

# PINsafe Multifactor Authentication Solution

## Technical White Paper

### Abstract

PINsafe is a flexible authentication solution that offers a wide range of authentication models. The use of the patented one-time code extraction protocol means that PINsafe can offer both single and multi-channel and single and multi-factor authentication solutions.

### Introduction

This paper describes the principles and technology behind the PINsafe Multifactor Authentication Solution. It looks at the principles and advantages of multi-factor and multi-channel authentication and how PINsafe implements these approaches to authentication.

### Overview

PINsafe is a multi-factor authentication system. The core of the solution is the Swivel one-time code (OTC) extraction protocol whereby a user is sent a security string, the user then combines this security string with their PIN number to derive a one-time code. They then use this one-time code to authenticate themselves.

The strength of this system is that the user needs both the security string and their PIN in order to authenticate. The one-time code extraction protocol is simple to use, the PIN determines which characters are to be used and in which order, for the one-time code.

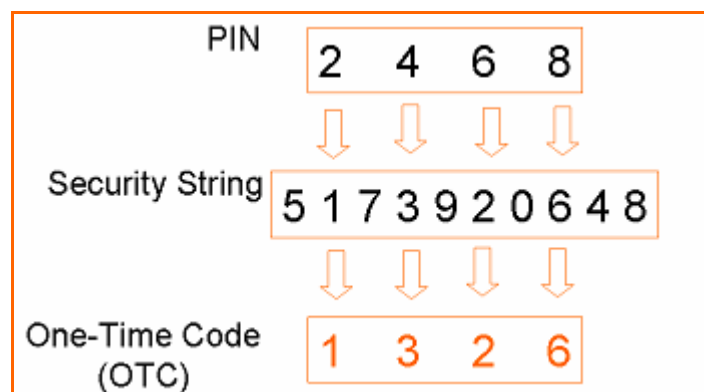


Figure 1: OTC Extraction Protocol

The security string can be presented to the user in a number of ways, in an SMS message, via a Java Applet or as an obfuscated image (known as a TURING image), as show in figure 2.<sup>1</sup>

<sup>1</sup> The obfuscation of the image provides defense against Optical Character Recognition (OCR) attacks.

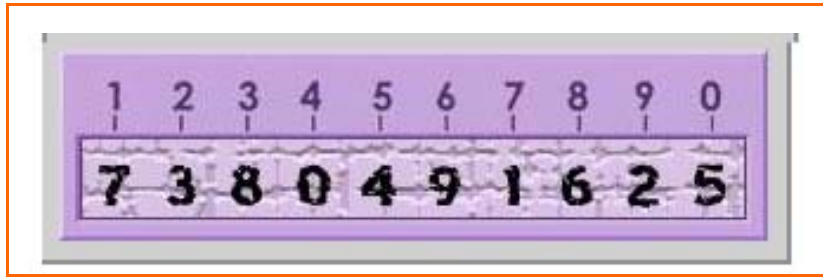


Figure 2: Security String as an Obfuscated Image

There are a number of advantages to this approach and these include:

- The one-time code that the user enters is different for every authentication which provides defence against key-logging attacks, and many simple man-in-the-middle and phishing attacks.
- The user never enters their PIN to authenticate, again providing defence against the attacks listed above.
- As authentication requires two elements, the security string can be sent via a different channel to the authentication request, providing defence against man-in-the-middle attacks.
- The delivery of the security string can be tied to a specific device, eg a mobile phone, providing a two-factor authentication solution.

## PINsafe High Level View

The PINsafe server sits at the centre of the authentication system. It manages the user data, creating and managing credentials as required, enforcing authentication policies and distributing security strings.

Agents submit authentication requests to the PINsafe server, these agents are, or are deployed on, the resource that is being protected by PINsafe. Agents may be a SSL VPN server or a piece of agent software installed on a web site or web application; or indeed any application.

PINsafe can deliver security strings to users in a number of different ways:

- An agent can request a security string on behalf of the user. The security string can be returned to the agent or delivered to the user via the Transport module.
- A user can request security strings directly from the server. The security string can be returned to the user or delivered to the user via the Transport module.
- The PINsafe server can automatically send a security string to the user via the Transport Module.

It is this splitting of the delivery of security strings and the submission of authentication requests into two separate logical channels that delivers the advantages described above.

PINsafe has a user database in which it stores the data relating to the user's PINsafe Account. It can create all the PINsafe accounts that an enterprise needs by synchronizing with an enterprise's existing user database, typically an Active Directory installation.

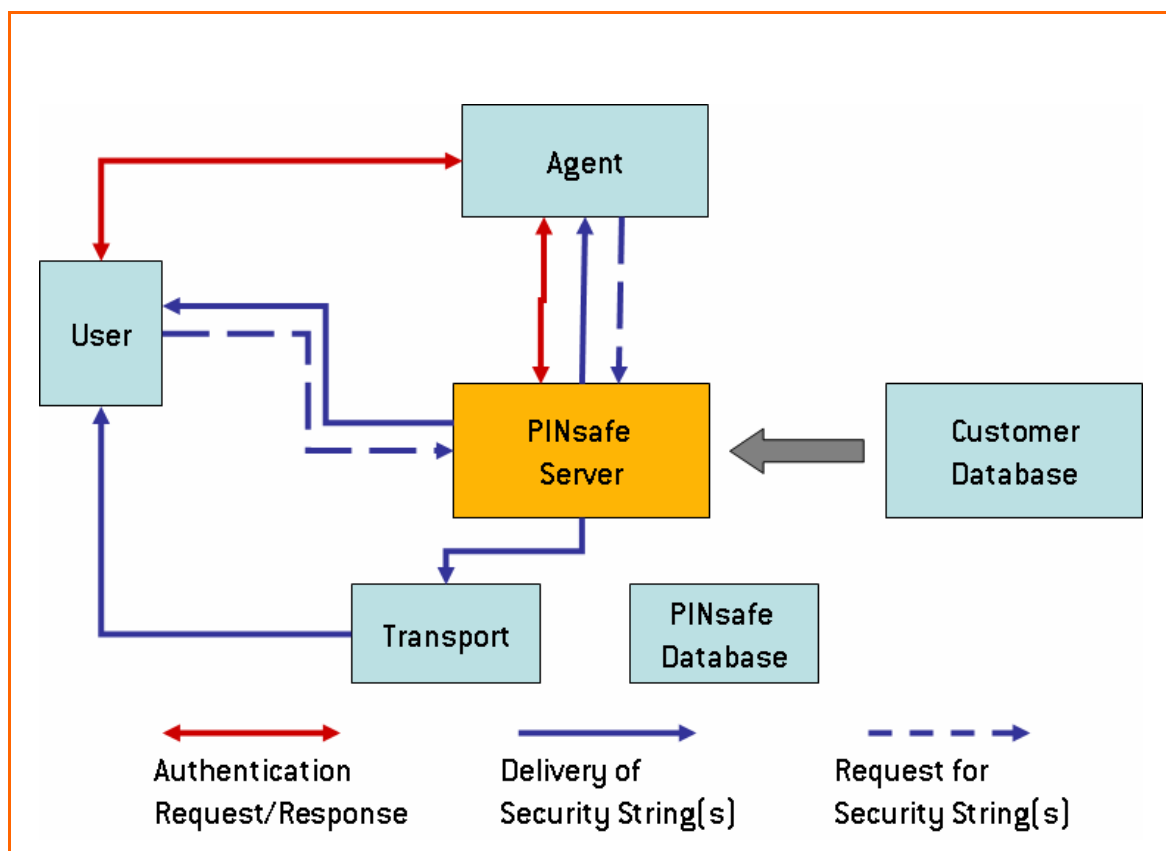


Figure 3: High Level view of PINsafe

With different agents, PINsafe server configurations and transports, this basic model can provide a range of different authentication models; two examples are described below.

### Single Channel, Radius, On-Demand, Active Directory

This is typical for a VPN solution. PINsafe synchronises with the Active Directory and creates user-accounts in the PINsafe database, as required.

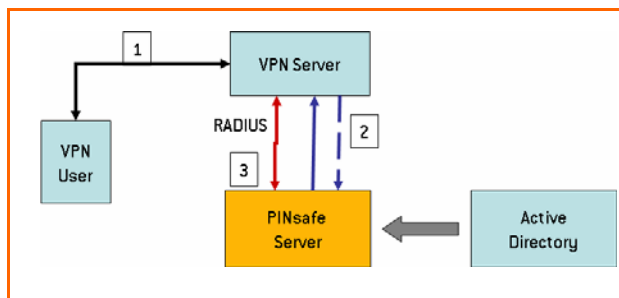


Figure 4: High Level view of PINsafe with VPN

The agent in this case is the VPN server. The VPN server's log on page can be modified to include a Turing image.

1. The user enters their username and initiates an authentication session.
2. The agent requests and receives a security string from the PINsafe server and displays the security string as a Turing image.
3. The user derives and enters their one-time code. The VPN server submits an authentication request, using the RADIUS protocol, to the PINsafe server. PINsafe responds, via RADIUS, with either "pass" or "fail".

### Two Channel, Agent-XML, Automatic, XML Repository

This model maybe more typical of using PINsafe to protect a web site or web application. Users are entered into an XML repository. PINsafe synchronises with this database and creates user-accounts in the PINsafe database, as required.

The agent in this case is a piece of software residing on the website. This software examines the resources that a user is trying to access and then requires them to authenticate as appropriate.

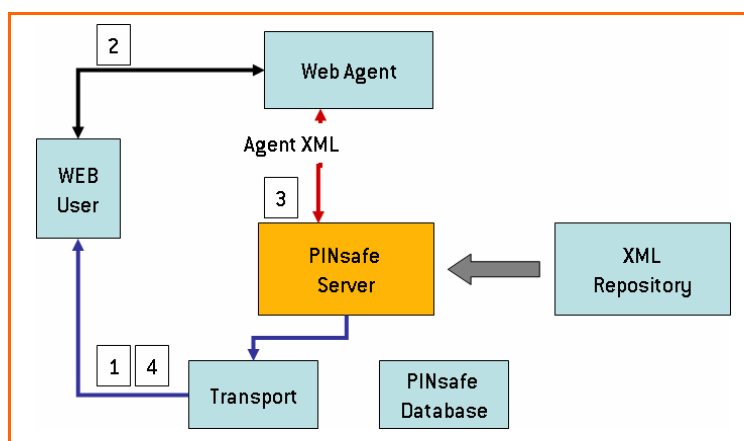


Figure 3: High Level view of PINsafe with Web Application

1. As a result of account creation or a previous authentication attempt, the PINsafe server has sent the user, via SMS (transport) a security string.
2. The agent asks the user for their username and one-time code.
3. The user enters their one-time code, the Agent uses the Agent-XML API to make the authentication request and the PINsafe server responds with pass or with an error message.
4. In Automatic mode the user is automatically sent a new security string.

## Security String Delivery

The two examples above show how PINsafe offers strong authentication by requiring the user to have a security string and a PIN in order to authenticate.

There are a number of ways that security strings can be:

- Generated
- Transported
- Displayed

This section describes the different methods to help to show PINsafe's flexibility and the different ways that it can strengthen authentication.

### *On-Demand or In Advance*

Security strings can be sent only as and when an Agent requires a user to authenticate or they can be sent to the user in advance so the user has them ready at the time of authentication.

The default behavior for SMS operation is for the user to be sent a security string in advance, after each authentication attempt, this has the advantage that the user always has a valid security string in their SMS inbox when they come to authenticate. In this model the PINsafe server initiates the sending of a security string to the user after each authentication attempt, whether the attempt was successful or not. Security strings that are delivered in advance are valid until they are used or until more security strings are requested.

An alternative model is for the security string only to be sent to the user when they are attempting to authenticate. In this model the Agent user requests a security string on the user's behalf y making a request to the PINsafe server Security Strings module.

If a security string is requested as in the on-demand mode, it has a limited validity period; once this period has expired the user cannot authenticate using that security string.

The advantage that on-demand brings with it is tighter control of the security by the agent. This can be used as a defense against phishing attacks as security strings will only be sent out when the real site deems it necessary.

This defense is strengthened as it enables the agent to add context information to the security string. As the authentication session is under the control of the Agent, the Agent knows what the user is authenticating for. If, for example, the user is authenticating to authorize a purchase this information can be included in the security string sent to the user, eg the SMS could state "Security string for \$340 purchase".

Generally speaking when the TURING image is used in on demand mode and the PINsafe mobile client (Swivlet, see later in this paper) requests security strings in advance, however this is not mandated by the architecture of the server.

### *Two Channel vs Single Channel*

Security strings can be delivered to the user as part of their log-on dialogue, eg incorporated in a VPN log-on page, or delivered to a mobile phone or other device. In the first case the security string is sent via the same channel over which the authentication takes place; hence single channel. In the second case the security string is sent via a different route; hence two channel. Two channel is generally more secure because in order to determine someone's PIN, you need to know the security

string that they have been sent and the one-time code that they have entered. Sending these two via separate channels makes intercepting them more difficult.

In single channel mode the security strings are sent to the agent that requested them, ie the TURING image is sent to the agent that requested the security string and that will then go on to make an authentication request.

In two channel mode the security string is delivered directly to the user via a separate logical channel. This is done by sending the security string via the transport module or by the user retrieving the strings directly from the server. An example of using the transport module is the sending of security strings via SMS. An example of the user retrieving the security strings directly is the PINsafe systray utility that is a small client that resides in the users System tray which can retrieve and display TURING images.

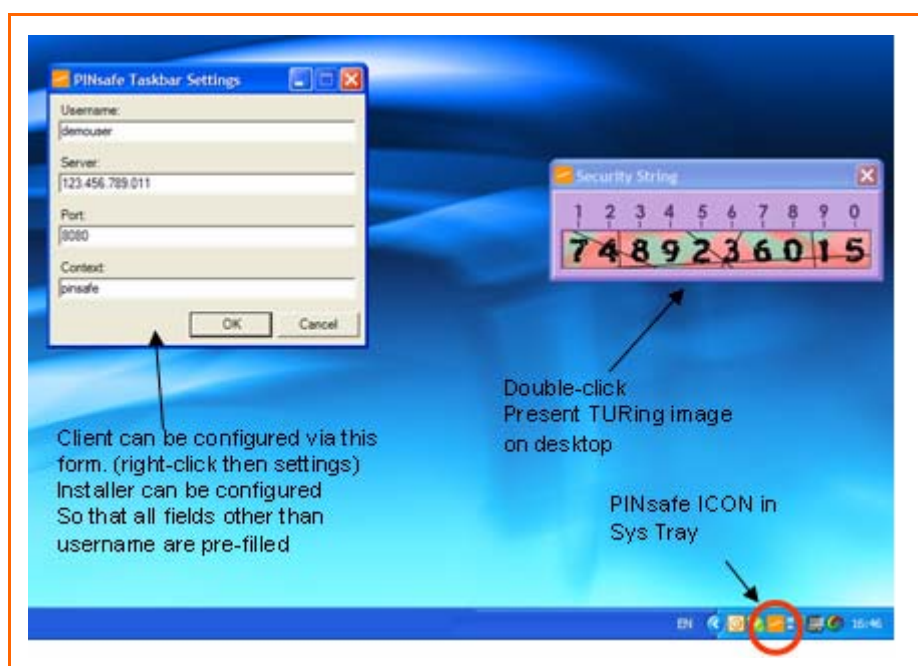


Figure 6: PINsafe Systray Utility

Use of clients such as the systray client does mean that the security strings are delivered over a separate logical channel, but the same physical channel, ie the internet. True two channel is achieved where the security string delivery channel is also physically separate. This can be achieved by using the Swivel mobile application (Swivlet) that retrieves security strings over GPRS, or by using the a SMS transport module to deliver security strings via the SMS network.

### ***Single-Factor vs Two-Factor***

Single-factor authentication relies on the user knowing something that only the user would know, the most common example being a password. Two-factor adds another element, generally something that a user owns; referred to as a token.

PINsafe can be used as a single-factor or two-factor authentication system. In single-factor operation the security string can be retrieved by the user or agent on any device, therefore access to a token is not required to access the security string.

In two-factor operation the security string will only be sent to a specific device, ie the user's registered mobile phone. For example in SMS mode the security string will be sent by SMS to the mobile phone associated with that user's account. In order to successfully authenticate a user

needs to be in possession of that specific mobile phone (the second factor) and then to know the PIN to extract the one-time-code (first factor)

Two-factor is a much stronger form of authentication as it means any attacker has to successfully compromise both factors of authentication. The PINsafe implementation of two-factor authentication is particularly affective as the attacks need to be simultaneous, as the PIN is never entered as part of the authentication, it cannot be attained in isolation via a key-logging attack.

### ***Security String Rendering***

The security string sent to the user can be rendered in a number of different ways to optimize security and usability. Here are some examples.

- The PINsafe server can supply the security string as an obfuscated TURING image as a defense against OCR attacks
- The PINsafe server can supply the security string as plain text to the Swivel mobile client (Swivlet). The user can enter their PIN onto the Swivlet and then it can display the required one-time code.
- Alternatively if the mobile device has the capability, the Swivlet can “speak” the one-time code to make improve its usability for those that are vision-impaired.
- The security string can also be delivered to an agent for the agent to render the security string in any way it requires.

### ***User Groups***

The flexibility of PINsafe is further facilitated by the ability to segregate users into different groups and to associate different authentication models with those different groups. User groups can be associated with RADIUS authentication, single channel authentication and two channel authentication. Users can then be made a member of one of more of these groups to determine which authentication models are available to them. When using PINsafe with Active Directory these user groups can me matched to existing groups with Active Directory meaning that PINsafe users' rights can be managed directly from Active Directory.

## Architecture

Swivel Secure has architected PINsafe to provide flexibility within the product. The Architecture reflects the fact that PINsafe forms a key part of an overall IT solution and as such must be easy on integrate with a variety of systems.

It also reflects the flexibility of the product in terms of the different methods of managing authentication sessions.

A feature of the architecture is the use of “plug-in” classes. These pieces of software sit between the PINsafe server and the infrastructure with which PINsafe integrates. These plug-ins are flexibility points that enable interfaces to be changed without the need to change the core product.

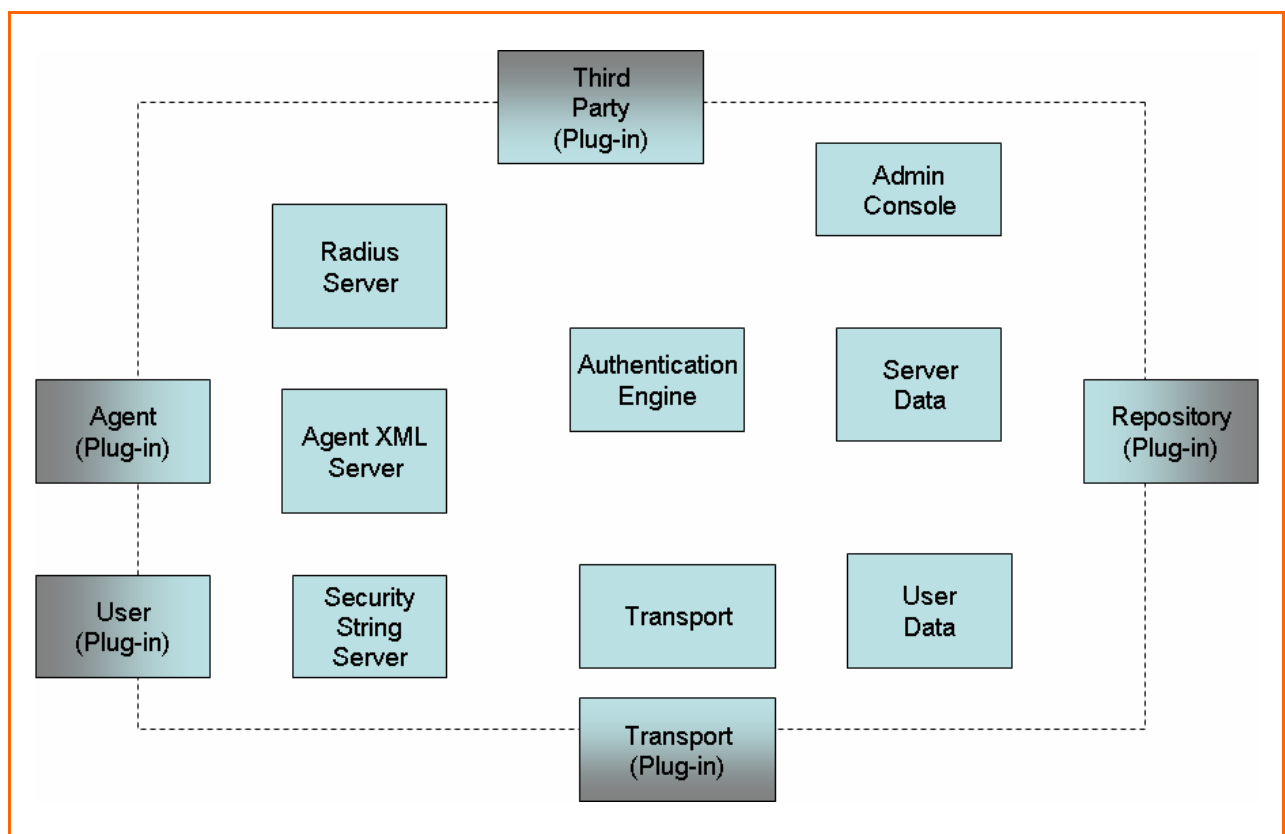


Figure 7 PINsafe Logical Architecture

## Agent (Plug-In)

Agent Plug-in is a generic term for any external piece of software that makes authentication requests to the PINsafe server. The protocol used for these requests will either be RADIUS or Agent-XML. There are many examples of this including the following:

- An SSL VPN server configured to make authentication request to PINsafe (RADIUS).
- A PINsafe IIS filter installed on an IIS webserver (Agent-XML).
- A web application making authentication requests to PINsafe (Agent-XML).

### ***RADIUS Server***

PINsafe has a RADIUS server to enable it to service RADIUS authentication requests. The RADIUS server ensures incoming requests are properly formatted, forwards the requests to the PINsafe Authentication Engine and then forms the RADIUS response and sends it to the RADIUS client (NAS).

### ***Agent-XML Server***

The Agent-XML server performs the same role for Agent clients that the RADIUS server does for RADIUS clients. The Agent-XML API is a much richer interface than the RADIUS interface and supports functions such as change-PIN. It also allows third-party authentication data to be included in the Agent-XML API call, refer to Third-Party plug-in module for more details. The interface is http based and the Agent-XML server is implemented as a servlet.

### ***Security String Server***

As stated before it is possible for users to request security strings, either directly or via an Agent. The Security String server provides these security strings. The security strings server also manages the validity of security strings. Some security strings are valid for a finite period of time, others remain valid until they are used or until a user requests new strings.

### ***Third-Party (Plug-in)***

The principle of having an open system extends to the ability of PINsafe to make authentication requests to third-party authentication systems, this is the role of the third-party module of the architecture. PINsafe can be configured to collect credential for third party authentication systems and then to submit those credentials to the third party system. This means that PINsafe can be used with other authentication systems either to add further factors of authentication or to support single-sign-on type applications. Examples include:

- Using PINsafe with a biometric, eg fingerprinting, system
- Using PINsafe with a device authentication, eg Postive ID, a PC device finger-printing product
- Using PINsafe to authenticate to a Windows™ domain

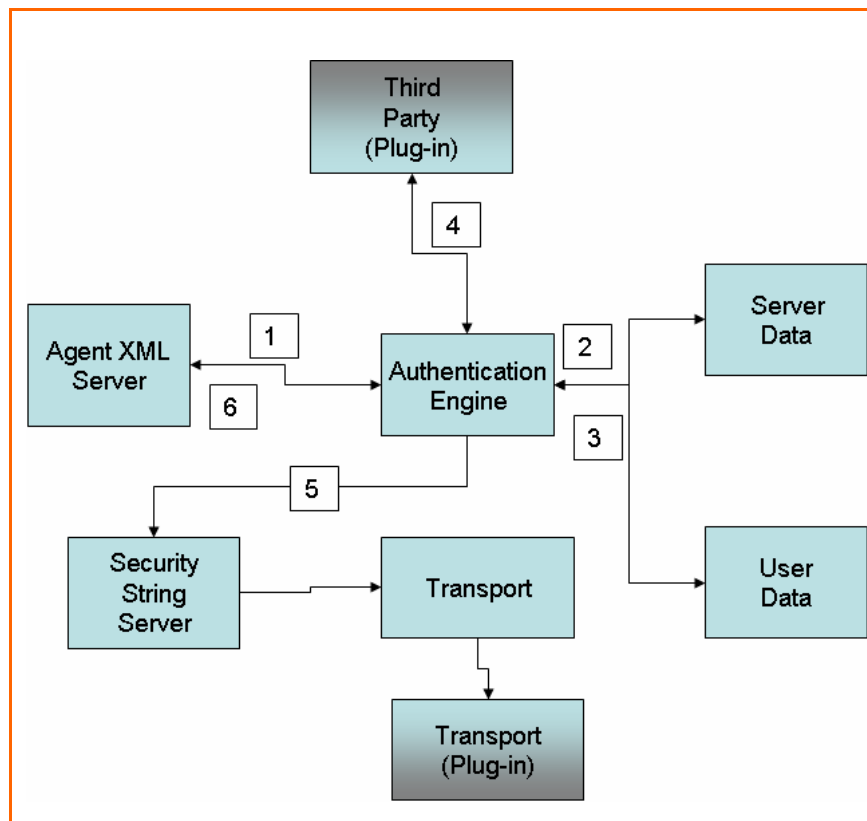


Figure 8: Authentication Engine

### ***Authentication Engine***

The Authentication Engine module sits at the center of the PINsafe server and processes the authentication request and instigates any further actions that may be required. The role of the Authentication Engine is not just to ensure that the username and one-time code match but to also ensure that the user is allowed to authenticate via the channel they have attempted, that the account is not locked, whether there is any third-party authentication data that also needs to be checked etc.

The diagram above illustrates the interactions that the Authentication Engine undertakes during an authentication

1. Agent-XML Server forwards an authentication request from an Agent
2. Authentication Engine checks server configuration to ensure that authentication requests are allowed from this agent and whether third party authentication data needs to be checked.
3. Authentication Engine then checks the user data to check that the user can authenticate, eg they are a member of the appropriate user group and their account is not locked. Then the Authentication Engine checks to see if the submitted credentials match those stored on the server.
4. If required, the Authentication Engine then forwards the third-party data to the third-party module to check those credentials.
5. In automatic mode, the Authentication Engine then prompts the sending of a new security string to the user.
6. Finally the Authentication Engine returns the result of the authentication attempt back to the agent.

### ***Transport***

The transport module of the architecture implements the means by which security string messages and alerts are sent to the user. Different users may use different transport media, so the transport module must determine which transport to use, format the message appropriately and submit it to a transport queue. The transport module must then manage that queue, dispatching messages via the appropriate Transport Plug-in, managing fails and retries and reporting errors.

### ***Admin Console***

The admin console allows the user to read and modify server configuration and user data. It is a web based admin console.

### ***Server Data***

This module manages all the server data; this is all the data relating to the PINsafe server rather than the PINsafe users. This includes all the server configuration data editable via the admin console, including the server-wide authentication policies.

### ***User Data***

This module manages the user data; this is the data specific to a user's PINsafe account. This includes their credential (username, password, PIN) as well as their user rights and associated policies that apply. When written to disk this data is encrypted using DES encryption.

### ***Repository (plug-in)***

The repository plug-in is the piece of software that manages the integration of the third-party user repository with the PINsafe user-data. This plug-in basically facilitates the mapping of the third-party user repository schema to the schema used by the User Data module. For example, which active directory group memberships correspond to which PINsafe user rights.

## Technology Overview

The PINsafe server is developed entirely in Java and uses the Apache Tomcat product as a servlet container. The Apache Tomcat Servlet Container is one of the most widely user web/application servers [source: <http://www.onjava.com/pub/a/onjava/2004/05/19/2004-survey.html>] with over 10 Million downloads [source: <http://wiki.apache.org/tomcat/PoweredBy>]

For more details visit <http://tomcat.apache.org/>

Tomcat is an open-source product which provides a number of advantages to PINsafe as a product, the main ones being:

- Well supported open source products have a large number of contributing developers improving the product.
- Well supported open source products have a large number of contributing developers identifying and rectifying defects and vulnerabilities.
- As an open-source product, it conforms to industry standards rather than proprietary ones.
- There are no licensing costs to pass onto customers.

Tomcat also provides all the advantages of running an application within a servlet container.

One further advantage of Apache Tomcat is that, just like Java, provide operating system and hardware independence. There are versions of Apache Tomcat for all the major operating systems and, as such, PINsafe can run on any of these operating systems.

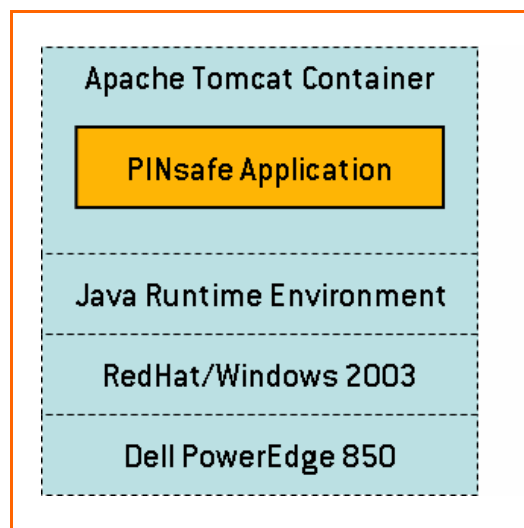


Figure 9: PINsafe appliance technology stack

### ***Appliances***

As stated above one of the advantages of building PINsafe within a servlet container is hardware and operating system independence. This means the PINsafe product can be purchased as software only for the user to install on any compatible platform of choice. This option is useful for organizations that already have an operating system and hardware vendor of choice.

For those wanting a pre-configured appliance, Swivel can supply such an appliance with a choice of operating system. Again, this choice of operating system allows make it easier for customers to

integrate PINsafe into their existing IT infrastructure, operating procedures and skill sets. The two operating systems used within PINsafe appliances are hardened versions of:-

- Red Hat Linux Enterprise Edition
- Windows 2003 Web Edition<sup>2</sup>

The appliances themselves are currently based on Dell Computers Power Edge 850 and have the following features (subject to change)

- Pentium 4 2.8 Ghz Processor with 1 MB RAM
- 2 x 73 Gbyte Hard Drives with RAID 1 controller
- 8x DVD ROM Drive

The appliances feature two hard disk drives for resilience; they are configured as a RAID1 pair. This means in the event of a disk failure the appliance will continue to operate. The faulty drive can be replaced and rebuilt to re-establish the RAID 1 pair.

### ***High Availability***

Further resilience can be provided by configuring a high-availability (HA) pair of PINSafe servers. In this configuration one PINsafe server is live and the other is acting as a hot standby. In normal operation the live server services all the authentication requests and writes a copy of any data to the standby server. If servers send out heartbeat packets to each other that indicate that it is still up and running. If the live server fails, the standby servers detects this by the lack of the heartbeat signal and the standby server takes over.

The two servers “share” an IP address so that in the event of a failover the standby server takes over the IP address meaning the failover is seamless and no other systems need to be re-configured as an event.

### **Conclusion**

The PINsafe server is architected to and successfully delivers a flexible, strong authentication system. The lack of reliance on a dedicated authentication token and eliminates the hassle and cost of managing such tokens and drastically decreases deployment time.

The flexibility of the product means that it can provide different authentication solutions to different users, even within the same PINsafe installation and it can accommodate an enterprise’s authentication requirements both now and as they change over time.

Its open design makes integration easy and a single PINsafe server can be integrated with many different IT resources.

All of these factors make PINsafe a low-risk choice of authentication solution

---

<sup>2</sup> Currently the high availability configuration is only available for Linux based appliances.