Sysgem AG
Casa Bergenia
Postfach 159
CH-7031 Laax
Switzerland

Tel: +41 (0) 81 921 6853

office@sysgem.com
www.sysgem.com

Proxy Server Design

Sysgem Enterprise Manager

Authorization sheet

Authors			
Function	**name**	**date**	**signature**
Project Manager			
Design Leader	Ben A L Jemmett		

Approvals			
Function	**name**	**Date**	**signature**
Customer – Technical Approval			

Reviewers			
Function	**name**	**Date**	**signature**
Sysgem Development			
Reseller – Quality Assurance			
Customer – Technical Review			

Authorization			
Function	**name**	**Date**	**signature**
Sysgem Managing Director			

Contents

Document history

Revision	Name author(s)	Revision description	Revision date
1.0	Ben A L Jemmett	Initial version of the document	October 2, 2007

1 Scope and Requirements

A need has been identified at several customer sites for Sysgem Enterprise Manager (SEM) to operate through a proxy server. This requirement has three aspects:

- A practical aspect: in some secure networks, it is not possible to open ports through firewalls in order for SEM management consoles to communicate with the SEM agent processes on managed machines.
- A security aspect: it desired from a security viewpoint that no access should be made directly to privileged processes on managed machines.
- An auditing aspect: some customers have asked that we provide a method for auditing the scripts run on managed machines, in addition to the actions taken by individual users (e.g. the SAcM Audit Trail mechanism.)

The most complete list of requirements is shown in this example from one customer:

- Management machines should only be accessing non-privileged processes directly.
- All messages between management machines and managed machines to be subject to:
    - Auditing
    - Authentication
    - Authorization
    - Validation
- The ability should be provided to tunnel connections through SSH sessions.
- The ability should be provided to work through multiple levels of proxying, for situations where multiple firewalls may sit between the management machines and the managed agents.

Additionally, Sysgem have identified the following possible requirement:

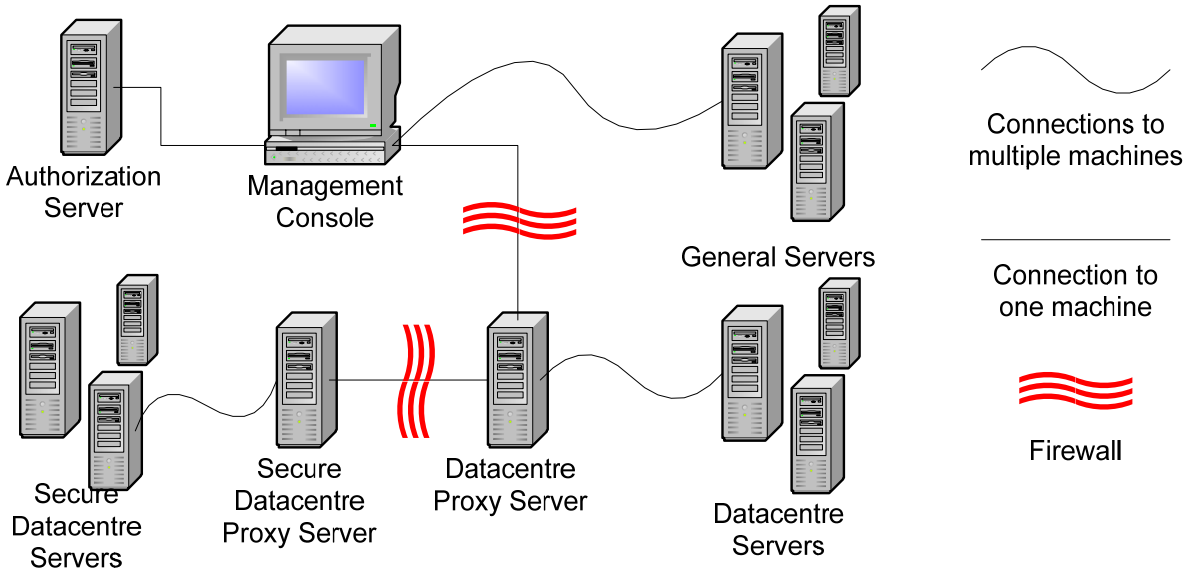
- The proxy server should be able to proxy connections back from the managed agents to the SEM Authorization Server.

This is due to new features added in both SEM 2.1 – the Installation Key security method requires a connection to be made to the Authorization Server at installation time – and SEM 2.2, where Sysgem databases and other shared data are accessed via the Authorization Server. This last requirement is not yet a hard requirement; it is possible that such functionality is not required from the agent machines.

2 Overall architecture

2.1 Network layout

An example network layout is shown below, in order to illustrate the requirements when several layers of proxy are present:



In this example system, the network is divided into three sections:

- The management network, in which the SEM Authorization Server, Management Consoles and other general-purpose servers reside
- The datacenter network, which contains production servers and is separated from the management network by a firewall.
- The secure datacenter network, contained within the datacenter network but again separated from the less-secure area by a firewall.

2.2 Software design

The proxy server software shall be a non-privileged process, running on Windows and potentially Linux (or another Unix platform). This process accepts incoming connections from SEM Management Consoles, receives messages using the standard SEM Universal Transport protocol, and forwards them on to SEM Agents running on desired target machines. Replies from the agents are similarly received and forwarded back to the management console.

The connection to the proxy shall be made as a standard TCP connection, and the proxy's connection out to the agent shall be made either as a standard TCP connection directly to the agent, a standard TCP connection to another proxy server, or an SSH-tunnelled connection to either an agent or another proxy.

In addition to forwarding messages, the proxy server will optionally subject the messages to audit logging and/or authorization checks and validation.

2.3 SEM Configuration

Since agent definitions are currently stored at the SEM Authorization Server on a per-user basis, it would seem incongruous to keep the agent definitions for a proxy at the proxy server since these would then be global to all users. Instead, the information regarding proxies to use shall be stored alongside the current agent definitions.

Each agent definition currently consists of a name, address (either as an IP or an address resolved through a name service), port number and ancillary information such as description, icon etc. This definition shall be extended with a "connection type" field, set to "direct connection" by default, and connections to that agent will be made in the same fashion as currently.

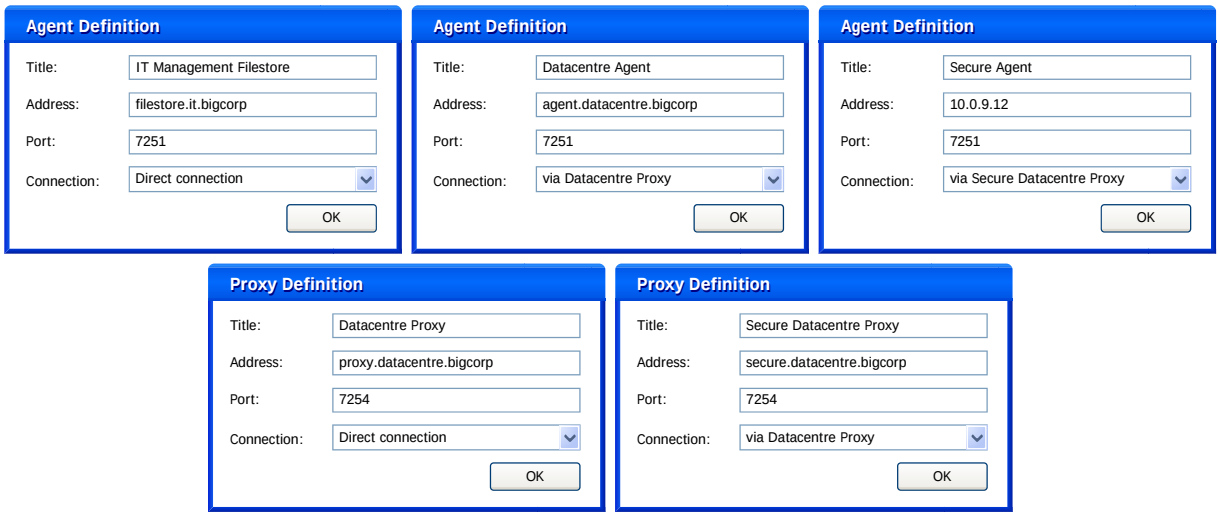
In addition, a new "proxy definition" shall be introduced, which consists of the same fields as an agent definition (except for the ancillary fields, which are purely for end-user convenience). Proxy definitions will not be presented in any agent lists in the GUI.

If any proxy servers are defined, it will be possible to select them in the "connection type" field of a proxy or agent definition. Once this is done for an agent, connections to that agent will proceed as follows:

1. Establish connection to the address and port configured for the selected proxy
2. Provide identifying information (SEM username, OS username, computer name, Authorization Server address, etc.)
3. Provide the details of the agent as entered in the agent definition. The proxy server will establish a connection to the specified address (having resolved the address locally to the proxy server if required.)
4. Messages will be now passed between the management console and agent as normal.

Note that this allows for connections to be established via several layers of proxy; the definition of Proxy B may specify a connection type of "via Proxy A", in which case step 1 above consists of "connect to Proxy A and follow steps 2 and 3 to establish a connection to Proxy B."

This configuration details required for the example network are illustrated below.



2.4 Proxy configuration

In addition to the SEM agent definitions, the proxy server itself requires some configuration data:

- Port to listen on
- Optional address restrictions for incoming connections (listen only on a specified IP address, and/or accept connections only from specified addresses.)
- Optional list of addresses available for outbound connections – by default, any address the SEM Management Console requests can be connected to, but in some situations this should be restricted.
- Optional connection details for certain addresses – e.g. “addresses in this range are to be connected to by connecting to this SSH server with this certificate and establishing a secure tunnel.”

It is suggested that these details are kept in a text file on the proxy server, so that they can be edited by the system administrator without requiring access via SEM itself. However, an interface could be provided to the configuration through SEM; for instance, an Edit Configuration File button could be present in the Proxy Definition editor.

2.5 AAAA (Authentication, Authorization, Accounting and Auditing)

The exact form the AAAA aspects of the proxy service will take is to be discussed further with specific customers, to gain a better understanding of the specific requirements. The following general notes may provide a starting point:

- Authentication: SEM currently provides an end-to-end authentication mechanism, in the form of security keys. These are available in two types: the standard Security Key, which allows new machines to be added to the SEM network by any user knowing the key(s) in use, and the Installation Key which adds an additional layer of security by not revealing the keys in a reusable form.

  A decision to be made is whether the proxy server should participate in this handshaking process at all. The current handshake protocol is a two-way challenge/response system between management console and agent; when a proxy server is introduced in between, several options are possible:

  1. The protocol remains unchanged, and the proxy server simply relays the authentication messages between console and agent. In this scenario, the proxy’s security keys are not checked unless privileged (e.g. configuration) functions are invoked upon it.
  2. The management console handshakes with each component it connects to in turn: first the proxy (or each proxy in the chain if several are in use), then the agent. In this scheme, the security keys on the proxy are always checked, and there must be at least one key shared by the management console and proxy, and the management console and agent.
  3. The management console handshakes with the proxy, which in turn handshakes with the agent before allowing access. In this scenario, the management console and agent need not share a security key; however, the management console and proxy must share a key, and a (possibly different) key must be shared by the proxy and the agent.

- Authorization: Upon each connection, the proxy server will be provided with the SEM username, authorization server details (address, domain ID if configured), OS username and computer name of the user making the connection. This information can be used to decide if a user is authorized to access the given agent or functionality[*] and accept or reject the transaction accordingly. The user authorization information is held on the proxy and edited either directly or through SEM.

  An additional possibility is to have the proxy server retrieve user authorization information from the Authorization Server, as well as check that a certain user session is registered with the Authorization Server that the proxy is configured to trust. This would allow security against unauthorized Authorization Server installations being used to connect to secure agents through a proxy, even if Installation Keys were not in use.

- Accounting: The proxy will be able to maintain a record of communications with each agent, listing the date and time of each operation alongside the user name, computer name etc. This information will be based on the sessions established with the agent – see SEM's Agent Connections window for an example.

- Auditing: Since the summary of communications with each agent will only show an overview of windows in use, it may be required to audit actions more fully. This could be achieved by storing complete details of every communication between the management console and agent

  A point to be aware of when auditing actions in this level of detail is that all information transmitted to the agent will be visible in the log – this includes new passwords transmitted as part of a Reset Password operation, for instance[†]. Accordingly, the auditing log will need to be properly secured.

  The complete audit log is likely to grow rapidly, and so a rolling size limit could be employed to limit growth – e.g. storing only the last six months of audit data.

2.6 Validation of Scripts

More information on the exact requirements will be needed from the customer before the feasibility of this requirement can be determined. Since the scripts we execute are as powerful as any other programs, attempting to validate their behaviour before executing them is generally not possible[‡].

If the scope of this requirement is sufficiently defined (e.g. disallow any scripts containing certain strings) then a less complete method of validation could be implemented to provide some security against malicious scripts, but Sysgem would not be able to guarantee such a validation mechanism to be free of potential loopholes.

---

[*] The practicality of enforcing authorization on a per-user basis at the basic operation level is debatable. SEM provides a flexible set of permissions at the management console; however these reflect operations performed at a lower level on the agent (e.g. "read/write file", "get file security" and "execute script"). In addition, most (if not all) SEM operations could also be accomplished via scripts, on which the SEM management displays depend. The most practical form of authorization would be to enforce a restriction on which users can access which agents.

[†] All data sent between SEM and its agents or the Authorization Server can be encrypted – the levels of security available are "none", "encoded" (fast, but only offering a moderate level of security, enough to obscure data from casual snooping) and "encrypted" (slower but more secure, using the Blowfish algorithm). The default is the faster "encoding" setting.

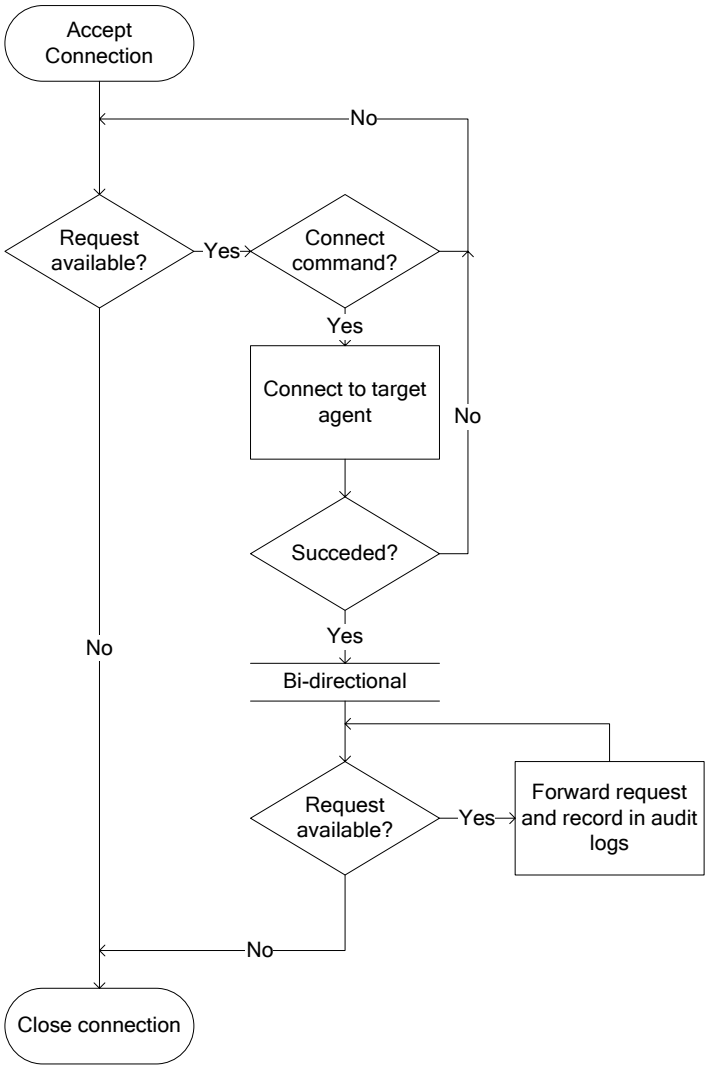
[‡] At a theoretical level, since the scripting languages employed by SEM are Turing-complete, validation of scripts could well be a case of the halting problem – which has been shown to be undecidable in general. Such computability theoretic discussion is probably outside the scope of this document, but should be noted as a practical obstacle to being able to guarantee the validity of user-supplied scripts.

3 Design and implementation notes

3.1 Protocol

The standard SEM Universal Transport will be employed, so that the proxy server integrates cleanly into the existing data stream. The currently-defined set of agent transport operations will need to be supplemented with a Connect op-code, which takes as parameters the address (numeric IP address or host name) and port of the target machine.

Upon receiving this, the proxy server will attempt to establish a connection to the specified machine, and then return either a successful result code or a failure result code along with a textual error message. If the request completed successfully, the proxy server is now passing messages between the two connections and will no longer respond to any requests on its own behalf. An outline of the per-connection processing taken by the proxy is shown below:



The proxy may also respond to some of the other defined op-codes if they may be useful for performing remote management of the proxy – e.g. the read / write file operations for accessing the configuration file, and the get machine information operation to provide information on the proxy in a style similar to the Machines display. This will be decided in the final design.

3.2 Client Connection Procedure

The client currently connects to agents using the following procedure:

1. Resolve hostname to IP address if required
2. Create and initialise socket, and connect to remote host
3. Retrieve remote host status to ensure agent is responding
4. Check remote host for domain membership
5. Perform two-way challenge/response authentication of relevant security keys

This will need to be modified by replacing steps 1 and 2 with the following outlined procedure:

1a. If a proxy is specified in the agent definition, push the specified agent address onto an address stack and use the proxy's address as the destination address. Otherwise proceed as before.

1b. If a proxy is specified in the current proxy definition, push the current proxy address onto the address stack and use the next proxy's address as the destination address. Repeat until no further proxies are specified.

2a. Create and initialise socket and connect to the destination address as determined in step 1.

2b. Whilst there are addresses on the address stack, retrieve the top-most address and send it to the proxy in a Connect transaction.

Once these steps are followed, a chain of proxy connections will be established between the client and the target agent.

3.3 SSH Tunnels

The proxy server is required to be able to connect to agents (or further proxy servers) over SSH tunnels. This will be accomplished by keeping, for each SSH-contactable agent, the following information:

- Definition name
- SSH server name or address, and port
- SSH username and corresponding certificate file or password
- Remote server name or address, and port

When a connection request is made for a certain target name, the name is first looked up against the list of SSH tunnel definitions. If a match is found, the tunnel is first established between a random local port and the specified remote port, and the connection attempt is made to the random local port.

SSH connections will be made using a suitable SSH client application or library. It is likely that PuTTY shall be used for this purpose, since it is small, self-contained, reliable and available for both Windows and Unix platforms under a suitable license for inclusion into commercial software. Connections will either be made using PuTTY as an external application, on a one-instance-per-target-agent basis, or by incorporating the PuTTY components into the server directly and establishing tunnels on a one-instance-per-SSH-server basis. This decision will be made once the architecture of the PuTTY code has been examined.