tH1g0VaDUx2+ZdQ==
        -----END RSA PRIVATE KEY-----

        Public key:

        ssh-rsaAAAAB3NzaC1yc2EAAAABIwAAAQEA0ayDS70qND+k4ZtyFs8LifniAHFR
        InD/Ygii0CoSj8YCXdr2e8ahEFmt0fvc9N53+blBDPGdo7cVBrC1BmY4ocWH+ZRp
        dpsTjD4mQ33ARJcvypyQq1ipfwXsp2E1QeQD6CiNAoZAl2qAlvBxBkbIt6UZBfVe
        Vo0LzbXCDzzLOvCXSwxxiLf2ktwFsY9XBak9jYJcDs8nEwwhwDNnfhn8tJ8ZnNjv
        mmSfjykdqbCSiUUImY4xjBSsgIZbuPeBLnjGQTcxhZiZ58ASw+799FScBwcjVxjZ
        Ae6iZlbQmycElpun0DEjEDsdcqJgfypq0XgDaUsOnBjF+axgTN5AlTouvQ==

        Key fingerprint:
        23:08:e1:2a:ad:fe:37:3e:8f:a0:67:ed:00:f2:3c:24
SSH Config$
```

The *host-key* follows the OpenSSL PEM format (like the DSA key). Consequently, it meets the same guidelines with a change in the delimiters (header and footer):

-----BEGIN RSA PRIVATE KEY-----

and

-----END RSA PRIVATE KEY-----

The *host-key* is encrypted to increase confidentiality. This also forces the first lines to display:

Proc-Type: 4,ENCRYPTED

DEK-Info: DES-EDE3-CBC, ...

This must be followed by a blank line and the context of the key itself, which is encoded in Base 64 as well as being encrypted (in this example) with the DES-EDE3-CBC algorithm.

If you need to copy the *host-key* to another configuration, make sure you copy the whole thing including the delimiters (header and footer), and paste it in the destination configuration using the **HOST-KEY RSA INSERT** command. The presence of the blank spaces at the beginning of the line does not pose a problem, but you must be careful not to include tabulations.

2.2.3.2.4 LIST HOST-KEY RSA1

Describes the RSA1 key, where applicable. This example shows the command executed in the static configuration. In the dynamic configuration, this also lists the public key and the *fingerprint*.

Syntax:

```
SSH config>list host-key rsa1
```

Example:

```
SSH Config>list host-key rsa1
    RSA1 (only for SSHv1)
        Host key:
        U1NIIFBSSVZBVEUgS0VZIEZJTEUgRk9STUFUIDEuMQoAAwAAAAAAAAQABADkrAMA
        VHLs6btQpuo/JU3e0t4ZGqaxdeUmNk23hKFzfXGYVOMeA1qvSBvHlnvTHKg7oIah
        Of6zkTofLnUW3i+okfnAqK2XRJ98woh/enW8Fj6DKk9Pd4Ek1KnuKn8BWPEOU+dJ
        7caGmgbfQj1t2crgcqiAYdjGN6LsToT9ytnsSwAGIwAAAA11bmtub3duX3VzZXJA
        h18uq51TTPQG+P9u7O/fU69F2EGdS5k28PW5wPExDPyiHWGz+D1Z/uHV51XBT+3K
        CMjnrWPRaiXvnDwAJ1B+isEtUuke1kykQZ6pSTppivQj6oynNPxGSQPcCJqaxLy4
        0/9g+tRZW0Kfz+/ELXtuzXl7cc1VZi9uVgFdjCqi+qRbUxmcS/QpgpbJTph895MZ
        zxVNGSOoR6iA32tyNcJlanpRauN7xAipbrNEQflVZu+H0k67EUb3T1JQ+jJmrZC+
        nsB0iBOgXo6s2KM+TdYxU+NQwa7DH0Ag6UJczAvbj+KhofTGIOskztC8Z+k4aVuz
        8YvMprTjkoJ+dcjAatCttyGZDMeM9xdYbNQyHAsyXLCvRojB7laS1yG6lZKyUgC1
        E3rm52XicGkN5+Yl4pWUZcKdeEUO7GOVn/erkXAcnNaEYUGyKkxLc2Su8Hdhn7tT

SSH Config>
```

This *host-key* does not have the OpenSSL PEM format, it follows the format used for RSA1 in many implementations: OpenSSH, ssh.com, Putty, etc. Strictly speaking, the *host-key*, which includes the private and public parts, is saved in binary code, but in order to simplify management via console, it is presented in Base 64, without headers or other fields.

The content encoded in Base 64 firstly includes the text:

“SSH PRIVATE KEY FILE FORMAT 1.1\n”

Following that there is encryption information, public key fields and finally the parameters for the private key itself, encrypted with the algorithm indicated at the beginning.

If you need to copy the *host-key* in another configuration, you need to copy all the text in Base 64 and paste it in the destination configuration using the **HOST-KEY RSA1 INSERT** command. The presence of the blank spaces at the beginning of the line does not cause any problems but you must be careful not to include tabulations. The length of each inserted line must not surpass 190 characters (if necessary, divide the lines into shorter lengths). The key being introduced is considered complete when a blank line is inserted.

2.2.3.3 LIST SERVER

This command, followed by its options, lists the values for the various parameters available in the SSH server configuration. The functionality of this command is duplicated in the **LIST** command, found in the server configuration menu (*SSHS>*). The only difference is that, depending on the main menu you are in, the term "**SERVER**" must be included.

Syntax:

```
SSH config>list server <options>
```

The use and options for this command are found in the SSH server configuration section, on entering the **LIST** command.

Command history:

Release	**Modification**
11.01.09	The *key-exchange* option was introduced as of version 11.01.09.
11.01.05.40.07	The *key-exchange* option was introduced as of version 11.01.05.40.07.
11.01.05.70.07	The *key-exchange* option was introduced as of version 11.01.05.70.07.
11.01.06.53.01	The *key-exchange* option was introduced as of version 11.01.06.53.01.

2.2.4 NO

The **NO** static command deletes the configuration for a parameter or resets its default values. The available subcommand is:

Command	Function
HOST-KEY	Deletes one of the *host-keys* present in the device.

2.2.4.1 NO HOST-KEY

Deletes the selected *host-key* from the configuration. If it does not exist, no action is taken. The possible options to define the full command are:

Command	Function
DSA	Deletes the DSA *Host-key* (for version 2).
RSA	Deletes the RSA *Host-key* (for version 2).
RSA1	Deletes the RSA1 *Host-key* (for version 1).

Before using this command, you can see that the *host-key* has been completely deleted from the configuration. This change is permanent if you use the **SAVE** command in the device's main configuration menu. If you subsequently wish to use the same key again and not a new one, then you can only do so if the user has saved it in a previous configuration file or copied the text through the **LIST HOST-KEY [TIPO]** command.

Warning

In cases where you do not deliberately generate a new key and if the SSH server is enabled, the next time the device starts up, the server searches the *host-keys* in the configuration. If it doesn't find one it needs because it's been deleted or because the compatibility has increased, it automatically generates one. This means server start up takes longer. In addition, the device has to identify with the clients using the new key, which, if not saved in the dynamic configuration, is lost on next startup.

2.2.4.1.1 NO HOST-KEY DSA

Eliminates the DSA *host-key*, used for SSHv2.

Syntax:

```
SSH config>no host-key dsa
```

Example:

```
SSH Config>no host-key dsa
SSH Config>
```

2.2.4.1.2 NO HOST-KEY RSA

Eliminates the RSA *host-key*, used for SSHv2.

Syntax:

```
SSH config>no host-key rsa
```

Example:

```
SSH Config>no host-key rsa
SSH Config>
```

2.2.4.1.3 NO HOST-KEY RSA1

Eliminates the RSA1 *host-key*, used for SSHv1.

Syntax:

```
SSH config>no host-key rsa1
```

Example:

```
SSH Config>no host-key rsa1
SSH Config>
```

2.2.5 SERVER

By means of this command, you can enter a new configuration menu: the SSH server. This is described in a different section (SSH Server Menu).

Syntax:

```
SSH config>server
```

Example:

```
SSH Config>server
-- SSH Server --
SSHS>?
  accounting         Set AAA accounting options
  auth-time          Maximum interval to complete authentication
  authentication     Available client authentication methods
  authorization      Set AAA authorization options
  ciphers            Allowed ciphers (v2)
  client-alive       Client-alive messages (v2)
  compression        Packet payload compression
  enable             Enable SSH server
  ephemeral-key      Ephemeral server key (v1)
  keep-alive         Send TCP Keep-alive messages
  key-exchange       Allowed key exchange algorithms (v2)
  list               Server configuration
  login              Set AAA login options
  macs               Allowed Message Authentication Codes (v2)
  max-auth-tries     Maximum number of authentication attemps
  max-connections    Maximum number of SSH connections
  no                 Negate a command or set its defaults
  port               Listening port
  version            Version compatibility
  exit
SSHS>
```

The text that appears at the prompt is now “SSHS”, an abbreviation for “ *Secure Shell Server*”.

Command history:

Release	Modification
11.01.09	The *key-exchange* option was introduced as of version 11.01.09.
11.01.05.40.07	The *key-exchange* option was introduced as of version 11.01.05.40.07.
11.01.05.70.07	The *key-exchange* option was introduced as of version 11.01.05.70.07.
11.01.06.53.01	The *key-exchange* option was introduced as of version 11.01.06.53.01.

2.2.6 EXIT

Returns to the configuration prompt.

Syntax:

```
SSH Config>exit
```

Example:

```
SSH Config>exit
Config>
```

2.3 SSH Server Menu

The commands to configure the SSH server are as follows:

Command	Function
ACCOUNTING	Configures the AAA accounting options.
AUTH-TIME	Maximum time permitted to complete client authentication.
AUTHENTICATION	Possible methods for client authentication.
AUTHORIZATION	Configures the AAA authorization options.
CIPHERS	Permitted encryption algorithms (SSHv2).
CLIENT-ALIVE	Messages to check that the connection is still operative (SSHv2).
COMPRESSION	Data compression in SSH packets.
ENABLE	Actives the SSH server.
EPHEMERAL-KEY	Server ephemeral key (SSHv1).
KEEP-ALIVE	Sends TCP Keep-alive packets.
KEY-EXCHANGE	Permitted key exchange algorithms (SSHv2).
LIST	Lists the server configuration.
LOGIN	Configures the AAA authentication options.
MACS	Authentication algorithms for the message (SSHv2).
MAX-AUTH-TRIES	Maximum number of client authentication attempts permitted.
MAX-CONNECTIONS	Maximum number of simultaneous SSH connections permitted.
NO	Restores the default value for a parameter.
PORT	Number of the TCP port where the SSH server listens.
VERSION	Compatibility with versions SSHv1 and SSHv2.
EXIT	Returns to the *SSH Config>* prompt.

Command history:

Release	Modification
11.01.09	The *key-exchange* option was introduced as of version 11.01.09.
11.01.05.40.07	The *key-exchange* option was introduced as of version 11.01.05.40.07.
11.01.05.70.07	The *key-exchange* option was introduced as of version 11.01.05.70.07.
11.01.06.53.01	The *key-exchange* option was introduced as of version 11.01.06.53.01.

2.3.1 ACCOUNTING

Associates an *accounting exec* or *commands* method list that has been defined using the AAA feature. This way, the SSH service applies the *accounting exec* list methods when it registers an access to the Shell and the *accounting commands* methods when an executed command is registered. The following options are available:

Command	Function
COMMANDS	Associates an *accounting commands* method list.
EXEC	Associates an *accounting exec* method list.

Method lists can only be applied if the AAA feature is enabled. To do this, once you have finished configuring the AAA, you must enable it in order to apply the lists to the distinct services. For further information on how to configure the AAA feature, please see manual *Teldat-Dm800-I AAA Feature* .

2.3.1.1 ACCOUNTING COMMANDS

Associates an *accounting commands* method list.

Syntax:

```
SSHS config>accounting commands <level> <listname>
```

<level>: Access level for the commands you want accounted.

<listname>: Identifier for the accounting method list.

Example:

```
SSHS config>accounting commands 5 AccCmds
SSHS config>
```

In the example, the *AccCmds* method list has been configured so that it can be used when accounting is executed for a level 5 command executed from SSH.

2.3.1.2 ACCOUNTING EXEC

Syntax:

```
SSHS config>accounting exec <listname>
```

<listname>: Identifier for the accounting method list.

Example:

```
SSHS config>accounting exec AccExec
SSHS config>
```

In the example, the *AccExec* method list has been configured so that it can be used when accounting is executed when the SSH Shell is accessed.

2.3.2 AUTH-TIME

Sets the maximum time SSH clients have to correctly authenticate with the server. This period begins when the process starts, once the TCP/IP connection has been established. If the maximum time period times out without the user being able to authenticate, the server starts the disconnection process. By default, this is 2 minutes.

Syntax:

```
SSHS>auth-time <interval>
```

Example:

```
SSHS>auth-time 8m30s
SSHS>
```

2.3.3 AUTHENTICATION

This command, which must be followed by an option in order to execute it, configures the authentication methods available for the client. The implemented methods are: *password*, RSA (public key for SSHv1) and Public-Key (for SSHv2). The options for this command are as follows:

Command	Function
PASSWORD	Authentication through password (SSHv1 and SSHv2).
PUBLIC-KEY	Authentication through RSA or DSA public key (SSHv2).
RSA	Authentication through RSA1 public key (SSHv1).

Users must be configured in the device so that authentication can be carried out by the client. The device must know the user that wants to connect, assigning him a certain privilege level. In cases where there aren't any users configured in the device, you need to add them using the **USER** command found in the device's root configuration menu.

The following are the only two exceptions to the above:

- Authentication is executed through a password and allows the AAA feature to manage it.
- Authentication is executed through a password and a RADIUS server is used, external to the AAA feature, which has the users configured.

In these particular cases, you do not have to have a user in the device. Having said this, if in the second case you lose the connection with the RADIUS server, the device might be left inaccessible through SSH.

2.3.3.1 AUTHENTICATION PASSWORD

Enables client authentication through a password. This is activated by default.

Syntax:

```
SSHS>authentication password
```

Example:

```
SSHS>authentication password
SSHS>
```

To deactivate this, enter the **NO AUTHENTICATION PASSWORD** command.

2.3.3.2 AUTHENTICATION PUBLIC-KEY

Accesses a submenu for the client authentication configuration through a public key for SSHv2. The reason for creating a new submenu, like **AUTHENTICATION RSA** (SSHv1), is the need to introduce the client public keys in the configuration.

Syntax:

```
SSHS>authentication public-key
```

Example:

```
SSHS>authentication public-key
-- Public-Key configuration in SSH server --
SSHS PK>?
  enable    Enable this authentication method
  key       New client key
  no        Negate a command or set its defaults
  exit
SSHS PK>
```

As you can see, the prompt has changed to “SSHS PK>”. The following commands are found in this submenu.

Command	Function
ENABLE	Enables the authentication mechanism through public key (SSHv2).
KEY	Allows you to configure the public key for a client.
NO	Eliminates a parameter or resets its default value.
EXIT	Returns to the *SSHS>* prompt.

2.3.3.2.1 ENABLE

Enables authentication through a public key. It is enabled by default.

Syntax:

```
SSHS PK>enable
```

Example:

```
SSHS PK>enable
SSHS PK>
```

To disable this method, use the **NO ENABLE** command in this submenu.

2.3.3.2.2 KEY [NAME]

Accesses a new submenu to create a public key with the name passed as a parameter. In cases where this already exists, it can be modified in said submenu. The maximum length for this is 10 characters.

Syntax:

```
SSHS PK>key <name>
```

Example:

```
SSHS PK>key pubkey1
SSHS KEY pubkey1>?
  add       Add line of data of the key
  end       Last line of data of the key
  insert    Paste client public key
  no        Negate a command or set its defaults
  user      Associate user with this key
  exit
SSHS KEY pubkey1>
```

The key name is indicated in the prompt itself. The commands available in the key submenu are as follows:

Command	Function
ADD	Adds a line to the public key that is not the last one.
END	Adds the last line to the public key.
INSERT	Allows you to enter a public key by pasting it.
NO	Deletes a parameter or restores its default value.
USER	Allows a user to use this public key.
EXIT	Returns to the *SSHS PK>* prompt.

The aim of these commands is to add a public key that is saved with the submenu name. To eliminate a key, use the **NO KEY [NAME]** command found in the *SSHS PK>* menu.

ADD [LINE]

Adds a line (the first or a middle one) to a public key. We recommend users run the **INSERT** command, which carries out checks on the public key, and then internally use the **ADD** command to store the key being configured.

END [LINE]

Adds the last line to a public key. We recommend that users run the **INSERT** command, which carries out checks on the public key, and then internally use the **END** command to store the key being configured.

INSERT

Allows you to paste a client's public key. This, which has generated the RSA or DSA key you wish to use in authentication, must configure the server so that the public part of the generated key is considered valid. You do not have to indicate the type of key, since this information is implied. This command allows you to insert the key divided into lines of at least 190 characters (when in OpenSSH format) or directly paste the file content with the public key (if the format is the one described in RFC4716).

Syntax:

```
SSHS KEY <name>>insert
```

Example 1:

```
SSHS KEY pubkey1>insert
Enter the public key (type + base64)
<cr> to escape
ssh-dss AAAAB3NzaC1kc3MAAACAdQpx45QBksY+YdceMCv1a7OYFT/nKkFghAcEE3fGbz4vPJjXpSOI
kgTARylPx+uEkmokN9nvRY0CT53r/+QlfxIBW1Z8Nu1TvI8qm0Ea0OYyDI6oEhMZhWFWKHHQmUQy1oCl
3ndqgMT4rr3fEl2hbZeujJzrNVY4EQsfSF9KhV8AAAAVAI3hGhP6mxl/FEE7Xva+JfraCwHPAAAAgGMN
blPcZeM5Dgbj1Vj/VEhpvAyvCW5E30X8jMl8YvSr7w/qaJoGAIEkgHb8efKTuUBt9nzot+QhLAiTwEe3
```

```
Nf4GxeH9ifHLRrYTh/jPKTpYucK66OcU4X/9JJzcyUl+eqQDgDWhEPHviUb3EPQTuP19nOA65TBFj3ZB
xv+tsftTAAAAgGtEZEcLQmDxsnM5pjelVtdnu7N/MHBTQTw8I+Pm+BKEOjCiBMFKB/4l5+TL1T1ocP0Q
CIttx15A9QRCkab4VEJ8pNIPnrkkNvyuk2BLgnNijwBpdVNfWEi7JSZrzadIJjlXGOcueztvggqF9CR0
f9WveLE2VMiLq7Jf2cp79yt9 dsa-key-20080128
SSHS KEY pubkey1>
```

Below is a brief explanation on the two possible formats.

The first is where the user is asked to

Enter the public key (type + base64)

This is simple and familiar to any OpenSSH user. It consists of a single line that begins with the “ssh-dss” string if the key is DSA, or with “ssh-rsa” if it’s RSA. Subsequently, after a blank space, you can see the whole public key encoded in Base 64. Finally (and optionally), there is a comment that is only used to let the user get the public key identification. Internally, this is not used and is omitted unless its specifically requested. Normally, the total line length exceeds the maximum 190 characters stipulated. Consequently, it must be inserted in shorter lines (as seen in Example 1).

Example 2:

```
SSHS KEY pubkey2>insert
Enter the public key (type + base64)
<cr> to escape
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "rsa-key-20080128"
AAAAB3NzaC1yc2EAAAABJQAAAQEAn+U+8tigcZ7FkWlRjEjPW5nMqlH2LMA9SJqJ
xE1+UVzCrb0VGYmCQkn5fC6ZoRldB2kIPUNjNxss3KU2rRZLPx+k9Jlf9lyz2+LT
8oMVKHzN0G7Nh+ZKakJu8HFweb83VnngW08gf27hy5Cn/lO1dy9t9Ib4dAlWhvf2
n0ozMCach5xRkaDrq8mj1tuGE+OOGEVu8SZIY4NXUw0buJ3BfiMPMaWnCH7Wbhuw
tfyqYvY7X9yq5grOf9qTfKXjQ8iZJPdXSNw1e9FdD1ueLjLZx72a5Pz7UcHJMUwy
DmAxGa6jZJYNtiAJobnIV3HLo+T2r3V5sR5L9qMTR3VN4RNSnw==
---- END SSH2 PUBLIC KEY ----
SSHS KEY pubkey2>
```

The second format is the one that appears in RFC4716. Its content is delimited by the lines:

---- BEGIN SSH2 PUBLIC KEY ----

and

---- END SSH2 PUBLIC KEY ----

There is no difference in the delimiters for RSA / DSA. The code is located between both lines codified in Base 64. A comment (‘Comment’) can precede this, but it is ignored. Although the public key is entered with this format, it’s translated to the former so that it can be saved in the configuration. In this case, the source file already has the length delimiter so fragmentation is not required.

NO USER [USER]

Allows you to eliminate users from among those already associated to the current key.

Syntax:

```
SSHS KEY <name>>no user <user>
```

USER [USER]

Associates a user to the current public key. If there is no associated user, the client with the corresponding private key can authenticate like any valid user in the device. However, thanks to this command, the use of this key to authenticate can be restricted to the users you select. Normally, only one user is associated to a key.

Syntax:

```
SSHS KEY <name>>user <user>
```

Example 1:

```
SSHS KEY pubkey1>user admin
SSHS KEY pubkey1>
```

EXIT

Returns to the public key configuration menu for SSHv2.

2.3.3.2.3 NO

Resets the default value for a parameter or eliminates a configuration element. There are two possible options in this submenu:

Command	Function
ENABLE	Does not allow authentication through public key (SSHv2).
KEY	Eliminates the public key identified through its name.

NO ENABLE

Disables the public key SSHv2 authentication method. By default, this is enabled.

Syntax:

```
SSHS PK>no enable
```

Example:

```
SSHS PK>no enable
SSHS PK>
```

If you want to enable this mechanism again, use the **ENABLE** command found in this submenu.

NO KEY [NAME]

Deletes the public key identified through the indicated name from the configuration. If no key with this name is found, no action is taken.

Syntax:

```
SSHS PK>no key <name>
```

Example:

```
SSHS PK>no key pubkey1
SSHS PK>
```

2.3.3.2.4 EXIT

Returns to the SSH server configuration menu.

Syntax:

```
SSHS PK>exit
```

Example:

```
SSHS PK>exit
SSHS>
```

2.3.3.3 AUTHENTICATION RSA

Accesses a submenu for client authentication through public key configuration (known as the RSA method in the SSHv1 protocol version). This is configured in a new submenu that can be accessed from the client's RSA1 public key configuration. This command is dynamic (i.e. this characteristic can be configured while the server is executing).

Syntax:

```
SSHS>authentication rsa
```

Example:

```
SSHS>authentication rsa
-- RSA configuration in SSH server --
SSHS RSA>?
  enable    Enable this authentication method
  key       New client key
  no        Negate a command or set its defaults
  exit
SSHS RSA>
```

As you can see, the prompt has changed to “SSHS RSA>”. The following commands are found in this submenu.

Command	Function
ENABLE	Enables the authentication mechanism for RSA (SSHv1).
KEY	Allows you to configure the RSA1 client public key.
NO	Eliminates a parameter or resets it to its default value.
EXIT	Returns to the *SSHS>* prompt.

2.3.3.3.1 ENABLE

Enables authentication for RSA. It is enabled by default.

Syntax:

```
SSHS RSA>enable
```

Example:

```
SSHS RSA>enable
SSHS RSA>
```

To disable this method, use the **NO ENABLE** command found in this submenu.

2.3.3.3.2 KEY [NAME]

Accesses a new submenu to create a RSA1 public key with the name parsed as a parameter. Whenever this already exists, it can be modified in this submenu. The name can have a maximum length of 10 characters.

Syntax:

```
SSHS RSA>key <name>
```

Example:

```
SSHS RSA>key rsa_k
SSHS KEY rsa_k>?
  add       Add line of data of the key
  end       Last line of data of the key
  insert    Paste client RSA public key
  no        Negate a command or set its defaults
  user      Associate user with this key
  exit
SSHS KEY rsa_k>
```

The key name is given at the prompt. The available commands in the key submenu are:

Command	Function
ADD	Adds a line to the RSA1 public key that is not the last one.
END	Adds the last line to the RSA1 public key.
INSERT	Allows you to enter an RSA1 public key by inserting it.
NO	Deletes a parameter or restores its default value.
USER	Allows a user to use this RSA1 public key.
EXIT	Returns to the *SSHS RSA>* prompt.

The aim of these commands is to add an RSA1 public key, which is saved with the submenu name. To delete a key, use the **NO KEY [NAME]** command found in the *SSHS RSA>* menu.

ADD [LINE]

Adds a line (the first or a middle one) to an RSA1 public key. We recommend that users run the **INSERT** command, which carries out checks on the RSA1 public key and then internally uses the **ADD** command to store the key being configured.

END [LINE]

Adds the last line to a public key. We recommend that users run the **INSERT** command, which carries out checks on the public key and then internally uses the **END** command to store the key being configured.

INSERT

Allows you to paste a client's RSA1 public key. The client that has generated the key you wish to use in authentication must configure the server so that the public part of the generated key is considered valid. The difference between this and SSHv2 public key configuration (**AUTHENTICATION PUBLIC-KEY**) is that, while there are two possible formats for the SSHv2 public key, the RSA1 only has one.

Syntax:

```
SSHS KEY <name>>insert
```

Example:

```
SSHS KEY rsa_k>insert
Enter the RSA public key (3 decimal numbers)
<cr> to escape
1023 37 798458743537890626287808985236489077472652668128634929439006436521582579
44058563390058992294511737156396078462633383104234138587153259159643740138784585
04153068211033284042621548285023396301262842759307031340330112910180566122424515
9936059098978755493043272344124688747878076757541173783689165017928211763217
" rsa-key-20080128"
SSHS KEY rsa_k>
```

The format for an RSA1 public key basically consists of three decimal numbers separated by blank spaces. These correspond to the module length. Like in SSHv2 public keys, a comment can be added at the end for the user’s benefit. In cases where a new line is entered, this must be in inverted commas and with a blank space at the beginning (thus indicating this is not part of the key).

NO USER [USER]

Only allows you to eliminate users from among those already associated to the current key.

Syntax:

```
SSHS KEY <name>>no user <user>
```

USER [USER]

Associates a user to the current RSA1 public key. If there is no associated user, the client that has the corresponding private key can authenticate like any valid user in the device. However, thanks to this command, the use of this key to authenticate can be restricted to the users selected. Normally, only one user is associated to a key.

Syntax:

```
SSHS KEY <name>>user <user>
```

Example 1:

```
SSHS KEY rsa_k>user admin
SSHS KEY rsa_k>
```

EXIT

Returns to the public key configuration menu for SSHv1.

2.3.3.3.3 NO

Resets the default value for a parameter or deletes a configuration element. There are two possible options in this submenu:

Command	Function
ENABLE	Does not permit authentication through public key.
KEY	Eliminates the public key identified through its name.

NO ENABLE

Disables the SSHv1 RSA authentication through the public key. This is enabled by default.

Syntax:

```
SSHS RSA>no enable
```

Example:

```
SSHS RSA> no enable
SSHS RSA>
```

If you want to enable this mechanism again, use the **ENABLE** command found in this submenu.

NO KEY [NAME]

Deletes the RSA1 public key identified through the indicated name from the configuration. If no key with this name is found, no action is taken.

Syntax:

```
SSHS RSA>no key <name>
```

Example:

```
SSHS RSA>no key rsa_k
SSHS RSA>
```

2.3.3.3.4 EXIT

Returns to the SSH server configuration menu.

Syntax:

```
SSHS RSA>exit
```

Example:

```
SSHS RSA>exit
SSHS>
```

2.3.4 AUTHORIZATION

Associates an *authorization exec* or *commands* method list defined through the AAA feature. This way, the SSH service applies the *authorization exec* method list when it requires authorization from Shell and the *authorization commands* method list when it requires authorization from a command. The following options are available:

Command	Function
COMMANDS	Associates an *authorization commands* method list.
EXEC	Associates an *authorization exec* method list.

Method lists can only be applied if the AAA feature is enabled. To do this, once you have finished configuring the AAA, you must enable the feature to apply the lists to the services selected. For further information on how to configure the AAA feature, please see manual *Teldat-Dm800-I AAA Feature* .

2.3.4.1 AUTHORIZATION EXEC

Associates an *authorization exec* method list.

Syntax:

```
SSHS config>authorization exec <listname>
```

<listname>: Identifier for the authorization method list.

Example:

```
SSHS config>authorization exec AuthorExec
SSHS config>
```

In the example, the *AuthorExec* method list has been configured so that it can be used when authorization is required from the SSH Shell.

2.3.4.2 AUTHORIZATION COMMANDS

Associates an *authorization commands* method list.

Syntax:

```
SSHS config>authorization commands <level> <listname>
```

<level>: Access level for the commands that require authorization.

<listname>: Identifier for the authorization method list.

Example:

```
SSHS config>authorization commands 10 AuthorCmds
SSHS config>
```

2.3.5 CIPHERS

Through this command, you can select the cipher algorithms allowed to encrypt the SSH connection for SSHv2. Since they are all enabled by default, the user can decide which one to use. The algorithms implemented are as follows:

Command	Function
3DES-CBC	Triple DES with *cipher-block chaining*.
AES128-CBC	AES with *cipher-block chaining* and a 128 bits key.
AES128-CTR	AES with cipher counter and a 128 bits key.
AES192-CBC	AES with *cipher-block chaining* and a 192 bits key.
AES192-CTR	AES with cipher counter and a 192 bits key.
AES256-CBC	AES with *cipher-block chaining* and a 256 bits key.
AES256-CTR	AES with cipher counter and a 256 bits key.
ARCFOUR	Alleged-RC4.
ARCFOUR128	Alleged-RC4 with a 128 bits key.
ARCFOUR256	Alleged-RC4 with a 256 bits key.
BLOWFISH-CBC	Blowfish with *cipher-block chaining*.
CAST128-CBC	CAST with *cipher-block chaining* and a 128 bits key.

To disable a cipher algorithm, execute the **NO CIPHERS [NAME]** command.

2.3.5.1 CIPHERS [NAME]

Enables the cipher algorithm with the indicated name.

Syntax:

```
SSHS>ciphers <name>
```

Example:

```
SSHS>ciphers aes256-cbc
SSHS>
```

2.3.6 CLIENT-ALIVE

The SSHv2 *Client-Alive* feature executes a periodic check on the SSH client to make sure it's still active. To do this, the server sends *Client-Alive* messages according to the intervals set during configuration. When the configured number of messages has been sent without any response from the client, the connection is considered down and the server closes it. The subcommands are as follows:

Command	Function
INTERVAL	Time period between two consecutive *Client-Alive* messages.
MAX-COUNT	Maximum number of messages sent without a response.

2.3.6.1 CLIENT-ALIVE INTERVAL [TIME]

Establishes the time interval that must be observed before sending a *Client-Alive* message when there is no traffic in the connection. A 0 period means this feature is disabled. Default is 15 seconds.

Syntax:

```
SSHS>client-alive interval <time>
```

Example:

```
SSHS>client-alive interval 1m30s
SSHS>
```

2.3.6.2 CLIENT-ALIVE MAX-COUNT [COUNTER]

If, after a given number of consecutive *Client-Alive* messages, there is still no response, the server closes the connection and considers the client to be down. Default is 3 messages.

Syntax:

```
SSHS>client-alive max-count <counter>
```

Example:

```
SSHS>client-alive max-count 5
SSHS>
```

2.3.7 COMPRESSION

Configures the server-related permissions for negotiation and the subsequent use of compression in the SSH connection. There are three options for this parameter: delayed compression, compression together with delayed compression (**YES**) and no compression (**NO**). Default is **YES**.

Command	Function
YES	Allows compression and delayed compression.
NO	Compression not permitted.
DELAYED	Allows compression only after client authentication (used in OpenSSH as zlib@openssh.com).

SSH clients often don't use compression by default, although this is enabled in the server.

2.3.7.1 COMPRESSION YES

Enables normal and delayed compression methods. This is the default configuration.

Syntax:

```
SSHS>compression yes
```

Example:

```
SSHS>compression yes
SSHS>
```

2.3.7.2 COMPRESSION NO

Disables all SSH packet compression methods. Even if the client wishes to establish a connection with compression, the server doesn't allow it.

Syntax:

```
SSHS>compression no
```

Example:

```
SSHS>compression no
SSHS>
```

2.3.7.3 COMPRESSION DELAYED

Only allows delayed compression, known as *zlib@openssh.com*. This happens when the content of SSH packets is encrypted after the client correctly authenticates with the server. Only clients that support this method can use it. Otherwise, compression is not possible.

Syntax:

```
SSHS>compression delayed
```

Example:

```
SSHS>compression delayed
SSHS>
```

2.3.8 ENABLE

This parameter is particularly important as it globally enables the SSH server. By default, the server is disabled. Therefore, you need to execute this command in order to use the protocol.

Syntax:

```
SSHS>enable
```

Example:

```
SSHS>enable

Hostkeys for SSHv1 and SSHv2 not found in config
Generating public/private rsa1 key pair...
Please wait for a few seconds.
Key generation done.

        Public key:
        1024 35 1568556586875626099971500785026211653036874306758671168З
        22184788826833043935933543920769056839765015219442542539349020 54
        671967161708139115215784939202078040765890242686353268235077335 4
        552725435518360387020292577714980517382428073487968168260250356 9
        1993500883799434553852193298247418895281943943053504296554551

        The key fingerprint is:
        c2:c4:54:e2:2e:c6:35:be:b0:be:d4:16:83:4f:3b:9d

Generating public/private rsa key pair...
Please wait for a few seconds.
Key generation done.

        Public key:
        ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAyMEOF3crF4aP9HqXXCAljVYqMGLk
        2KicRzqxvy+d+CyKo0bwb5T8Wg11ksPntnGku5s0284Ou1uzTyxZMSAqG8fdSgzf
        UL8Ow8FfH3GjyrSBoS55gkyKm0jQXJQAQDruEkVP/Bug2L+QynuAELo+B4hYi7Gg
        IO5NttdTL6Uck7k=

        The key fingerprint is:
        49:41:e6:c5:9c:60:73:b2:4c:60:eb:01:29:a4:fc:21

Hostkeys have been generated. Remember to save config!
SSHS>
```

Here, the command has triggered the generation of *host-keys* to be used in the device. This happened because the required keys were not found in the configuration. Through the **VERSION** command, the server's compatible versions are established (SSHv1 and SSHv2 by default). On executing this command, this checks the configured version and the *host-keys* present in the device. If the required keys aren't found, then an RSA1 or RSA 1024 bits key is created (depending on the version). The example reports that keys have not been found for either SSHv1 or SSHv2 and, as a result, both have been generated.

If you want to use a *host-key* that has been previously used in other configurations, we recommend using the **HOST-KEY [TYPE] INSERT** command before executing the **ENABLE** command. Consequently, the necessary keys are located in the configuration and the current command does not trigger the key generation process.

If you want to disable the server again (default state), use the **NO ENABLE** command found in the *SSHS>* menu.

2.3.9 EPHEMERAL-KEY

Configures the server's ephemeral key, used in SSHv1 connections. A new key is created with the configured number of bits when the server initiates. This is known as ephemeral, as it is periodically generated according to the interval established by clients. If no client connects, then the server doesn't use this key and it isn't renewed. Two clients that connect to the server after a period longer than the one configured has lapsed use different ephemeral keys. The configurable parameters are as follows:

Command	Function
BITS	Number of key bits, between 512 and 2048 bits.
REGENERATION-INTERVAL	Time period to regenerate the key.

2.3.9.1 EPHEMERAL-KEY BITS [NUM_BITS]

Sets the number of bits the ephemeral key should have. Default is 768 bits.

Syntax:

```
SSHS>ephemeral-key bits <num_bits>
```

Example:

```
SSHS>ephemeral-key bits 1536
SSHS>
```

2.3.9.2 EPHEMERAL-KEY REGENERATION-INTERVAL [TIME]

Configures the time period from the moment an ephemeral key is first used until it is regenerated to make it more robust. If a second client triggers a connection before this interval expires, the same key is used. Please note that, since the key generating process can entail delays, a short interval is not recommended. This is one hour by default.

Syntax:

```
SSHS>ephemeral-key regeneration-interval <time>
```

Example:

```
SSHS>ephemeral-key regeneration-interval 30m
SSHS>
```

2.3.10 KEEP-ALIVE

Activates the sending of TCP *Keep-alive* packets. This way, a periodic check is executed to make sure the TCP connection is still established between client and server. This is activated by default.

Syntax:

```
SSHS>keep-alive
```

Example:

```
SSHS>keep-alive
SSHS>
```

To deactivate this functionality, use the **NO KEEP-ALIVE** command.

2.3.11 KEY-EXCHANGE

Allows the key exchange algorithms used to generate per-connection keys in SSHv2 to be selected. By default, they're all enabled so the client can select one. The algorithms available are:

Command	Function
diffie-hell-man-group-exchange-sha1	This set of ephemerally generated key exchange groups uses SHA-1.
diffie-hell-man-group-exchange-sha256	This set of ephemerally generated key exchange groups uses SHA2-256.
diffie-hellman-group1-sha1	This method uses Oakley Group 2 (a 1024-bit MODP group) and SHA-1.
diffie-hellman-group14-sha1	This method uses group14 (a 2048-bit MODP group) and SHA-1.

To disable an algorithm, execute **NO KEY-EXCHANGE [NAME]** .

Syntax:

```
SSHS>key-exchange <name>
```

Example:

```
SSHS>key-exchange diffie-hellman-group-exchange-sha256
SSHS>
```

Command history:

Release	Modification
11.01.09	The *key-exchange* option was introduced as of version 11.01.09.
11.01.05.40.07	The *key-exchange* option was introduced as of version 11.01.05.40.07.
11.01.05.70.07	The *key-exchange* option was introduced as of version 11.01.05.70.07.

Release	Modification
11.01.06.53.01	The *key-exchange* option was introduced as of version 11.01.06.53.01.

2.3.12 LIST

Displays the values for the configurable parameters in the SSH server on the console. Not only does it list those configured, but also the ones with a default value. This command is used in the same way as the **LIST SERVER** command, found in the main SSH configuration menu (*SSH Config*>). You can list the whole configuration (**LIST ALL**) or simply focus on specific parameters. The available options are:

Command	Function
ALL	Lists the SSH server configuration in full.
AUTH-TIME	Maximum time permitted to complete client authentication.
AUTHENTICATION	Possible methods for client authentication.
CIPHERS	Encryption algorithms allowed (SSHv2).
CLIENT-ALIVE	Messages used to check that the connection is still operating (SSHv2).
COMPRESSION	Data compression in SSH packets.
ENABLE	Activates the SSH server.
EPHEMERAL-KEY	Server ephemeral key (SSHv1).
KEEP-ALIVE	TCP *Keep-alive* packet sending.
KEY-EXCHANGE	Key exchange algorithms allowed (SSHv2).
MACS	Message authentication algorithms (SSHv2).
MAX-AUTH-TRIES	Maximum number of client authentication attempts allowed.
MAX-CONNECTIONS	Maximum number of simultaneous SSH connections permitted.
PORT	TCP port number where the SSH server listens.
VERSION	Compatibility with SSHv1 and SSHv2 versions.

Command history:

Release	Modification
11.01.09	The *key-exchange* option was introduced as of version 11.01.09.
11.01.05.40.07	The *key-exchange* option was introduced as of version 11.01.05.40.07.
11.01.05.70.07	The *key-exchange* option was introduced as of version 11.01.05.70.07.
11.01.06.53.01	The *key-exchange* option was introduced as of version 11.01.06.53.01.

2.3.12.1 LIST ALL

Displays the SSH server configuration. The output is similar to consecutively listing each of the remaining options.

Syntax:

```
SSHS>list all
```

Example:

```
SSHS>list all

SSH Server configuration:
        Server status: enabled


        Version compatibility: SSHv1 and SSHv2


        Listening port:    22


        Payload compression: enabled


        Ciphers:
                    3des-cbc : available
                  aes128-cbc : available
                  aes192-cbc : available
                  aes256-cbc : available
                  aes128-ctr : available
```

```
                aes192-ctr : available
                aes256-ctr : available
                arcfour128 : available
                arcfour256 : available
                   arcfour : available
              blowfish-cbc : available
               cast128-cbc : available

        Message Authentication Codes:
                  hmac-md5 : available
                 hmac-sha1 : available
            hmac-ripemd160 : available
              hmac-sha1-96 : available
               hmac-md5-96 : available
             hmac-sha2-256 : available
             hmac-sha2-512 : available

        KEX Algorithms:
                   diffie-hellman-group1-sha1 : available
                  diffie-hellman-group14-sha1 : available
           diffie-hellman-group-exchange-sha1 : available
         diffie-hellman-group-exchange-sha256 : available

        Authentication methods:
          password : available
          public-key : available
          rsa : available

        Maximum number of authentication attempts:  6
        Maximum time to complete authentication:   2m0s
        Maximum number of SSH connections:  4

        Keep-alive activated: yes

        Client-alive message parameters:
                Maximum number of messages sent without response:  3
                Interval between messages:    15s

        Ephemeral server key parameters:
                Number of bits:   768
                Interval to regenerate server key: 1h0m0s


SSHS>
```

When executed, a default configuration is shown where only the client public keys have been added.

Command history:

Release	Modification
11.01.09	The command output has changed as of version 11.01.09. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.05.70.06	The command output has changed as of version 11.01.05.70.06. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.05.40.07	The command output has changed as of version 11.01.05.40.07. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.06.53.01	The command output has changed as of version 11.01.06.53.01. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.09	The command output has changed as of version 11.01.09. The possibility of selecting the key exchange algorithms to generate per-connection keys in SSHv2 has been added on the SSH server and this command shows their availability.
11.01.05.40.07	The command output has changed as of version 11.01.05.40.07. The possibility of select-

Release	Modification
	ing the key exchange algorithms to generate per-connection keys in SSHv2 has been added on the SSH server and this command shows their availability.
11.01.05.70.07	The command output has changed as of version 11.01.05.70.07. The possibility of selecting the key exchange algorithms to generate per-connection keys in SSHv2 has been added on the SSH server and this command shows their availability.
11.01.06.53.01	The command output has changed as of version 11.01.06.53.01. The possibility of selecting the key exchange algorithms to generate per-connection keys in SSHv2 has been added on the SSH server and this command shows their availability.

2.3.12.2 LIST AUTH-TIME

Shows the time available to execute authentication.

Syntax:

```
SSHS>list auth-time
```

Example:

```
SSHS>list auth-time
        Maximum time to complete authentication:   2m0s
SSHS>
```

2.3.12.3 LIST AUTHENTICATION

Provides information on the available client authentication mechanisms.

Syntax:

```
SSHS>list authentication
```

Example:

```
SSHS>list authentication
        Authentication methods:
          Password : available
          Public-key : available
            Key : pubkey1
                Users :
                   admin

            ssh-dss AAAAB3NzaC1kc3MAAACAdQpx45QBksY+YdceMCv1a7
            OYFT/nKkFghAcEE3fGbz4vPJjXpSOIkgTARylPx+uEkmokN9nv
            RY0CT53r/+QlfxIBW1Z8Nu1TvI8qm0Ea0OYyDI6oEhMZhWFWKH
            HQmUQy1oCl3ndqgMT4rr3fEl2hbZeujJzrNVY4EQsfSF9KhV8A
            AAAVAI3hGhP6mxl/FEE7Xva+JfraCwHPAAAAgGMNblPcZeM5Dg
            bj1Vj/VEhpvAyvCW5E30X8jMl8YvSr7w/qaJoGAIEkgHb8efKT
            uUBt9nzot+QhLAiTwEe3Nf4GxeH9ifHLRrYTh/jPKTpYucK66O
            cU4X/9JJzcyUl+eqQDgDWhEPHviUb3EPQTuP19nOA65TBFj3ZB
            xv+tsftTAAAAgGtEZEcLQmDxsnM5pjelVtdnu7N/MHBTQTw8I+
            Pm+BKEOjCiBMFKB/4l5+TL1T1ocP0QCIttx15A9QRCkab4VEJ8
            pNIPnrkkNvyuk2BLgnNijwBpdVNfWEi7JSZrzadIJjlXGOcuez
            tvggqF9CR0f9WveLE2VMiLq7Jf2cp79yt9 dsa-key-2008012
            8
            Key : pubkey2
                Users :
                   client

            ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAn+U+8tigcZ7FkW
            lRjEjPW5nMqlH2LMA9SJqJxE1+UVzCrb0VGYmCQkn5fC6ZoRld
            B2kIPUNjNxss3KU2rRZLPx+k9Jlf9lyz2+LT8oMVKHzN0G7Nh+
            ZKakJu8HFweb83VnngW08gf27hy5Cn/lO1dy9t9Ib4dAlWhvf2
            n0ozMCach5xRkaDrq8mj1tuGE+OOGEVu8SZIY4NXUw0buJ3Bfi
            MPMaWnCH7WbhuwtfyqYvY7X9yq5grOf9qTfKXjQ8iZJPdXSNw1
            e9FdD1ueLjLZx72a5Pz7UcHJMUwyDmAxGa6jZJYNtiAJobnIV3
            HLo+T2r3V5sR5L9qMTR3VN4RNSnw==
          RSA : available
```

```
            Key : rsa_k
                Users :
                   client2

            1023 37 1023 37 798458743537890626287808985236489077472652
            66812863492943900643652158257944058563390058992294
            51173715639607846263338310423413858715325915964374
            01387845850415306821103328404262154828502339630126
            28427593070313403301129101805661224245159936059098

            97875549304327234412468874787807675754117378368916
            5017928211763217 rsa-key-20080128
SSHS>
```

Apart from the availability of each method, this command also displays each of the public keys admitted by the Public-key and RSA methods, together with the users that can use them. When it comes to user and password configuration, this is not part of the SSH protocol but is general for all devices. The relevant command is **USER**, found in the root configuration menu.

You can list each authentication method separately, adding an option:

Command	Function
PASSWORD	Password use.
PUBLIC-KEY	Use of the RSA / DSA (SSHv2) public key and admitted keys.
RSA	Use of the RSA (SSHv1) and admitted keys.

2.3.12.3.1 LIST AUTHENTICATION PASSWORD

Permission notification so that the client uses a password in authentication. Associated users are specified for each key.

Syntax:

```
SSHS>list authentication password
```

Example:

```
SSHS>list authentication password
          Password : available
SSHS>
```

2.3.12.3.2 LIST AUTHENTICATION PUBLIC-KEY

Permission notification so that the client uses a public key (SSHv2) in authentication. The command also lists those admitted.

Syntax:

```
SSHS>list authentication public-key
```

Example:

```
SSHS>list authentication public-key
          Public-key : available

            Key : pubkey1
                Users :
                   admin

            ssh-dss AAAAB3NzaC1kc3MAAACAdQpx45QBksY+YdceMCv1a7
            OYFT/nKkFghAcEE3fGbz4vPJjXpSOIkgTARylPx+uEkmokN9nv
            RY0CT53r/+QlfxIBW1Z8Nu1TvI8qm0Ea0OYyDI6oEhMZhWFWKH
            HQmUQy1oCl3ndqgMT4rr3fEl2hbZeujJzrNVY4EQsfSF9KhV8A
            AAAVAI3hGhP6mxl/FEE7Xva+JfraCwHPAAAAgGMNblPcZeM5Dg
            bj1Vj/VEhpvAyvCW5E30X8jMl8YvSr7w/qaJoGAIEkgHb8efKT
            uUBt9nzot+QhLAiTwEe3Nf4GxeH9ifHLRrYTh/jPKTpYucK66O
            cU4X/9JJzcyUl+eqQDgDWhEPHviUb3EPQTuP19nOA65TBFj3ZB
            xv+tsftTAAAAgGtEZEcLQmDxsnM5pjelVtdnu7N/MHBTQTw8I+
            Pm+BKEOjCiBMFKB/4l5+TL1T1ocP0QCIttx15A9QRCkab4VEJ8

            pNIPnrkkNvyuk2BLgnNijwBpdVNfWEi7JSZrzadIJjlXGOcuez
            tvggqF9CR0f9WveLE2VMiLq7Jf2cp79yt9 dsa-key-2008012
            8
            Key : pubkey2
                Users :
                   client

            ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAn+U+8tigcZ7FkW
            lRjEjPW5nMqlH2LMA9SJqJxE1+UVzCrb0VGYmCQkn5fC6ZoRld
            B2kIPUNjNxss3KU2rRZLPx+k9Jlf9lyz2+LT8oMVKHzN0G7Nh+
            ZKakJu8HFweb83VnngW08gf27hy5Cn/lO1dy9t9Ib4dAlWhvf2
            n0ozMCach5xRkaDrq8mj1tuGE+OOGEVu8SZIY4NXUw0buJ3Bfi
            MPMaWnCH7WbhuwtfyqYvY7X9yq5grOf9qTfKXjQ8iZJPdXSNw1
            e9FdD1ueLjLZx72a5Pz7UcHJMUwyDmAxGa6jZJYNtiAJobnIV3
            HLo+T2r3V5sR5L9qMTR3VN4RNSnw==

SSHS>
```

2.3.12.3.3 LIST AUTHENTICATION RSA

Permission notification so that the client uses the RSA public key in authentication, with RSA1 format (SSHv1). This command also lists the admitted keys. Associated users are specified for each key.

Syntax:

```
SSHS>list authentication rsa
```

Example:

```
SSHS>list authentication rsa
          RSA : available
            Key : rsa_k
                Users :
                   client2

            1023 37 798458743537890626287808985236489077472652
            668128634929439006436521582579440585633900589922 94
            511737156396078462633383104234138587153259159643 74
            013878458504153068211033284042621548285023396301 26
            284275930703134033011291018056612242451599360590 98
            978755493043272344124688747878076757541173783689 16
            5017928211763217 rsa-key-20080128

SSHS>
```

2.3.12.4 LIST CIPHERS

Lists the availability of each of the encryption algorithms implemented.

Syntax:

```
SSHS>list ciphers
```

Example:

```
SSHS>list ciphers
        Ciphers:
                  3des-cbc : available
                  aes128-cbc : available
                  aes192-cbc : available
                  aes256-cbc : available
                  aes128-ctr : available
                  aes192-ctr : available
                  aes256-ctr : available
                  arcfour128 : available
                  arcfour256 : available
                  arcfour : available
                  blowfish-cbc : available
                  cast128-cbc : available
SSHS>
```

2.3.12.5 LIST CLIENT-ALIVE

This command should be executed followed by one of the two following options:

Command	Function
INTERVAL	Time interval between two consecutive *Client-Alive* messages.
MAX-COUNT	Maximum number of messages sent without a response.

2.3.12.5.1 LIST CLIENT-ALIVE INTERVAL

Indicates how often *Client-Alive* messages are sent (without data traffic).

Syntax:

```
SSHS>list client-alive interval
```

Example:

```
SSHS>list client-alive interval
        Client-alive message parameters:
                Interval between messages:    15s
SSHS>
```

2.3.12.5.2 LIST CLIENT-ALIVE MAX-COUNT

Indicates the number of *Client-Alive* messages that can be sent without a response before the server closes the connection.

Syntax:

```
SSHS>list client-alive max-count
```

Example:

```
SSHS>list client-alive max-count
        Client-alive message parameters:
                Maximum number of messages sent without response:  3
SSHS>
```

2.3.12.6 LIST COMPRESSION

Shows if data compression in SSH packets is permitted, whether this includes standardized compression (zlib) or only delayed compression after authentication (zlib@openssh.com).

Syntax:

```
SSHS>list compression
```

Example:

```
SSHS>list compression
        Payload compression: enabled
SSHS>
```

2.3.12.7 LIST ENABLE

Use this command to discover whether the SSH server is enabled or not.

Syntax:

```
SSHS>list enable
```

Example:

```
SSHS>list enable
        Server status: enabled
SSHS>
```

2.3.12.8 LIST EPHEMERAL-KEY

This command should be executed followed by one of the two following options:

Command	Function
BITS	Ephemeral key length in bits.
REGENERATION-INTERVAL	Interval established to regenerate the ephemeral key.

2.3.12.8.1 LIST EPHEMERAL-KEY BITS

Gives the number of bits used to generate the SSHv1 ephemeral key.

Syntax:

```
SSHS>list ephemeral-key bits
```

Example:

```
SSHS>list ephemeral-key bits
        Ephemeral server key parameters:
                Number of bits:   768
SSHS>
```

2.3.12.8.2 LIST EPHEMERAL-KEY REGENERATION-INTERVAL

Indicates the time interval since an ephemeral key is used until its next generation.

Syntax:

```
SSHS>list ephemeral-key regeneration-interval
```

Example:

```
SSHS>list ephemeral-key regeneration-interval
        Ephemeral server key parameters:
                Interval to regenerate server key: 1h0m0s
SSHS>
```

2.3.12.9 LIST KEEP-ALIVE

Lists the TCP *Keep-Alive* packet sending configuration, used to ensure connection is established.

Syntax:

```
SSHS>list keep-alive
```

Example:

```
SSHS> list keep-alive
        Keep-alive activated: yes
SSHS>
```

2.3.12.10 LIST KEY-EXCHANGE

Lists the availability of each of the key exchange algorithms implemented.

Syntax:

```
SSHS>list key-exchange
```

Example:

```
SSHS> list key-exchange
        KEX Algorithms:
                     diffie-hellman-group1-sha1 : available
                    diffie-hellman-group14-sha1 : available
             diffie-hellman-group-exchange-sha1 : available
           diffie-hellman-group-exchange-sha256 : available
SSHS>
```

Command history:

Release	Modification
11.01.09	The *key-exchange* option was introduced as of version 11.01.09.
11.01.05.40.07	The *key-exchange* option was introduced as of version 11.01.05.40.07.

Release	Modification
11.01.05.70.07	The *key-exchange* option was introduced as of version 11.01.05.70.07.
11.01.06.53.01	The *key-exchange* option was introduced as of version 11.01.06.53.01.

2.3.12.11 LIST MACS

Displays the availability of each message authentication code (MAC) implemented.

Syntax:

```
SSHS>list macs
```

Example:

```
SSHS>list macs
        Message Authentication Codes:
                    hmac-md5 : available
                   hmac-sha1 : available
              hmac-ripemd160 : available
                hmac-sha1-96 : available
                 hmac-md5-96 : available
               hmac-sha2-256 : available
               hmac-sha2-512 : available
SSHS>
```

Command history:

Release	Modification
11.01.09	The command output has changed as of version 11.01.09. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.05.70.06	The command output has changed as of version 11.01.05.70.06. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.05.40.07	The command output has changed as of version 11.01.05.40.07. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.
11.01.06.53.01	The command output has changed as of version 11.01.06.53.01. The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command shows their availability.

2.3.12.12 LIST MAX-AUTH-TRIES

Lists the maximum number of client authentication attempts permitted. When this value has been reached and all attempts have failed, the server closes the connection.

Syntax:

```
SSHS>list max-auth-tries
```

Example:

```
SSHS>list max-auth-tries
        Maximum number of authentication attempts:  6
SSHS>
```

2.3.12.13 LIST MAX-CONNECTIONS

Maximum number of simultaneous SHH connections admitted by the server.

Syntax:

```
SSHS>list max-connections
```

Example:

```
SSHS>list max-connections
        Maximum number of SSH connections:  4
SSHS>
```

2.3.12.14 LIST PORT

Displays the TCP port number where the SSH server listens.

Syntax:

```
SSHS>list port
```

Example:

```
SSHS>list port
        Listening port:    22
SSHS>
```

2.3.12.15 LIST VERSION

Lists server compatibility with SSHv1 and SSHv2 versions.

Syntax:

```
SSHS>list version
```

Example:

```
SSHS>list version
        Version compatibility: SSHv1 and SSHv2
SSHS>
```

2.3.13 LOGIN

Links an *authentication login* method list that has been defined using the AAA feature. This way, the SSH service applies the methods from the associated list when it needs to execute authentication for a user.

Syntax:

```
SSHS config>login authentication <listname>
```

<listname>: Identifier for the authentication methods list.

Example:

```
SSHS config>login authentication AutheLogin
SSHS config>
```

In the example, the *AutheLogin* method list has been configured so that it can be used when a user accessing through SSH requires authentication.

Method lists can only be applied if the AAA feature is enabled. To do this, once you have finished configuring the AAA, you must enable it in order to apply the lists to the distinct services. For further information on how to configure the AAA feature, please see manual *Teldat-Dm800-I AAA Feature* .

2.3.14 MACS

Through this command, the permitted Message Authentication Codes (*MACs*) are selected. These are used to verify the packet integrity in SSHv2. By default, they're all enabled so the client can choose one. The implemented codes are:

Command	Function
HMAC-MD5	MD5 *hash* algorithm.
HMAC-MD5-96	MD5 *hash* algorithm truncated to 96 bits.
HMAC-RIPEMD160	RIPEMD *hash* algorithm with 160 bits.
HMAC-SHA1	SHA-1 *hash* algorithm.
HMAC-SHA1-96	SHA-1 *hash* algorithm truncated to 96 bits.
HMAC-SHA2-256	SHA-2 *hash* algorithm with a digest length and a key length of 256 bits.
HMAC-SHA2-512	SHA-2 *hash* algorithm with a digest length and a key length of 512 bits.

To disable an encryption code, execute **NO MACS [NAME]**.

Command history:

Release	Modification
11.01.09	The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes were introduced as of version 11.01.09.
11.01.05.70.06	The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes were introduced as of version 11.01.05.70.06.
11.01.05.40.07	The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes were introduced as of version 11.01.05.40.07.
11.01.06.53.01	The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes were introduced as of version 11.01.06.53.01.

2.3.14.1 MACS [NAME]

Enables the message authentication code through the indicated name.

Syntax:

```
SSHS>macs <name>
```

Example:

```
SSHS>macs hmac-md5
SSHS>
```

2.3.15 MAX-AUTH-TRIES

Establishes the maximum number of client authentication arrempts. Once the number of failed attempts has been reached, the server closes the connection. Default is 6.

Syntax:

```
SSHS>max-auth-tries <number>
```

Example:

```
SSHS>max-auth-tries 3
SSHS>
```

You must take into account the maximum authentication time (**AUTH-TIME**) since, if this interval is surpassed, the connection closes anyway (even when the maximum number of attempts hasn't been reached). Lastly, please note that the client cannot be authenticated with a different user name if this is the erroneous parameter, so care must be taken when introducing the user name.

2.3.16 MAX-CONNECTIONS

Sets the maximum number of simultaneous SSH connections that the server is able to maintain. Consequently, there can be as many SSH clients connected as indicated by the value in this parameter. Default is a maximum of 4 connections.

Syntax:

```
SSHS>max-connections <number>
```

Example:

```
SSHS>max-connections 10
SSHS>
```

In cases where, when dynamically configuring this parameter, the new maximum is lower than the number of working SSH connections, none of the connections are closed. However, the server does not admit new clients until the number of current connections is less than the new maximum established.

Another aspect, although one that can't be configured, is the maximum number of clients connected to the server but not authenticated. This value is 4 unauthenticated connections. Consequently, when a new client tries to connect to the server, the latter rejects the connection if the maximum has been reached. Once the unauthenticated clients satisfactorily complete the authentication processes or close their connections, the server begins to accept new connections until the maximum set through **MAX-CONNECTIONS** has been reached.

2.3.17 NO

Re-establishes the default values or disables an SSH server feature.

Command	Function
ACCOUNTING	Configures the AAA accounting options.
AUTH-TIME	Default value for the maximum time permitted for authentication.
AUTHENTICATION	Disables a client authentication method.
AUTHORIZATION	Configures the AAA authorization options.
CIPHERS	Prevents the use of an encryption algorithm.
CLIENT-ALIVE	Default value for a *Client-Alive* parameter.
COMPRESSION	Compression default value (compression permitted).
ENABLE	Disables the SSH server.
EPHEMERAL-KEY	Default value for an ephemeral key parameter.
KEEP-ALIVE	Deactivates *Keep-Alive* TCP packet sending.
KEY-EXCHANGE	Prevents the use of a key exchange algorithm.
LOGIN	Configures the AAA authentication options.
MACS	Prevents the use of a message authentication method.
MAX-AUTH-TRIES	Default value for maximum authentication tries.
MAX-CONNECTIONS	Default value for the maximum simultaneous SSH connections.
PORT	Default value for the TCP listening port for the server.
VERSION	Default value for compatibility (both versions enabled).

Command history:

Release	Modification
11.01.09	The *key-exchange* option was introduced as of version 11.01.09.
11.01.05.40.07	The *key-exchange* option was introduced as of version 11.01.05.40.07.
11.01.05.70.07	The *key-exchange* option was introduced as of version 11.01.05.70.07.
11.01.06.53.01	The *key-exchange* option was introduced as of version 11.01.06.53.01.

2.3.17.1 NO ACCOUNTING

Deletes a method list defined using AAA for SSH accounting.

Syntax:

```
SSHS>no accounting {commands <privilege-level> | exec}
```

Example:

```
SSHS>no accounting commands 10
SSHS>
```

2.3.17.2 NO AUTH-TIME

Re-establishes the default value for the maximum time permitted for authentication. This is 2 minutes.

Syntax:

```
SSHS>no auth-time
```

Example:

```
SSHS>no auth-time
SSHS>
```

2.3.17.3 NO AUTHENTICATION PASSWORD

Disables client authentication through user and password. Default is enabled.

Syntax:

```
SSHS>no authentication password
```

Example:

```
SSHS>no authentication password
SSHS>
```

As for the remaining authentication methods (*Public-Key* and *RSA*), the corresponding **NO** commands can be found in the *SSHS PK>* and *SSHS RSA>* menus. For further information, please see the **AUTHENTICATION PUBLIC-KEY** and **AUTHENTICATION RSA** commands found in the *SSHS>* server configuration menu.

2.3.17.4 NO AUTHORIZATION

Deletes a methods list defined using AAA for SSH authorization.

Syntax:

```
SSHS>no authorization {commands <privilege-level> | exec}
```

Example:

```
SSHS>no authorization exec
SSHS>
```

2.3.17.5 NO CIPHERS

Disables the encryption method indicated through its name, changing its status to not permitted. This is only relevant for SSHv2. By default, all implemented encryption algorithms are enabled:

Command	Function
3DES-CBC	Triple DES with *cipher-block chaining*.
AES128-CBC	AES with *cipher-block chaining* and a 128 bits key.
AES128-CTR	AES with encryption with counter and a 128 bits key.
AES192-CBC	AES with *cipher-block chaining* and a 192 bits key.
AES192-CTR	AES with encryption with counter and a 192 bits key.
AES256-CBC	AES with *cipher-block chaining* and a 256 bits key.
AES256-CTR	AES with encryption with counter and a 256 bits key.
ARCFOUR	Alleged-RC4.
ARCFOUR128	Alleged-RC4 with a 128 bits key.
ARCFOUR256	Alleged-RC4 with a 256 bits key.
BLOWFISH-CBC	Blowfish with *cipher-block chaining*.
CAST128-CBC	CAST with *cipher-block chaining* with a 128 bits key.

Syntax:

```
SSHS>no ciphers <number>
```

Example:

```
SSHS>no ciphers arcfour256
SSHS>
```

2.3.17.6 NO CLIENT-ALIVE

Re-establishes the default value for the selected parameter. These parameters set the sending of *Client-Alive* messages generated by the server with version SSHv2 in charge of checking that the SSH connection with the client is still working. The possible options are:

Command	Function
INTERVAL	Time interval between two consecutive *Client-Alive* messages.
MAX-COUNT	Maximum number of messages sent without response.

2.3.17.6.1 NO CLIENT-ALIVE INTERVAL

Re-establishes the default value for the interval between consecutive *Client-Alive* messages. This is 15 seconds. This is the time period during which there is no data traffic between client and server.

Syntax:

```
SSHS>no client-alive interval
```

Example:

```
SSHS>no client-alive interval
SSHS>
```

2.3.17.6.2 NO CLIENT-ALIVE MAX-COUNT

Establishes the maximum number of consecutive *Client-Alive* messages without response to 3, the initial value for this parameter. If you want to disable the *Client-Alive* feature, set the **CLIENT-ALIVE INTERVAL** value to 0 seconds.

Syntax:

```
SSHS>no client-alive max-count
```

Example:

```
SSHS>no client-alive max-count
SSHS>
```

2.3.17.7 NO COMPRESSION

Establishes the compression availability to its default value. This allows both normal and delayed compression (*zlib@openssh.com*); i.e. sets a **YES** value.

Syntax:

```
SSHS>no compression
```

Example:

```
SSHS>no compression
SSHS>
```

2.3.17.8 NO ENABLE

Deactivates the SSH server. This is the server's default state. Through the **ENABLE** command the *host-keys* considered necessary are generated. However, when using the **NO ENABLE** command, none of the *host-keys* are deleted. Consequently, the same keys which were previously used are re-used to identify the device again. If you want to delete these keys as well, use the **NO HOST-KEY [TYPE]** found in the main SSH configuration menu.

Syntax:

```
SSHS>no enable
```

Example:

```
SSHS>no enable
SSHS>
```

2.3.17.9 NO EPHEMERAL-KEY

Changes the selected parameter to its default value. This command configures the RSA key generation in the server used for the session in SSHv1. The possible options are:

Command	Function
BITS	Resets the ephemeral key length to 768 bits.
REGENERATION-INTERVAL	Default period is 1 hour to generate the key.

2.3.17.9.1 NO EPHEMERAL-KEY BITS

Resets the default value for the SSHv1 ephemeral key length. This is 768 bits.

Syntax:

```
SSHS>no ephemeral-key bits
```

Example:

```
SSHS>no ephemeral-key bits
SSHS>
```

2.3.17.9.2 NO EPHEMERAL-KEY REGENERATION-INTERVAL

Sets the ephemeral key generation period to 1 hour. The counter starts whenever the key is used for the first time. It remains inactive for an indefinite period when not used.

Syntax:

```
SSHS>no ephemeral-key regeneration-interval
```

Example:

```
SSHS>no ephemeral-key regeneration-interval
SSHS>
```

2.3.17.10 NO KEEP-ALIVE

Disables the TCP *Keep-Alive* packet sending. This is enabled by default. The aim of these packets is to check that the TCP connection continues to be up and running.

Syntax:

```
SSHS>no keep-alive
```

Example:

```
SSHS>no keep-alive
SSHS>
```

2.3.17.11 NO KEY-EXCHANGE

Disables the key exchange algorithm indicated through its name, changing its status to not permitted. This is only relevant for SSHv2. By default, all implemented key exchange algorithms are enabled:

Command	Function
diffie-hell-man-group-exchange-sha1	This set of ephemerally generated key exchange groups uses SHA-1.
diffie-hell-man-group-exchange-sha256	This set of ephemerally generated key exchange groups uses SHA2-256.
diffie-hellman-group1-sha1	This method uses Oakley Group 2 (a 1024-bit MODP group) and SHA-1.
diffie-hellman-group14-sha1	This method uses group14 (a 2048-bit MODP group) and SHA-1.

Syntax:

```
SSHS>no key-exchange <name>
```

Example:

```
SSHS>no key-exchange diffie-hellman-group1-sha1
SSHS>
```

Command history:

Release	Modification
11.01.09	The *key-exchange* option was introduced as of version 11.01.09.
11.01.05.40.07	The *key-exchange* option was introduced as of version 11.01.05.40.07.
11.01.05.70.07	The *key-exchange* option was introduced as of version 11.01.05.70.07.
11.01.06.53.01	The *key-exchange* option was introduced as of version 11.01.06.53.01.

2.3.17.12 NO LOGIN

Deletes a method list defined using AAA for SSH authentication.

Syntax:

```
SSHS>no login authentication
```

Example:

```
SSHS>no login authentication
SSHS>
```

2.3.17.13 NO MACS

Disables the message authentication method indicated by name, changing its status to not permitted. This is only relevant for SSHv2. All the implemented message authentication codes (MACs) are enabled by default.

Command	Function
HMAC-MD5	MD5 *hash* algorithm.
HMAC-MD5-96	MD5 *hash* algorithm with 96 bit truncation.
HMAC-RIPEMD160	160 bits RIPEMD *hash* algorithm.
HMAC-SHA1	SHA-1 *hash* algorithm.
HMAC-SHA1-96	SHA-1 *hash* algorithm with 96 bit truncation.
HMAC-SHA2-256	SHA-2 *hash* algorithm with a digest length and a key length of 256 bits.
HMAC-SHA2-512	SHA-2 *hash* algorithm with a digest length and a key length of 512 bits.

Syntax:

```
SSHS>no macs <name>
```

Example:

```
SSHS>no macs hmac-sha1-96
SSHS>
```

Command history:

Release	Modification
11.01.09	The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command allows you to disable them.
11.01.05.70.06	The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command allows you to disable them.
11.01.05.40.07	The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command allows you to disable them.
11.01.06.53.01	The "hmac-sha2-256" and "hmac-sha2-512" Message Authentication Codes have been introduced on the SSH server and this command allows you to disable them.

2.3.17.14 NO MAX-AUTH-TRIES

Re-establishes the maximum number of client authentication attempts to its default value of 6.

Syntax:

```
SSHS>no max-auth-tries
```

Example:

```
SSHS>no max-auth-tries
SSHS>
```

2.3.17.15 NO MAX-CONNECTIONS

Sets the maximum number of SSH connections that the server can simultaneously maintain to its initial value of 4 connections.

Syntax:

```
SSHS>no max-connections
```

Example:

```
SSHS>no max-connections
SSHS>
```

2.3.17.16 NO PORT

After using this command, the server listens, expecting new connections, in the TCP port assigned by the IANA for the SSH protocol. This is port 22.

Syntax:

```
SSHS>no port
```

Example:

```
SSHS>no port
SSHS>
```

2.3.17.17 NO VERSION

Changes the version compatibility to its default value (**ANY**). This allows as many clients with SSHv1 as clients using SSHv2 to connect to the server.

Syntax:

```
SSHS>no version
```

Example:

```
SSHS>no version
SSHS>
```

2.3.18 PORT

Allows you to configure the SSH server TCP listening port. Default is 22, assigned by IANA. The client must firstly establish a TCP connection with the server, using the configured port. Subsequently, SSH negotiation between the client and server takes place.

If the listening port dynamically changes, none of the currently established connections closes.

Syntax:

```
SSHS>port <number>
```

Example:

```
SSHS>port 2690
SSHS>
```

2.3.19 VERSION

Selects compatibility with the SSH version required. By default, the server accepts client connections with versions SSHv1 and SSHv2. In this case, “SSH-1.99” is announced in the initial message. Use of SSHv1 is not recommended, but is maintained in case the client can only use this version. If you can see that all clients support SSHv2, then it’s a good idea to disable this feature. The possible options are:

Command	Function
1	Only compatible with SSHv1 (NOT RECOMMENDED).
2	Only compatible with SSHv2.
ANY	Compatible with SSHv1 and SSHv2 (default).

Syntax:

```
SSHS>version <option>
```

Example:

```
SSHS>version 2
SSHS>
```

If SSH version compatibility dynamically changes, none of the connections already established closes (even if the version is no longer compatible).

2.3.20 EXIT

Returns to the main SSH configuration menu (*SSH Config>* prompt).

Syntax:

```
SSHS>exit
```

Example:

```
SSHS>exit
SSH Config>
```

2.4 Steps to configure correctly SSH

The various steps you need to take to configure the SSH protocol are described below. Some commands that are not explained in this manual must be used. For further information on them, please see the corresponding manuals.

2.4.1 Configuring the server

The SSH server configuration process requires a set of commands to be run. Some pertain to SSH, while others not. We are assuming that the device already has an IP address through which clients can connect.

When configuring the application used by the client when connecting to the device's SSH server, the user should consult the information supplied by the application manufacturer.

Note

The steps described below correspond to a static server configuration. However, when configuring the server dynamically, the last two steps ("*Saving the configuration*" and "*Restarting the device*") are not necessary.

2.4.1.1 Creating the user and password

The SSH connection process requires client authentication in the server device. Therefore, it is essential for you to know which users are registered. If the device doesn’t have any, you need to create them using the **USER** command found on the main configuration menu. The only two exceptions to the above are as follows:

- Authentication is executed through a password and allows the AAA feature to manage it.
- Authentication is executed through a password and a RADIUS server, external to the AAA feature and with configured users, is used.

An example of how to create a user account in the device with its associated password is shown below:

```
*p 4
Config>user teldat password secreto
Config>
```

The configuration does not show the password in clear, it’s encrypted:

```
Config>show config
; Showing Menu and Submenus Configuration for access-level 15 ...
; ATLAS50 Router 9 24 Version 10.7.12-Alfa

   log-command-errors
   no configuration
   user teldat hash-password 17BC63FEA1F40F68187E6C435E412BB7
;
;
   network ethernet0/0
; -- Ethernet Interface User Configuration --
      ip address 192.168.121.33 255.255.0.0
;
;
;
;
;
   exit
;
   dump-command-errors
   end
Config>
```

For further and more advanced information (such as user management), please see manual *Teldat-Dm 704-I Configuration and Monitoring*.

2.4.1.2 Version compatibility

At this point, you need to decide which SSH versions are going to be compatible with our server. By default, versions SSHv1 and SSHv2 are enabled. If you don’t want to modify this, go on to the next step.

Since we don't recommend you using version SSHv1, if you assume potential clients support SSHv2 change the compatibility to the second version only. If you want to retain compatibility with both versions (in case some clients don’t support SSHv2), go to the next step.

```
Config>feature ssh
-- SSH protocol configuration --
SSH Config>server
-- SSH Server --
SSHS>version 2
SSHS>
```

2.4.1.3 Selecting the host-keys

After version compatibility has been verified, you must define the characteristics of the *host-keys* that are going to be used. If you want to insert *host-keys* from previous configurations, please see the **HOST-KEY [TYPE] INSERT** command.

If this is the first time the SSH is being configured, you need to generate the keys that match the server with the clients. When you enable the server, the necessary keys are automatically generated with a 1024 bit length. If you are enabling SSHv1 compatibility, an RSA1 key is generated. If SSHv2 is included, an RSA key is generated. If the operator agrees to use an RSA key for SSHv2 with a 1024 length, go to the next step.

If you wish to modify something in this process and not use automatic generation to save time, there are two possibilities:

If you want to maintain SSHv1 compatibility, then the *host-key* must be RSA1. If you want the length to be different from 1024 bits (e.g. 1536 bits), execute the following indicating the required length:

```
Config>feature ssh
-- SSH protocol configuration --
SSH Config>host-key rsa1 generate 1536
Generating public/private rsa1 key pair...
Please wait for a few seconds.
Key generation done.
        Public key:
            1536 35 15395870189579834143383499263836502730355937791246750842
        0766640556662090823310864817025228826352441749111783489552769911
        9639708956396044015448566573781658816138161042117522667342273473
        1978935240109989589240939176844801249252805213735013839068027601
        7515139729960831955943568873934949111279630668945662737719633606
        2052591783326421206270573383292451631263917245501815548196563428
        4064931566075988822771348850067450467840561217411044341822444224
        15692826628915662340199
        The key fingerprint is:
        8e:6a:55:63:f7:65:1b:f1:02:f4:bd:54:fb:4f:35:27

SSH Config>
```

As for the SSHv2 key, you can choose between RSA and DSA. By default, on enabling the server, a 1024 bit RSA key is generated. If you want to change the length, want a DSA *host-key* or want the server to have simultaneous RSA and DSA keys, execute one of the following processes (or both) paying particular attention to the RSA key length.

RSA:

```
Config>feature ssh
-- SSH protocol configuration --
SSH Config>host-key rsa generate 2048

Generating public/private rsa key pair...
Please wait for a few seconds.
Key generation done.

        Public key:
        ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA9R4+I16NwVedmbRRcT/GUS3XGTqk
        /4vquflLkvxxzgeBAuHQwvAw2/JuZJL+URL9wpLJp5kGI7ENweidqNhXmmETDyvs
        B45ZN7MxE4b096cOA8VnB1xecYYyfRKh85krCWDbwHQQpQZngHtIqrubo4E92wuS
        kBRParGltxEkDS9avAqj568fg2vXP9XrK0VaQNztGkApSuRdEXKatXH+oroGlNp5
        IULX5B/GC0bL1QBftzWIwkhzdGUQhPNPFZJ/kGSzo3kOCzoeeeacB7wZMk6OEUaq
        1PsQcRM0+g8jccZddMwkhtAx/taAu578YP1y4dIUUXgEp5i/EjKuqDshpw==

        The key fingerprint is:
        9f:b9:8d:af:d9:05:8a:4e:be:cb:3e:2b:8b:a1:8d:ad

SSH Config>
```

DSA:

```
SSH Config>host-key dsa generate
Generating public/private dsa key pair...
Please wait for a few seconds.
Key generation done.

        Public key:
        ssh-dss AAAAB3NzaC1kc3MAAACBAJ9hGEihxgSO1BZ3t/K/d0csAxC9VubAyKjV
        rkyAHl6u1snNQZM1u/HNA0vECH63T634Gpz7xzhWlce/CYl6OZbos6LWsHr/dlSR
        MwOC0kF1dvkI1vDTOtf+75TlhFbID3levC1Lo2JwrOqf3W+sBcXT3qv0uMVwGCil
        UGb4F7mJAAAAFQCoEF67qaVYwKsxHnj3cgRKxdu4wQAAAIBiA/4dt7nuFY41xEDD
        JW95WvamLZtfRH4E/f43Je6IsL6CYv0JhdodLULy9R8HOAQ2w+s7NC3gznhwyme9
        Sv4DzBIe5lUtf7DJigHAArXrbmRU45hJ6PoAmntrohmGh/nK+N9aPtRksoSTGU0N
        cn1M0Q2DV8CalONcqqJMeNNFhQAAAIBg24fhfSSMBazImq6Vjr9Kqt/Qtsubws4I
        lvBv14ffMpExeUS+/BAKj0GMrfFL75eDU3L8axdDVqruzqBAr2S2Ah3r+UZobQDy
        VxLhtiLtHygpq8NtjZD0+DDsmobQNaPbFAblUqH8hr5H8hBq6HsVAt6cFHhYh6ue
        lJCiilC8Ag==
        The key fingerprint is:
        6c:29:36:91:58:b7:bf:15:1b:72:54:3e:93:bc:cb:e4

SSH Config>
```

Key generation is a long process. It can last several minutes, depending on the device, the type of key that is going to be used and its length. However, this process is only necessary once per device.

2.4.1.4 Activating the server

Here, we enable the server so that, when the configuration has been saved and the device rebooted, the SSH server starts up and begins to listen to SSH clients. If this is dynamically done, there is no need to save the configuration and reboot the device. Depending on what you did previously regarding *host-key* selection, keys will be generated or not. Assuming that no *host-key* has been previously generated, the results would be similar to those shown below:

```
Config>feature ssh
-- SSH protocol configuration --
SSH Config>server
-- SSH Server --
SSHS>enable
Hostkeys for SSHv1 and SSHv2 not found in config
Generating public/private rsa1 key pair...
Please wait for a few seconds.
Key generation done.
        Public key:
        1024 35 15432786495890011256590334136108871371683443719953266934
        1669735536131263850541188761930253773041643800977721757100219858
        0654115081071493764373252615104438669848742056849742709313080675
        1196133356243526157859512801760325744289280950757198892914694769
        1949421675284134332362671477102618739031477877190305796041081

        The key fingerprint is:
        07:fe:99:4c:c4:e9:bd:f8:7f:c4:e0:8a:c8:94:8a:4e

Generating public/private rsa key pair...
Please wait for a few seconds.
Key generation done.
        Public key:
        ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEA1IyO4HqZk4WghKgm9aE+1puDgCyS
        hmzs8vpaf0SHdzZjKHVAbGv3UnNP7DDUUsc4LnhVbcfnmNnCJnb1yCs725E2awuH
        zMc/o0B4jPmnL8H/qk8dp65vjDXj5YJD6F7824Lasb4V2VjCvjVJw589ZDG/qvLU
        JFpeGJPJB9uiUp8=

        The key fingerprint is:
        53:30:ff:e4:1f:04:25:2a:df:34:d1:ac:19:33:04:fd
Hostkeys have been generated. Remember to save config!
SSHS>
```

In the example, the RSA1 and RSA keys have been generated as they weren't present in the configuration. There are two possible keys, and only those needed are generated.

If the keys the server needs have already been generated, you will get the following:

```
SSH Config>server
-- SSH Server --
SSHS>enable
SSHS>
```

2.4.1.5 Advanced server options

The SSH server default configuration can be considered permissive, meaning that the characteristics or features implemented are configured as available. In the server menu (SSH>), you can specify other values for each of the variables, normally imposing more restrictions. However, one of the parameters you normally need to configure is client authentication. If you don't want to change anything, go to the next step.

When it comes to client authentication, the first step is to create a user account with a password. If you want authentication without a password, or you choose other authentication mechanisms, you need to insert the public keys in the configuration (for SSHv1 or SSHv2). For further information, please see the **AUTHENTICATION PUBLIC-KEY** and **AUTHENTICATION RSA** commands.

A simple example is included herein. If, in the client application, you want to authenticate through and SSHv2 *Public-Key* and you have the following (all one line):

ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAIBOcV+xGKKNuQ4GXEMvbJ09ib9kmArxHrlvmtXtnZdnyW2KOLcCIbwWWZoqr6O2LPuwrhE/K2/nKe0Q+7U3K8aVmNLEtv/0DR61dYZGuLsto4nX8XbwSTLg3KF3EexE5AzP6ETM3tMSpd0OoJPYRxB0xUJGqrO/F+lJYu5QO0PtqQ== rsa-key-20080206

We first truncate it:

ssh-rsa

AAAAB3NzaC1yc2EAAAABJQAAAIBOcV+xGKKNuQ4GXEMvbJ09ib9kmArxHrlv

mtXtnZdnyW2KOLcCIbwWWZoqr6O2LPuwrhE/K2/nKe0Q+7U3K8aVmNLEtv/0

DR61dYZGuLsto4nX8XbwSTLg3KF3EexE5AzP6ETM3tMSpd0OoJPYRxB0xUJG

qrO/F+lJYu5QO0PtqQ== rsa-key-20080206

Then enter it under name "cliente1":

```
SSHS>authentication public-key
-- Public-Key configuration in SSH server --
SSHS PK>key cliente1
SSHS KEY cliente1>insert
Enter the public key (type + base64)
<cr> to escape
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIBOcV+xGKKNuQ4GXEMvbJ09ib9kmArxHrlv
mtXtnZdnyW2KOLcCIbwWWZoqr6O2LPuwrhE/K2/nKe0Q+7U3K8aVmNLEtv/0
DR61dYZGuLsto4nX8XbwSTLg3KF3EexE5AzP6ETM3tMSpd0OoJPYRxB0xUJG
qrO/F+lJYu5QO0PtqQ== rsa-key-20080206
SSHS KEY cliente1>
```

Once it's done, we associate the key with the relevant user name (e.g. the user you created at the beginning, "teldat").

```
SSHS KEY cliente1>user teldat
SSHS KEY cliente1>
```

2.4.1.6 About Telnet

One of the main aims of SSH is to provide security when accessing a device via console. A less secure mechanism, Telnet, is also available in the device. Depending on the case, it's a little odd to add a high security level using SSH to grant access through a console while letting a client access the same device via Telnet. If, however, authentication with a password in SSH is deactivated, the possibility of using Telnet is still available for teldat devices.

If security is important and you don't want the device to be accessed via Telnet, set the maximum number of Telnet sessions to 0, as shown below:

```
*p 4
Config>set telnet
-- Telnet user configuration --
Telnet config>set ?
  max-telnets    Maximum number of telnet sessions you can open
  port           Set port number
Telnet config>set max-telnets ?
  <0..20>    Value in the specified range
Telnet config>set max-telnets 0
Telnet config>
```

2.4.1.7 Saving the configuration

Once everything has been configured, you must save the configuration to keep the changes when you reboot. In the following example, the SSH_SERV_CFG file is created first and the configuration is then saved.

```
Config>set file-cfg
Config Media: Flash only
 A:                       GENERAL        392   01/16/08   10:18   Flash
Current config: GENERAL
File name [GENERAL]? SSH_SERV
Config>save
Save configuration (Yes/No)? y
Building configuration as text... OK
Writing configuration... OK on Flash as SSH_SERV
Config>
```

The saved configuration can be viewed by executing the **SHOW CONFIG** command:

```
Config>show config
; Showing Menu and Submenus Configuration for access-level 15 ...
; ATLAS50 Router 9 24 Version 10.7.12-Alfa
   log-command-errors
   no configuration
   user company hash-password 17BC63FEA1F40F68187E6C435E412BB7
;
;
   network ethernet0/0
; -- Ethernet Interface User Configuration --
      ip address 192.168.121.33 255.255.0.0
;
;
;
;
;
   exit
;
;
;
;
;
   set telnet
; -- Telnet user configuration --
      set max-telnets 0
   exit
;
;
;
```

```
;
   feature ssh
; -- SSH protocol configuration --

      Server
; -- SSH Server --
         authentication public-key
; -- Public-Key configuration in SSH server --
            key cliente1
               user company
;
               add "ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAIBOcV+xGKKNuQ4GXE"
               add MvbJ09ib9kmArxHrlvmtXtnZdnyW2KOLcCIbwWWZoqr6O2LPuw
               add rhE/K2/nKe0Q+7U3K8aVmNLEtv/0DR61dYZGuLsto4nX8XbwST
               add Lg3KF3EexE5AzP6ETM3tMSpd0OoJPYRxB0xUJGqrO/F+lJYu5Q
               end "O0PtqQ== rsa-key-20080206"
            exit
;
         exit
;
         enable
      exit
;
   exit
;
   dump-command-errors
   end
Config>
```

As you can see above, the generated *host-keys* appear not to have been saved in the configuration. This is because they are not shown through **SHOW CONFIG**, but form part of the configuration file. By obtaining the file through FTP, or executing the **LIST HOST-KEY ALL** command found in the SSH menu, you can check that the keys have been saved. Regarding the public key used for authentication purposes, this is optional and depends on the client's keys.

2.4.1.8 Restarting the device

Basic static configuration is now complete. In order to activate it, the device must be restarted.

```
Config>end
*restart
Are you sure to restart the system(Yes/No)? y
  Done
Restarting. Please wait ................................................
```

Once completed, the SSH server is operating and waiting for new clients. If it didn't have a user account, you can now see the device asks for a user and password to access the router through the console. Please note that, in the example used, the user is “teldat” and the password “secreto”.

If you want to check that the server is listening:

```
*p 3
Console Operator
+protocol ip
-- IP protocol monitor --

IP+tcp-list
LOCAL ADDR      LOCAL PORT  REMOTE ADDR     REMOTE PORT  STATE
--------------  ----------  --------------  -----------  --------
0.0.0.0         21          0.0.0.0         0            LISTEN
0.0.0.0         23          0.0.0.0         0            LISTEN
0.0.0.0         22          0.0.0.0         0            LISTEN
IP+
```

In the TCP connections table, found under IP protocol monitoring, you can see that the device is listening for new connections in port 22. If you have changed the listening port for an non-standardized port in the SSH configuration, this one will appear.

Chapter 3 Monitoring

3.1 Monitoring SSH connections

A monitoring command that shows SSH connections in progress is available. To execute this, enter the following command from the console prompt (+):

```
*p 3
Console Operator
+system ssh
Time unit: minutes
  ID  USER   LEVEL     IP ADDRESS:PORT   CONNECTION-TIME INACTIV-TIME  IDLETIME TIMEOUT
-------------------------------------------------------------------------------------
  0  teldat 15          Local Console  06/02/08 16:15:29        3        0        0
  1  teldat 15     192.168.51.155:4851 06/02/08 16:30:48        0       10        0 *
+
```

This command was executed via SSH from a remote console. The information displayed, from left to right, is:

- A user number, different for each console session.
- User name; name introduced for authentication.
- User privilege level.
- Source IP address and port (or “Local Console”, if this corresponds to the local console).
- Time period when this connected.
- Console idle time, with no data exchanged.
- Maximum idle time permitted. 0 if there is no limit.
- Maximum session time permitted. 0 if there is no limit.
- Asterisk in the line that corresponds to the console session that executed the command.

Regarding idle time, if the time period set in **SET INACTIVITY-TIMER** times out, the console session automatically closes.

Chapter 4 Annex A

4.1 Third Party Software

When it comes to TLS negotiation, CIT uses the OpenSSL library code.

Please see a copy of the OpenSSL license below: