Caché Security Administration Guide

Version 2017.2
2020-06-25

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Caché Security Administration Guide
Caché Version 2017.2 2020-06-25

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

List of Figures

List of Tables

About This Book

This book describes the functionality for securing Caché applications and deployed instances.

This book covers the following fundamental topics:

- "About Caché Security" provides an overview of the available security features.
- "Authentication" explains the theory and practice of configuring Caché to authenticate its users.
- "Assets and Resources" begins the authorization chapters by describing the data and activities Caché protects — called assets — and how they are represented within Caché — as what are called "resources."
- "Privileges and Permissions" continues the authorization chapters by describing the mechanism for granting access to those resources.
- "Roles" continues the authorization chapters by describing how privileges are aggregated into the logical form called "roles" and how users can be associated with roles.
- "Users" continues the authorization chapters by describing the representation of those entities — called "users" — within Caché.
- "Services" continues the authorization chapters by describing how Caché protects the various means by which users and programs can connect to it, which are called "services."
- "Applications" concludes the authorization chapters by describing how to create and use application-based security with Caché
- "Auditing" describes how to configure Caché to log various activities.
- "Managed Key Encryption" explains how to encrypt data on disk.

This book's advanced topics are:

- SQL Security
- System Management and Security
- Using SSL/TLS with Caché
- The InterSystems Public Key Infrastructure
- Using Delegated Authentication
- Using LDAP Authentication
- Using Delegated Authorization

The book's appendices are:

- Tightening Security for a Caché Instance
- Using the cvencrypt Utility
- Frequently Asked Questions about Caché Security
- Relevant Cryptographic Standards and RFCs
- Using Character-based Security Management Routines
- About Public Key Infrastructure (PKI)

For a detailed outline, see the Table of Contents.

Other related topics in the Caché documentation set are:

- *The Caché Installation Guide*, the "Preparing for Caché Security" appendix provides preliminary setup information for an instance that will use security.
- *Using Caché Server Pages (CSP)*, the "CSP Architecture" chapter describes how to configure CSP applications, including security-related properties.
- *Using Caché SQL*, the Users, Roles, and Privileges chapter provides the SQL perspective on Caché security.

For general information, see *Using InterSystems Documentation*.

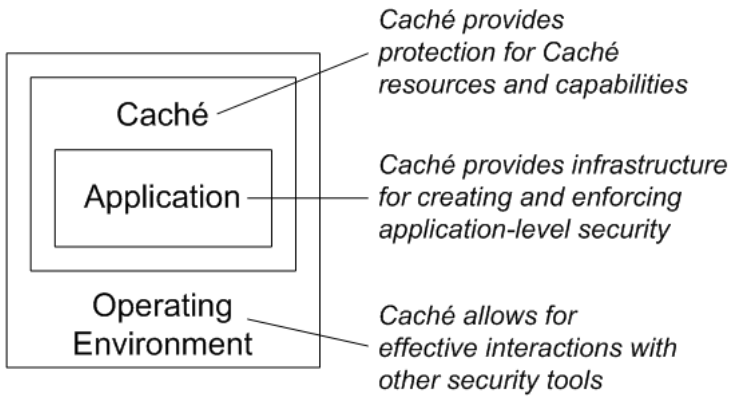
1

About Caché Security

Caché provides a simple, unified security architecture with the following features:

- It offers a strong, consistent, and high-performance security infrastructure for applications.
- It meets certification standards.
- It makes it easy for developers to build security features into applications.
- It places a minimal burden on performance and operations.
- It ensures that Caché can operate effectively as part of a secure environment and that other applications and Caché can work together well.
- It provides infrastructure for policy management and enforcement.

Figure 1–1: Caché Security and Different Levels of the Computing Environment



Caché security is based on authentication, authorization, auditing, and database encryption:

- Authentication verifies the identity of all users. It is described in the section "Authentication: Establishing Identity."
- Authorization ensures that users can access the resources that they need, and no others. It is described in the section "Authorization: Controlling User Access."
- Auditing keeps a log of predefined system and application-specific events. It is described in the section "Auditing: Knowing What Happened."
- Managed key encryption protects information against unauthorized viewing. It is described in the section "Managed Key Encryption: Protecting Data on Disk."

Caché also supports the use of SSL/TLS and provides tools for a public key infrastructure (PKI).

Note: Caché SQL security uses the Caché authentication infrastructure. The tools for Caché security authorization are described in this book, while the Caché SQL security authorization system is described in the "Users, Roles, and Privileges" chapter of the *Using Caché SQL* book.

1.1 Authentication: Establishing Identity

Authentication is how you prove to Caché that you are who you say you are. Without trustworthy authentication, authorization is moot — one user can impersonate another and then take advantage of the fraudulently-obtained privileges.

The authentication mechanisms available depend on how you are accessing Caché. Caché has a number of available authentication mechanisms:

- Kerberos — The most secure means of authentication. The Kerberos Authentication System provides mathematically proven strong authentication over a network.
- Operating-system–based — OS-based authentication uses the operating system's identity for each user to identify that user for Caché purposes.
- LDAP — With the Lightweight Directory Access Protocol (LDAP), Caché authenticates the user based on information in a central repository, known as the LDAP server.
- Caché login — With Caché login, Caché prompts the user for a password and compares a hash of the provided password against a value it has stored.
- Delegated authentication — Delegated authentication provides a means for creating customized authentication mechanisms. The application developer entirely controls the content of delegated authentication code.

You can also allow all users to connect to Caché without performing any authentication. This option is appropriate for organizations with strongly protected perimeters or in which neither the application nor its data are an attractive target for attackers.

1.1.1 About Kerberos

For maximally secure connections, Caché supports the Kerberos authentication system, which provides a highly secure and effective means of verifying user identities. Kerberos was developed at the Massachusetts Institute of Technology (MIT) to provide authentication over an unsecured network, and protects communications using it against sophisticated attacks. The most evident aspect of this protection is that a user's password is never transmitted over the network — even encrypted.

Kerberos is what is called a *trusted-third-party system*: the Kerberos server holds all sensitive authentication information (such as passwords) and is itself kept in a physically secure location.

Kerberos is also:

- Time-tested — Kerberos was originally developed in the late nineteen-eighties. Its principal architecture and design have been used for many years at many sites; subsequent revisions have addressed issues that have been discovered over the years.
- Available on all supported Caché platforms — Originally developed for UNIX®, Kerberos is available on all Caché-supported variants of UNIX®; Microsoft has integrated Kerberos into Windows 2000 and subsequent versions of Windows. (Note that because the Microsoft .NET framework does not include direct Kerberos support, Caché does not support Kerberos for the Caché Managed Provider for .NET.)
- Flexibly configurable — It accommodates heterogeneous networks.

- Scalable — The Kerberos protocol minimizes the number of interactions with its Key Distribution Center (KDC); this prevents such interactions from becoming a bottleneck on larger systems.
- Fast — As an open-source product, the Kerberos code has been scrutinized and optimized extensively over the years.

Underlying Kerberos authentication is the AES encryption algorithm. AES — the Advanced Encryption Standard — is a royalty-free, publicly-defined symmetric block cipher that supports key sizes of 128, 192, and 256 bits. It is part of the US Federal Information Processing Standard (FIPS), as chosen by United States National Institute of Standards and Technology (NIST).

For detailed content, see "Configuring for Kerberos Authentication" in the "Authentication" chapter.

1.1.2 About Operating-System–Based Authentication

Caché supports what is called *operating-system–based* (or *OS-based*) authentication. With operating system authentication, Caché uses the operating system's user identity to identify the user for Caché. When operating system authentication is enabled, the user authenticates to the operating system using according to the operating system's protocols. For example, on UNIX®, this is traditionally a login prompt where the operating system compares a hash of the password to the value stored in the /etc/passwd file. When the user first attempts to connect to Caché, Caché obtains the process' operating system level user identity. If this identity matches a Caché username, then that user is authenticated.

This capability only applies to server-side processes, such as terminal-based applications (for example, connecting through the Terminal) or batch processes started from the operating system. It is not available for an application that is connecting to Caché from another machine, such as when a copy of Studio on one machine is connecting to a Caché server on another.

This mechanism is typically used for UNIX® systems, in addition to the Windows console.

For detailed content, see "Configuring for Operating-System–Based Authentication" in the "Authentication" chapter.

1.1.3 About LDAP Authentication

Caché supports authentication through the Lightweight Directory Access Protocol (LDAP). In this case, Caché contacts an LDAP server to authenticate users, relying on its database of users and their associated information to perform authentication. The LDAP server also controls all aspects of password management, password policies, and so on.

For detailed content, see the "Using LDAP" chapter.

1.1.4 About Caché Login

Caché itself can provide a login mechanism. Specifically, Caché maintains a password value for each user account and compares that value to the one provided by the user at each login. (As with traditional OS-based authentication, Caché stores a hashed version of the password. When the user logs in, the password value entered is hashed and the two hashed versions are compared.) The system manager can configure certain password criteria, such as minimum length, to ensure a desired degree of robustness in the passwords selected by users.

For detailed content, see "Configuring for Authentication with Caché Login" in the "Authentication" chapter.

1.1.5 About Delegated Authentication

Caché supports delegated authentication, which allows you to create your own authentication mechanism. As the application developer, you fully control the content of delegated authentication code. Caché includes a routine, ZAUTHENTICATE.mac, that serves as a template for creating custom authentication code.

For detailed content, see the "Using Delegated Authentication" chapter.

1.2 Authorization: Controlling User Access

Once a user is authenticated, the next security-related question to answer is what that person is allowed to use, view, or alter. This determination and control of access is known as *authorization*. Authorization manages the relationships of users and *resources* — entities being protected. Resources are as diverse as databases, Caché services (such as for controlling web access), and user-created applications. Each user has one or more *roles*, each of which authorizes the user to perform particular activities with particular resources Caché provides tools so you can manage each resource, as well as each role's privileges in relation to each resource.

Caché also supports various role-assignment mechanisms. A *role-assignment mechanism* allows you to associate particular roles with particular authenticated users. Caché uses these associations to determine the authorized activities for the user. Each role-assignment mechanism is associated with one or more authentication mechanisms; configuring Caché includes specifying the supported combination(s) of authentication and role-assignment mechanisms.

The available role-assignment mechanisms are:

- Native authorization — Role assignment occurs within Caché. Available with the Kerberos, OS-based, and Caché login authentication mechanisms.
- Delegated authorization (**ZAUTHORIZE**) — Role assignment occurs as part of the **ZAUTHORIZE** routine. Available with the Kerberos and OS-based authentication mechanisms.
- LDAP — An LDAP (Lightweight Directory Access Protocol) server performs role assignment. Available with the OS-based and LDAP authentication mechanisms.
- **ZAUTHENTICATE** — Role assignment occurs as part of the **ZAUTHENTICATE** routine. Available with the delegated authentication mechanism, which calls **ZAUTHENTICATE**.

For a list of authentication mechanisms, role-assignment mechanisms, user types, and other security elements, see the "Authentication-Authorization Matrix" section of the "Authentication" chapter.

1.2.1 Authorization Basics

The fundamental purpose of Caché security is to establish the relationships between users and the resources that they attempt to use.

1.2.1.1 Resources, Permissions, and Privileges

The primary goal of security is the protection of *resources* — information or capabilities in one form or another. With Caché, resources can be databases, services, applications, tools, and even administrative actions. The system administrator grants access to these by assigning *permissions*. Together, a resource and an associated, assigned permission are known as a *privilege*. This is often described using the following shorthand:

```
Resource-Name:Permission
```

where *Resource-Name* is the specific resource for which permissions are being granted and *Permission* is one or more permissions being associated with the resource. For example, the granting of read and write permissions on the EmployeeInfo database is represented as:

```
%DB_EmployeeInfo:Read,Write
```

or

```
%DB_EmployeeInfo:RW
```

For most resource types, the relevant permission is Use; for databases, the permissions are Read and Write. Granting or revoking this permission enables or disables access to the resource's action(s).

For most resources, the name of a resource is <resource-type>_<specific-resource>, such as **%Admin_Operate**, **%Service_SQL**, or the **%DB_SAMPLES** database.

Differences between Resources and Assets

Resources differ from assets as follows:

- Assets are the items being protected while resources are their logical representation within the Caché security system.
- A single resource can protect multiple assets.

1.2.1.2 Users and Roles

Caché uses Role-Based Access Control (RBAC) for its authorization model. With this type of model, a user gains the ability to manipulate resources as follows:

1. *Resources* are associated with *permissions* to establish *privileges*.
2. *Privileges* are assigned to *roles*.
3. *Roles* have members, such as *users*.

A *user* connects to Caché to perform some set of tasks. A *role* describes a set of privileges that a user holds.

Roles provide an intermediary between users and privileges. Instead of creating as many sets of privileges as there are users, roles allow you to create sets of task-specific privileges. You can grant, alter, or remove the privileges held by a role; this automatically propagates to all the users associated with that role. Instead of managing a separate set of privileges for each and every user, you instead manage a far smaller number of roles.

For example, an application for a hospital might have roles for both a doctor making rounds (**RoundsDoctor**) and a doctor in the emergency room (**ERDoctor**), where each role would have the appropriate privileges. An individual user could be a member of just one of the two roles, or of both of them.

Caché comes with a set of pre-defined roles: **%Manager**, **%Operator**, **%Developer**, **%SQL**, and a role for each of the initially-installed databases; the database-related roles have names of the form **%DB_<database-name>**.

Caché also comes with a role called **%All**, which holds all privileges — that is, all permissions on all resources. There is no way to reduce this role's privileges, and at least one user must always belong to this role.

Each user has an associated *$Roles* variable, which contains the list of roles held. Once Caché authenticates a user (using one of the mechanisms described in the section "Authentication: Establishing Identity"), it grants that user all associated roles. This set of initially granted roles is known as *login roles*. A user's login roles establishes a default value for the *$Roles* variable. Once logged in, a user may temporarily be a member of additional roles — either from a Caché application or from some part of the Caché system itself; these are reflected in the value of the *$Roles* variable. The set of roles that a user has at any particular moment is called that user's *active roles*. If, at any point, a call sets the value of *$Roles* to NULL (""), then the value of *$Roles* reverts to the login roles.

When Caché adds new items to the list of roles in the *$Roles* variable, this is known as *escalating roles*. The removal of roles from the list is known as *role de-escalation*.

About Role Assignment

Role assignment depends on the role-assignment mechanism in use, which, in turn, varies by the authentication mechanism in use:

Table 1–1: Authentication Mechanisms and Role-Assignment Mechanisms

Authentication Mechanism	Role-Assignment Mechanisms
Kerberos	Caché authorization
	Delegated authorization (**ZAUTHORIZE**)
Operating System	Caché authorization
	Delegated authorization (**ZAUTHORIZE**)
	LDAP
Caché login	Caché authorization
LDAP	LDAP
Delegated Authentication (**ZAUTHENTICATE**)	**ZAUTHENTICATE**

For more information about the relationship between authentication and role assignment, see the "Authentication-Authorization Matrix" in the "Authentication" chapter.

For an instance that supports unauthenticated access, all users hold the privileges associated with the UnknownUser and _PUBLIC accounts; these accounts are described in the sections "The UnknownUser Account" and "The _PUBLIC Account," both of which are in the "Users" chapter.

Note: Regardless of how role assignment occurs, role *management* — that is, associating particular privileges with particular roles — occurs within Caché.

1.2.2 Resources and What They Protect

At the foundation of any security system is that which it protects. With Caché, resources are being protected. There are a number of kinds of resources and each governs a different key aspect of Caché:

- System resources — The ability to perform various tasks for system management, security administration, or application development. See the section "System Resources" for more information.
- Database resources — Caché databases, which can be altered or read, and which have executable code that can also be altered or run. See "Database Resources" for more information.
- Service resources — Tools for connecting to or among Caché servers. See the section "Service Resources" for more information.
- Application resources — User-defined applications, applications that come with Caché, an action in code, or a page in the Management Portal. See the section "Application Resources" for more information.

1.2.2.1 System Resources

There are several predefined administrative resources:

- **`%Admin_Operate`** — Controls a set of tasks for system operators. These include:
  - Starting and stopping (but not configuring) Caché
  - Performing backups
- **`%Admin_Secure`** — Controls a set of tasks for security administrators. These include:
  - User account management

  - Role management
  - Resource management
  - Starting and stopping services
  - Database encryption tasks

- **%Admin_Manage** — Controls a set of tasks related to managing a Caché instance. In addition to those listed for **%Admin_Operate** and **%Admin_Secure**, these include
  - Configuration management
  - Adding, modifying, and deleting databases
  - Modifying namespace mappings
  - Managing backup definition
  - Performing database restores
  - Performing journal restores

- **%Development** — Controls development tasks and tools, including:
  - Using direct mode (the programmer prompt)
  - Establishing Studio connections to a server
  - Using the global, routine, class, table, or SQL capabilities of the Management Portal.
  - Access to global, routine, class, table, or SQL capabilities programmatically
  - Caché debugging facilities

For each system resource, the Use permission enables access.

1.2.2.2 Service Resources

A service controls a user's ability to connect to Caché. For example, Telnet and ECP each have associated services. Each service can be enabled or disabled. If disabled, no one, regardless of privilege level, can use the service. If enabled, the service's specified authentication mechanism is used to authenticate the user; once authenticated, the service grants access according to the privilege level specified by the user's roles. (For services that support multiple authentication mechanisms, these are used in a pre-determined order.)

There are a large number of services available as part of Caché. These include:

- Client/server services, such as for SQL
- Console, Telnet, and the Terminal (for various kinds of terminal connections)
- A CSP connection service (for CSP and Zen web applications)
- Networking services, such as ECP

To use a service, you must hold the Use permission on the service's resource.

1.2.2.3 Database Resources

A database refers to a physical file that resides in a particular location. A database resource governs one or more Caché databases.

Note: Caché security does not directly control access to namespaces. Since a namespace can be mapped to multiple databases, there can be different security settings for its different underlying parts.

For database resources, there are two permissions available:

- Read — Allows viewing but not modification of content, and also running of routines
- Write — Allows viewing and modification of content

The Manager's Database

For each Caché instance (that is, each separately installed copy of Caché), there is a database called CACHESYS, which contains routines and globals required to administer the instance. This database is also known as the manager's database.

Because of the powerful tools available in the manager's database, it is necessary to carefully control access to it. Data and routines in it can affect the operation of Caché itself; therefore, to protect the instance as a whole, access to it should be carefully restricted.

1.2.2.4 Application Resources

Application resources can protect a number of different kinds of assets, all of which are associated with either user-defined applications or applications that come with Caché, and can include entire applications, individual actions in code, or pages in the Management Portal.

For the purposes of Caché security, an application is a software program or group of Caché routines. To protect applications, Caché supports what is called an "application definition." You can associate an application definition with an Application resource (that is, a resource of type Application); this allows you to establish a privilege that regulates its use. Any role that holds the privilege is entitled to run the application.

There are three types of application definitions:

- A *privileged routine application definition* is associated with one or more Caché routines.
- A *web application definition* is associated with a specific Caché Server Pages (CSP) or Zen application.
- A *client application definition* is associated with one or more specific executable programs, which have been created as clients for a Caché server.

For example, Caché comes with the DocBook web application for displaying documentation; if you are reading this in a browser, then you are using the DocBook application right now. For more information on applications, see the "Applications" chapter.

In addition to using Application resources to protect the application as a whole, you can also use these resources to perform authorization for a particular piece of code or with a particular Portal page. For more information on adding authorization code into an application, see the section "Checking Privileges" in the "Privileges and Permissions" chapter; for more information on authorization checks for a particular Portal page, see the section "Using Custom Resources with the Management Portal" in the "Resources" chapter.

1.2.3 For More Information on Authorization

For more information on authorization, see the following chapters in this book: "Assets and Resources," "Privileges and Permissions," "Roles," and "Users."

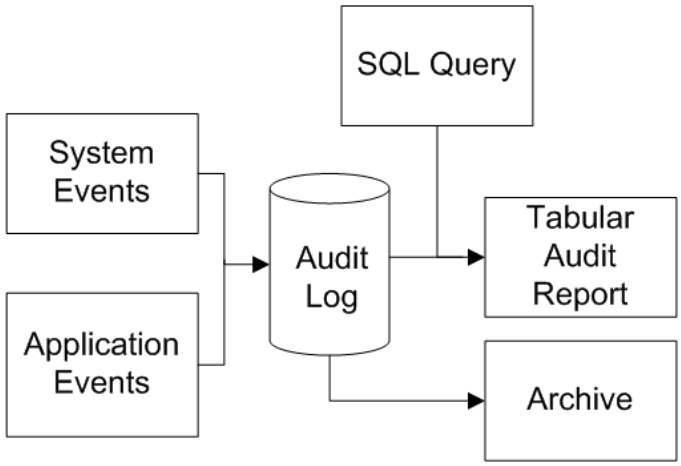
1.3 Auditing: Knowing What Happened

Auditing provides a verifiable and trustworthy trail of actions related to the system. Auditing serves multiple security functions:

- It provides proof — the proverbial "paper trail" — recording the actions of the authentication and authorization systems in Caché and its applications.
- It provides the basis for reconstructing the sequence of events after any security-related incident.
- Knowledge of its existence can serve as a deterrent for attackers (since they know they will reveal information about themselves during their attack).

The auditing facility allows you to enable logging for various system events, as well as user-defined events. Authorized users can then create reports based on this audit log, using tools that are part of Caché. Because the audit log can contain sensitive information, running an audit report itself generates an entry for the audit log. The included Caché tools support archiving the audit log and other tasks.

Figure 1–2: Caché Auditing System



For more information on auditing, see the "Auditing" chapter.

1.4 Managed Key Encryption: Protecting Data on Disk

The purpose of authentication is to ensure that Caché users are who they say they are. The purpose of authorization is to control access to data through Caché. The purpose of auditing is to keep a record of what has happened during interactions with Caché and its data. In addition to this, there is the need to prevent unauthorized access to data on disk. To protect against such access, Caché provides managed key encryption, a suite of technologies that protects data at rest. This suite includes block-level database encryption, data element encryption for applications, and encryption key management.

The tools protect data at rest — that is, they secure information stored on disk — by preventing unauthorized users from viewing this information. Caché implements encryption using the AES (Advanced Encryption Standard) algorithm. Database encryption and decryption occur when Caché writes to or reads from disk, and the information handled includes the data itself, indices, bitmaps, pointers, allocation maps, and incremental backup maps. Data element encryption is supported by a set of methods that allow an application to encrypt and decrypt content as desired. And, underlying the encryption tools are key management tools that allow for simple creation and management of data encryption keys and the key files that contain them.

Those experienced with encryption systems for databases may have concerns about encryption having dire effects on performance, but, with Caché, these concerns are unfounded. Encryption and decryption have been optimized, and their effects are both deterministic and small for any Caché platform; in fact, there is no added time at all for writing to the database.

For more information on these tools, see the chapter "Managed Key Encryption."

1.5 Managing Security with the Management Portal

To manage security for a Caché instance, use the Management Portal. From the Management Portal home page, the **System Administration** menu includes submenus for **Security** and **Encryption**. The **Security** submenu contains choices for managing the Caché instance as a whole, including users, roles, services, resources, auditing, and the security properties of any applications defined for the Caché instance. The **Encryption** submenu contains choices related to the technologies of managed key encryption: database encryption, data element encryption, and encryption key management.

1.6 Notes on Technology, Policy, and Action

Caché can play a significant role in providing security. However, it constitutes only part of a computing environment. To properly and fully secure that environment, Caché must be part of a solution that employs other security products and tools (such as firewalls and the security features of operating systems). This is why the security features in Caché are designed to successfully interoperate with those of other products.

Also, although Caché can do much to prevent attacks and misuse of data, it cannot do the entire job. If a user goes to lunch with a Terminal window open, your organization's data is vulnerable to attack.

And while technology can solve many security problems, it cannot teach users to behave responsibly. An organization must define clear policies that specify what can, cannot, and must be done. Further, it must educate its members in how to follow these policies and why. Without such an action, all the security in Caché and other products will do no good; with such an action, Caché can be part of a secure and productive environment.

1.7 A Note on Certification

Security certifications are a frequent requirement for government purchases, and are increasingly requested for private sector purchases. Effective February 15, 2007, Caché received certification according to the Common Criteria standard (EAL 3).

Common Criteria standard provides a set of common security standards for a wide and growing number of nations around the globe. It provides an evaluation assurance scale with levels from 1 to 4 (EAL), where a product's rating indicates the rigor of evaluation to which it has been subjected; commercially available products are rated from one (least rigorous) to four (most rigorous). Caché is certified at EAL 3. Such a level indicates that Caché can effectively serve as part of a highly secure operational environment.

2

Authentication

2.1 Authentication Basics

Authentication verifies the identity of any user attempting to connect to Caché. Once authenticated, a user has established communications with Caché, so that its data and tools are available. There are a number of different ways that a user can be authenticated; each is known as an *authentication mechanism*. Caché is typically configured to use only one of them. The supported authentication mechanisms are:

- Kerberos
- Operating-System–Based
- Caché Login
- LDAP Authentication
- Delegated Authentication

Caché supports authentication using user-defined code, which is known as delegated authentication. It also supports authentication using LDAP, the Lightweight Directory Access Protocol. Finally, for those sites that prefer no authentication at all, Caché supports unauthenticated access.

The authentication mechanism is used by what are called *connection tools*. These specify the means by which users establish their connection with Caché. Each connection tool (such as the Terminal, Java, or CSP) uses a Caché service that allows the administrator to specify the supported authentication mechanism(s). (A Caché service is a gatekeeper for connecting to Caché; for more information on services, see the chapter "Services.")

There are three categories of connection tools, each of which is known as an *access mode*. Each access mode has its own characteristics and has its own supported services. The access modes are:

- Local — The user interacts directly with the Caché executable on the machine where that executable is running.
- Client/Server — The user is operating a separate executable that connects to Caché.
- Web — The user has a Web browser and is interacting with Caché through a Web-based application.

An end-user uses a connection tool to interact with Caché in a particular access mode using a particular authentication mechanism. Remember that the processes described in this chapter do not themselves establish authenticated access. Rather, they establish the infrastructure that an application uses when authenticating users via a particular mechanism in a particular access mode.

It is recommended that each instance of Caché use only one authentication mechanism and that you choose the instance's authentication mechanism prior to installing Caché. Once installation has occurred, you can then begin configuring Caché to use the selected mechanism. This involves several steps:

- With Kerberos, ensure that all Caché users are listed in the Kerberos KDC (Key Distribution Center) or Windows Domain Controller.
- With operating-system–based authentication, ensure that all Caché users appear in the operating system list.
- For all authentication mechanisms, configure all supported services to use only the selected authentication mechanism.
- For all authentication mechanisms, disable all unsupported services.
- For all authentication mechanisms, configure all applications to use only the selected authentication mechanism.

Note: Regardless of the selected authentication mechanism, during start-up and shut-down, operating system authentication is always used.

Here's how to use this chapter:

1. If you have already chosen an authentication mechanism, read about it; if you have not chosen an authentication mechanism, read about them all and choose one. The relevant section for this is "About the Different Authentication Mechanisms."
2. Read about those access modes that are relevant for your situation in the section "About the Different Access Modes."
3. Configure your environment according to the instructions in "Configuring for Kerberos Authentication," "Configuring for Operating-System–Based Authentication," or "Configuring for Authentication with Caché Login." To use an external mechanism for authentication, Caché includes support for LDAP authentication and delegated (user-defined) authentication.
4. For all authentication mechanisms, Caché supports two-factor authentication. If you want to implement two-factor authentication, configure the instance according to the instructions in the section "Configuring Two-factor Authentication."

2.2 About the Different Authentication Mechanisms

Caché has several ways in which it can authenticate a user, that is, verify the identity of a user. These are:

- Kerberos Authentication
- Operating-System–Based Authentication
- Caché Authentication

- LDAP Authentication
- Delegated Authentication

A site may also be configured for unauthenticated access.

2.2.1 Kerberos Authentication

Where strong authentication is required, Caché can use the Kerberos protocol to enable users and Caché itself to identify each other and to ensure the validity of communications within a session. For a brief overview of Kerberos, see the "About Kerberos" section in the "Introduction" chapter of this book; for more detailed information, see the MIT Kerberos Web site and its list of available documentation.

In the Kerberos model, there are several different actors. All the different programs and people being authenticated by Kerberos are known as *principals*. The Kerberos system is administered by a Kerberos Key Distribution Center (KDC); on Windows, the Windows Domain Controller performs the tasks of a KDC. The KDC issues tickets to users so that they can interact with programs, which are themselves represented by *service principals*. Once a user has authenticated and has a service ticket, it can then use a program.

Specifically, Kerberos authentication involves three separate transactions:

1. The client receives what is called a "ticket-granting ticket" ("TGT") and an encrypted session key.
2. The client uses the TGT and session key to obtain both a service ticket for Caché as well as another encrypted session key.
3. The client uses the service ticket and second session key to authenticate to Caché and optionally establish a protected connection.

Aside from a possible initial password prompt, this is designed to be invisible to the user.

In order for strong authentication to be meaningful, all Caché services that support it must have Kerberos enabled and those that don't support it must be disabled. The exception to this is that services intended to operate within the Caché security perimeter, such as ECP, do not support Kerberos; you can simply enable or disable these services, since they are designed for use in an externally secured environment.

2.2.2 Operating-System–Based Authentication

With operating-system–based authentication, Caché uses the operating system's user identity to identify the user for Caché purposes. Specifically, the process is:

1. Caché obtains the process' operating-system user identity.
2. Caché checks if the operating system identity matches a Caché username. If so, then the user is automatically authenticated for Caché as well.

2.2.3 Caché Authentication

Caché has its own algorithms for providing password-based authentication. This mechanism is listed in the Management Portal as "Password" authentication. Since Kerberos and OS-based authentication both typically use passwords, this documentation refers to the native Caché mechanism as *Caché login*.

For password authentication, Caché maintains a password value for each user account and compares that value to the one provided by the user at each log in. (In actuality, Caché does not store the password value itself but a hashed version of it. When the user logs in, that entered password value is hashed and the two hashed versions are compared.) The system manager can establish certain password criteria, such as minimum length, to ensure a certain degree of robustness in the

passwords selected by users; the criteria are described in the section "Password Strength and Password Policies" in the chapter "System Management and Security."

Caché stores only irreversible cryptographic hashes of passwords. The hashes are calculated using the PBKDF2 algorithm with the HMAC-SHA-1 pseudorandom function, as defined in Public Key Cryptography Standard #5 v2.1: "Password-Based Cryptography Standard." The current implementation uses 1024 iterations, 64 bits of salt, and generates 20 byte hash values. There are no known techniques for recovering original passwords from these hash values.

2.2.4 LDAP Authentication

Caché supports authentication based on LDAP (the Lightweight Directory Access Protocol). With LDAP authentication, Caché retrieves user information from a central LDAP repository. Because an environment may already support LDAP authentication, such as with Windows Active Directory, an instance of Caché may be able to use LDAP for its authentication and simply fit into this larger infrastructure. For more details about LDAP authentication, see the chapter "Using LDAP Authentication."

2.2.5 Delegated Authentication

Caché supports the use of custom authentication mechanisms through what is known as "delegated authentication." Delegated authentication occurs if an instance of Caché has a ZAUTHENTICATE routine in its %SYS namespace. If such a routine exists, Caché uses it to authenticate users, either with calls to new or existing code. For more details about delegated authentication, see the chapter "Using Delegated Authentication."

2.2.6 Unauthenticated Access

You can configure any Caché service to operate without any authentication mechanism. This is known as *unauthenticated access*. Generally, if you configure Caché services to allow unauthenticated access, it is recommended there be unauthenticated access exclusively. If there is support for an authentication mechanism and then unauthenticated access if authentication fails, this is what is called *cascading authentication*, which is described in the section "Cascading Authentication"; the circumstances for using more than one authentication mechanism are described in the section "Using Multiple Authentication Mechanisms."

2.3 About the Different Access Modes

Caché supports three access modes:

- Local
- Client/Server
- Web

2.3.1 About Local Access

With local access, the end-user is on the same machine as the Caché server. To gain access to the data, the user runs a private image of Caché that is reading from and writing to shared memory. If there are multiple local users, each has an individual copy of the Caché executable and all the executables point to the same shared memory. Because the user and the executable are on the same machine, there is no need to protect or encrypt communications between the two, since nothing is being passed from one executable to another. Because communications between the user and Caché go on within a single process, this is also known as *in-process authentication*.

Local access is available for:

- The terminal — The Terminal uses the **%Service_Console** service on Windows and the **%Service_Terminal** service on other operating systems.
- Callin — Callin uses the **%Service_CallIn** service.

2.3.2 About Client/Server Access

With client/server access, the Caché executable is the server and there is a client executable that can reside on a separate machine. Caché accepts a connection, possibly over a wire, from the client. This connection can use any language or protocol that Caché supports. These include:

- ActiveX — Uses **%Service_Bindings**
- C++ — Uses **%Service_Bindings**
- Caché Direct — Uses **%Service_CacheDirect**
- ComPort — Uses **%Service_ComPort**
- Java — Uses **%Service_Bindings**
- JDBC — Uses **%Service_Bindings**
- ODBC — Uses **%Service_Bindings**
- Perl — Uses **%Service_Bindings**
- Python — Uses **%Service_Bindings**
- Telnet — Uses **%Service_Telnet**

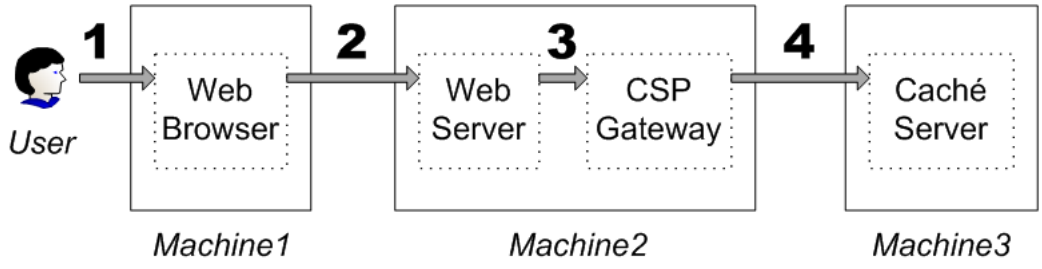
All connection tools support authentication through Kerberos or Caché login except **%Service_ComPort**, which only supports authentication through Caché login.

In each case, the server specifies the supported authentication type(s). When the client initiates contact with the server, it must attempt to use one of these supported types; otherwise, the connection attempt is rejected. Not all authentication types are available for all connection tools.

2.3.3 About Web Access

The web access mode supports connections of the following form:

Figure 2–1: Architecture of a Web Connection



1. A user requests content or an action in a Web browser.
2. The Web browser passes along the request to the Web server.
3. The Web server is co-located with the CSP Gateway and passes the request to the Gateway.

4. The Gateway passes the request to the Caché server.

When the Caché server provides content for or performs an action relating to the user, the entire process happens in the other direction.

For the user to authenticate to Caché, a username and password must be passed down the line. Hence, this access mode is also known as a proxy mode or proxy connection. Once the information reaches the Caché machine, the arrangement between user and server is similar to that in the local access mode. In fact, the web access mode also uses in-process authentication.

2.4 Configuring for Kerberos Authentication

To configure a Caché instance for Kerberos authentication, the process is:

1. Ensure that Caché is set up to run as a Kerberos service.

   The procedure varies, depending on the operating system of the Caché server and the type of environment; see the "Preparing the Security Environment for Kerberos" section of the "Preparing for Caché Security" appendix of the *Caché Installation Guide* for more information.

2. Enable the relevant Kerberos mechanisms on the **Authentication/CSP Session Options** page (**System Administration** > **Security** > **System Security** > **Authentication/CSP Session Options**).

3. Determine which services will be used to connect to Caché and disable all other services. For a list of which services are used by what connection tools, see the table "Connection Tools, Their Access Modes, and Their Services."

4. For client/server connections, specify what Kerberos connection security level the server requires. This is how you determine which Kerberos features are to be part of connections that use the service. See the section "Specifying Connection Security Levels" for more information.

5. For client/server connections, perform client-side setup. This ensures that the application has access to the information it needs at runtime. This information includes:

   - The name of the service principal representing Caché.
   - The allowed connection security levels.

   Setting up this information may involve configuring a Windows preferred server or some other configuration mechanism. See the section "Setting Up a Client" for more information.

6. Specify how the authentication process obtains user credentials. This is either by checking the user's Kerberos credentials cache or by providing a Kerberos password prompt for the user. See the section "Obtaining User Credentials" for more information.

7. To maximally secure web connections, set up secure channels for the following connections:

   - Web browser to Web server
   - CSP Gateway to Caché server

Important: On Windows, when logged in using a domain account, OS-based and Kerberos authentication are the same. When logged on locally, Kerberos is subject to a KDC spoofing attack and is therefore neither secure nor recommended.

2.4.1 About Kerberos and the Access Modes

Each connection tool uses a service to establish communications with Caché. It also uses a particular access mode. To ensure maximum protection, determine which services you need, based on which connection tools you are using. If you are not using a service, disable it.

The following is a list of connection tools, their access modes, and their services:

Table 2–1: Connection Tools, Their Access Modes, and Their Services

Connection Tool	Access Mode	Service
ActiveX	Client/Server	**%Service_Bindings**
C++	Client/Server	**%Service_Bindings**
Caché Telnet	Client/Server	**%Service_Telnet**
CallIn	Local	**%Service_CallIn**
Console	Local	**%Service_Console**
CSP	Web	**%Service_CSP**
Java	Client/Server	**%Service_Bindings**
JDBC	Client/Server	**%Service_Bindings**
ODBC	Client/Server	**%Service_Bindings**
Terminal	Local	**%Service_Terminal**
Zen	Web	**%Service_CSP**

2.4.1.1 Local

Kerberos authentication for a local service establishes that the user and Caché are both valid Kerberos principals. There is only one machine in use and only one process on that machine; hence, the configuration pages for these services in the Portal allow you to specify whether to use Kerberos prompting (labeled simply as Kerberos in the Management Portal) or Kerberos credentials cache.

In this scenario, there is no *connection* between the user and Caché, since both are using the same process on the same machine. Because the two are sharing a process, there is no information being passed through an insecure medium and therefore no need to offer special protections for this data. (This situation is known as *in-process authentication*.)

2.4.1.2 Client/Server

Client/server applications include connections from ActiveX, C++, Java, JDBC, ODBC, Perl, Python, and through Caché Direct and Telnet. For a client/server application using Kerberos authentication, the user needs credentials to interact with Caché via the application.

The server and client each require configuration. Server configuration specifies which type of connections are accepted; client configuration specifies what type of connection is attempted and may also specify how to obtain the user's credentials.

With client/server connections, Kerberos supports various connection security levels, which are configured on the Caché server machine:

- Kerberos — Kerberos manages the initial authentication between the user and Caché. Subsequent communications are not protected.

- Kerberos with Packet Integrity — Kerberos manages the initial authentication between the user and Caché; each subsequent message has a hash that provides source and content validation. This provides verification that each message in each direction is actually from its purported sender; it also provides verification that the message has not been altered in transit from sender to receiver.
- Kerberos with Encryption — Kerberos manages initial authentication, ensures the integrity of all communications, and also encrypts all communications. This involves end-to-end encryption for all messages in each direction between the user and Caché.

2.4.1.3 Web

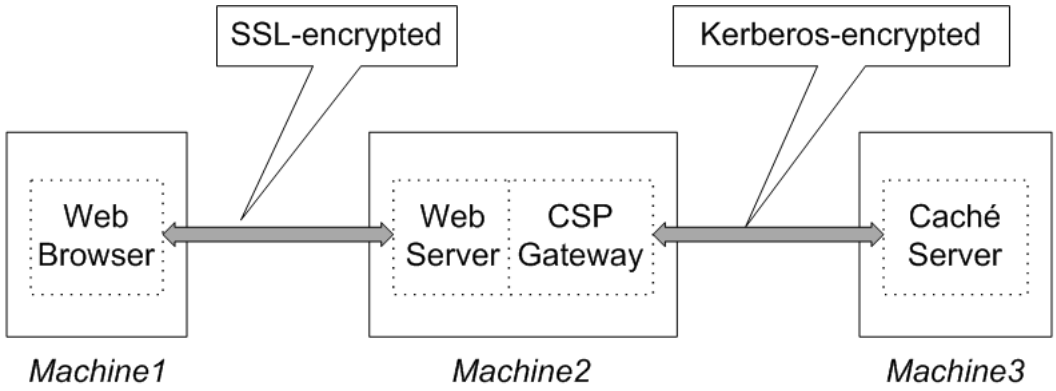
When running a web application (using either CSP or Zen), the user does not interact directly with the Caché server. To protect all information from monitoring, you need to encrypt the connections between the user and Caché as follows:

- Configure the Web server so that it uses SSL to secure browser connections to it.
- Co-locate the Web server and the CSP Gateway, so there is no need to secure the connection between them.
- Configure the CSP Gateway to use Kerberos authentication and encryption. Use the Gateway's Kerberos principal to establish such a connection.

This applies to both CSP and Zen, since Zen uses CSP for its underlying connection.

The architecture is:

Figure 2–2: Architecture of a Kerberos-Protected Web Connection



Any communications between the end-user and Caché occurs through SSL-encrypted or Kerberos-encrypted pipes. For Kerberos-secured connections, this includes the end-user's Kerberos authentication.

Because the Caché server cannot prompt the end-user for a password, it invokes an API that sends HTML content to the browser to prompt. The user completes this form that has been sent; it travels back to the Web server, which hands it to the CSP Gateway, which then hands it to the CSP server (which is part of Caché itself). The CSP server acts as a proxy on behalf of the user at the browser; this is why this kind of a connection is known as a *proxy* connection. At the same time, all information related to the user resides on the server machine (as with the local access mode); hence a web connection is also a form of in-process authentication.

2.4.2 Specifying Connection Security Levels

Client/server connections to Caché use one of the following services:

- **`%Service_Bindings`** — ActiveX, C++, Java, JDBC, ODBC, Perl, Python
- **`%Service_CacheDirect`** — Caché Direct

- **%Service_Telnet** — Telnet

For any Kerberos connection using one of these services, you must specify the connection security levels which the server accepts. To configure the service's supported connection security levels, the procedure is:

1. On the **Authentication/CSP Session Options** page (**System Administration** > **Security** > **System Security** > **Authentication/CSP Session Options**), specify which connection security levels to enable for the entire Caché instance, where these can be:

   - **Kerberos** — Initial authentication only
   - **Kerberos with Packet Integrity** — Initial authentication and packet integrity
   - **Kerberos with Encryption** — Initial authentication, packet integrity, and encrypting all messages

   For more information on the **Authentication Options** page, see the section "Authentication Options" in the chapter "System Management and Security."
2. On the **Services** page (**System Administration** > **Security** > **Services**), click the service name (in the **Name** column); this displays the **Edit Service** page for the service.
3. On the **Edit Service** page, specify which connection security levels to require as part of a Kerberos connection. After making this selection, click **Save**.

If a client attempts to connect to the server using a lower level of security than that which is specified for the server, then the connection is not accepted. If a client attempts to connect to the server using a higher level of security than that which is specified for the server, then the server connection attempts to perform authentication using the level of security that it specified.

2.4.3 Setting Up a Client

When using the client/server access mode, you need to configure the client. The particulars of this process depend on the connection technology being used.

2.4.3.1 Telnet and Caché Direct: Setting Up the Preferred Server for Use with Kerberos

With a Windows client, when establishing a connection using Caché Direct or Caché telnet for Windows, the client uses configuration information that has been stored as part of a remote server.

Important: Caché has its own telnet server for Windows. When connecting to a non-Windows machine, there is no Caché telnet server available — you simply use the telnet server that comes with the operating system. Once you have established the connection to the server machine, you can then start Caché using the **%Service_Terminal** service.

To configure a client connection coming in through telnet or Caché Direct, go to the client machine. On that machine, the procedure is:

1. Click on the Caché cube and select **Preferred Server** from the menu (the **Preferred Server** choice also displays the name of the current preferred server).
2. From the submenu that appears, choose **Add/Edit**.
3. To create a new remote server, click the **Add** button; to configure an already-existing server, choose the Caché server to which you are connecting and click the **Edit** button.
4. This displays the **Add Connection** dialog. In the **Authentication Method** area on that dialog, click **Kerberos**. This expands the dialog to display a number of additional fields.

5. If you are editing the values for an already-existing server, there should be no need to change or add values for the more general fields in this dialog, as they are determined by the server that you chose to edit.

   If you are adding a new server, the fields to complete are described in the section "Define a Remote Server Connection" of the "Connecting to Remote Servers" chapter of the *Caché System Administration Guide*.

6. In the dialog's Kerberos-related fields, specify values for the following fields:

   - The connection security level, where the choices are Kerberos authentication only; Kerberos authentication with packet integrity; or Kerberos authentication, packet integrity, and encryption
   - The service principal name. For information on setting up service principal names, see the section "Names and Naming Conventions" in the appendix "Preparing for Caché Security" of the *Caché Installation Guide*.
   - If you are configuring a telnet connection to a Windows machine, check the box specifying that the connection use the Windows Caché Telnet server.

7. Click **OK** to save the specified values and dismiss the dialog.

2.4.3.2 Setting Up an ODBC DSN for Use with Kerberos

Caché supports Kerberized ODBC connections from clients on Windows, UNIX®, and Mac to DSNs (Data Source Nodes) on all platforms. The ways of configuring client behavior vary by platform:

- On all platforms, the **SQLDriverConnect** function is available, which accepts a set of name-value pairs. **SQLDriverConnect** is a C call that is part of the ODBC API and is documented at the Microsoft Web site. Its name-value pairs are the same as those for the initialization file available on non-Windows platforms.
- On non-Windows platforms, use the Caché ODBC initialization file to specify name-value pairs that provide connection information. This file is described generally in *Using Caché ODBC*. The file has the following Kerberos-related variables:
  - *Authentication Method* — Specifies how the ODBC client authenticates to the DSN. 0 specifies Caché login; 1 specifies Kerberos.
  - *Security Level* — For Kerberos connections, specifies which functionality is used to protect the connection. 1 specifies that Kerberos is used for authentication only; 2 specifies that Kerberos is used for authentication and to ensure the integrity of all packets passed between client and server; and 3 specifies that Kerberos is used for authentication, packet integrity, and to encrypt all messages.
  - *Service Principal Name* — Specifies the name of Caché service that is serving as the DSN. For example, the service principal might have "cache/localhost.domain.com" as its name.

  The names of these variables must have spaces between the words. They are not case-sensitive.
- On a Windows client, you can specify connection information through a GUI: the ODBC DSN configuration dialog. Caché provides options on the **System DSN** tab. This screen has associated help that describes its fields. The path on the Windows Start menu to display this screen varies by version of Windows; it may be listed under **Administrative Tools**.

  Important: On 64-bit Windows, there are two versions of odbcad32.exe: one is located in the C:\Windows\System32\ directory and the other is located in the C:\Windows\SysWOW64\ directory. If you are running 64-bit Windows, configure DSNs through the one in C:\Windows\SysWOW64\.

2.4.3.3 Setting Up a Java or JDBC Client for Use with Kerberos

Caché provides a Java class that serves as a utility to assist with Java client configuration. Run it when you are ready to configure the client. The procedure is:

1. Make sure that the path to cachejdbc.jar and cachedb.jar is available to the *CLASSPATH* variable on the client. By default, this file is in the <cache-install-dir>/dev/java/lib/JDK16 directory. (For general information on setting up Java clients, see the "Installation and Configuration" section of "The Caché Java Binding" chapter of *Using Java with Caché*; for a description of the jar files, see "The Caché Java Class Packages" in the same chapter.)
2. To configure the client, issue the Java **Configure** command:

   ```
   > java com.intersys.jgss.Configure
   ```

   Note: This command is case-sensitive.

   This program uses Java Generic Security Services (JGSS) to perform the following actions:

   - If necessary, modifies the java.security file.
   - Creates or modifies the isclogin.conf file.

     Note: The parameters to the login module that appear in the isclogin.conf file depend on whether the server is using the Sun Java implementation or the IBM Java implementation. IBM AIX® and SUSE Linux use the IBM implementation; all other supported Caché platforms use the Sun implementation.

3. The program then prompts you to create and configure the krb5.conf file. If the file exists, the command prompts if you wish to use the existing krb5.conf or replace it; if you choose to replace it, it prompts for the following information:
   a. Kerberos realm — It offers the local domain in lowercase as a default value for the domain.
   b. Primary KDC — You only need include the local machine name, as the program appends the Kerberos realm name to the machine name for you.
   c. Secondary KDC(s) — You can specify the names of zero or more KDCs to replicate the content of the primary KDC.
4. After receiving this information, run the command a second time. (It instructs you to do this.)
5. When prompted to replace krb5.conf, choose to leave the existing file. The command then tests the connection by prompting for the username and password of a principal in the specified Kerberos realm.

If this succeeds, then client configuration is complete.

2.4.3.4 Setting Up a Client on C++, Perl, Python, and ActiveX for Use with Kerberos

To be able to establish a Kerberized connections through these bindings, you need only configure the Caché server to accept Kerberos connections. Once the server is properly configured, the application then can establish the connection by using the appropriate calls. For information on the appropriate language-specific calls, see the Caché Language Bindings books.

2.4.4 Obtaining User Credentials

For all access modes, you need to specify whether the application obtains the user's credentials from an existing credentials cache or by prompting for a username and password.

2.4.4.1 Obtaining Credentials for Local Access Mode

For the local access mode, the user's credentials reside on the same machine as Caché. In this situation, the application is using a service to connect to Caché. This includes the following services:

- **%Service_CallIn**
- **%Service_Console**

- **%Service_Terminal**

To specify how to get credentials, the procedure is:

1. On the **Services** page (**System Administration** > **Security** > **Services**) and select the service from the **Name** column. This displays the **Edit Service** page for the service.
2. On the **Edit Service** page, specify how to get credentials. Either select prompting (the **Kerberos** check box) or by using a credentials cache (the **Kerberos Credentials Cache** check box). Do not mark both.

   Click **Save** to use the settings.

Note: If you enable both Kerberos (prompting) and Kerberos credentials cache authentication for the service, then the credentials cache authentication takes precedence. This is behavior specified by Kerberos, not Caché.

On Windows with a Domain Controller (the likely configuration for Windows), logging in establishes a Kerberos credentials cache. On UNIX®, Linux, and MacOS, the typical default condition is to have no Kerberos credentials, so that Caché is then configured to use Kerberos prompting; on these systems, the user can obtain credentials in either of the following ways:

- Running **kinit** before invoking the Terminal
- Logging in to a system where the login process performs Kerberos authentication for the user

In these situations, Caché can be configured to use the credentials cache.

2.4.4.2 Obtaining Credentials for Client/Server Access Mode

For client/server access mode, the user's credentials reside on the machine that hosts the client application. In this case, the manner in which you specify how to obtain credentials varies according to how the client is connecting:

- ActiveX, Caché Direct, C++, ODBC, Perl, Python, and Telnet
- Java and JDBC

ActiveX, Caché Direct, C++, ODBC, Perl, Python, and Telnet

The underlying Caché code used by these connection tools assumes that end-users already have their credentials; no prompting is necessary.

On Windows, every user logged on in the domain has a credentials cache.

On other operating systems, a user has a credentials cache if the operating system has performed Kerberos authentication for the user, or if the user has explicitly run **kinit**. Otherwise, the user has no credentials in the cache and the connection tool fails authentication.

Note: Not all connection tools are available on all operating systems.

Java and JDBC

When using Java and JDBC, there are two different implementations of Java available — either Sun or IBM. These have several common behaviors and several differing behaviors.

Both implementations store information about a connection in properties of an instance of the java.util.Properties class. These properties are:

- *user* — The name of the user who is connecting to the Caché server. This value is only set for certain connection behaviors.
- *password* — That user's password. This value is only set for certain connection behaviors.

- *service principal name* — The Kerberos principal name for the Caché server. This value is set for all connection behaviors.
- *connection security level* — The type of protection that Kerberos provides for this connection. 1 specifies that Kerberos is used for authentication only; 2 specifies that Kerberos is used for authentication and to ensure the integrity of all packets passed between client and server; and 3 specifies that Kerberos is used for authentication, packet integrity, and to encrypt all messages. This value is set for all connection behaviors.

In the following discussions, the instance of the java.util.Properties class is referred to as the *connection_properties* object, where the value of each of its properties is set with a call to the **connection_properties.put** method, such as

```
String principalName = "MyCacheServer";
connection_properties.put("service principal name",principalName);
```

For both implementations, credentials-related behavior is determined by the value of a parameter in the isclogin.conf file (see "Setting Up a Java or JDBC Client for Use with Kerberos" for more information on this file).

There are two differences between the behavior of the two Java implementations:

- To specify credentials-related behavior, the parameter name to set in the isclogin.conf file differs for each implementation:
  - For IBM, it is *useDefaultCcache*.
  - For Sun, it is *useTicketCache*.
- There are different behaviors available on each implementation. These are described in the following sections.

Specifying Behavior on a Client Using the IBM Implementation

The options are:

- To use a credentials cache, set the value of the *useDefaultCcache* parameter to TRUE and do not set the values of the *user* or *password* properties. Note that if no credentials cache is available, then an exception is thrown.
- To use a username and password that are passed in programmatically, set the value of the *useDefaultCcache* parameter to FALSE and set the values of the *user* and *password* properties.
- To prompt for a username and password, set the value of the *useDefaultCcache* parameter to FALSE and do not set the values of the *user* or *password* properties. Because these properties do not have values set, classes from libraries supplied with Caché can be used to generate prompts for them.

Specifying Behavior on a Client Using the Sun Implementation

The options are:

- To exclusively use a username and password that are passed in programmatically, set the value of the *useTicketCache* parameter to FALSE and set the values of the *user* and *password* properties.
- To exclusively prompt for a username and password, set the value of the *useTicketCache* parameter to FALSE and do not set the values of the *user* or *password* properties. Because these properties do not have values set, classes from libraries supplied with Caché can be used to generate prompts for them.
- To exclusively use a credentials cache, set the value of the *useTicketCache* parameter to TRUE. To prevent any further action, set the values of the *user* and *password* properties to bogus values; this prevents prompting from occurring and ensures the failure of any authentication attempt based on the properties' values.
- To attempt to use a credentials cache and then fall through to using a username and password that are passed in programmatically, set the value of the *useTicketCache* parameter to TRUE and set the values of the *user* and *password* properties. If there is no credentials cache, then the properties' values are used.

- To attempt to use a credentials cache and then fall through to prompting for a username and password, set the value of the *useTicketCache* parameter to TRUE and do not set the values of the *user* or *password* properties. If there is no credentials cache, then classes from libraries supplied with Caché can be used to generate prompts for them.

2.4.4.3 Obtaining Credentials for Web Access Mode

With a Web-based connection that uses Kerberos, there is always a username and password prompt. If these result in authentication, the user's credentials are placed in memory and then discarded when no longer needed.

2.4.5 Setting Up a Secure Channel for a Web Connection

To maximally secure a web connection, it is recommended that the two legs of communication — both between the browser and the Web server and then between the CSP Gateway and Caché — use secure channels. This ensures that any information, such as Kerberos usernames and passwords, be protected in transmission from one point to another. To secure each communications channel, the procedure is:

- Between the Web browser and Web server
- Between the CSP Gateway and Caché

2.4.5.1 Securing the Connection between a Web Browser and Web Server

The typical means of securing a connection between a Web browser and a Web server is to use SSL (Secure Sockets Layer) or TLS (Transport Layer Security), its successor. While Caché does not provide implementations of these technologies to accomplish this, various third-party products provide this capability.

2.4.5.2 Setting Up a Kerberized Connection from the CSP Gateway to Caché

To set up a secure, encrypted channel between the CSP Gateway and the Caché server, you need a Kerberos principal that represents the Gateway. This principal establishes an encrypted connection to Caché, and all information is transmitted through the connection. This allows an end-user to authenticate to Caché and prevents any snooping during that process.

Note: For information on setting up a connection between the CSP Gateway and the Caché server that is protected by SSL/TLS, see the "Configuring the CSP Gateway to Connect to Caché Using SSL/TLS" section of the "Using SSL/TLS with Caché" chapter.

The procedure is:

1. Determine or choose the name of the Kerberos principal that represents the Gateway.

   For Windows, this is the principal name representing the Gateway host's network service session (that is, the name of the machine hosting the Gateway with the "$" appended to it — *machine_name*$, , such as Athens$). For other platforms, this is any valid principal name entered as the username in the Gateway configuration screen; this identifies the appropriate key in the key table file.

2. Create a user in Caché with the same name as the Gateway's Kerberos principal. To do this, follow the instructions in the section "Creating a New User" in the "Users" chapter.

3. Give that user permissions to use, read, or write any required resources (these are also known as privileges). This is done by associating those privileges with a role and then associating the user with the role.

4. Configure the **`%Service_CSP`** service. To do this, complete the fields described in the section "Service Properties" in the "Services" chapter.

5. Configure the Gateway so that it can contact the server. The procedure is:

a. From the Management Portal home page, go to the **Web Gateway Management** page (**System Administration** > **Configuration** > **CSP Gateway Management**).

b. On the Web Gateway management page, there are a set of choices on the left. Under **Configuration**, click **Server Access**. This displays the **Server Access** page.

c. On the **Server Access** page, you can add a new configuration or edit an existing one. To add a new configuration, click the **Add Server** button; to edit an existing one, select it from the list on the left, select the **Edit Server** radio button, and click **Submit**. This displays the page for editing or configuring server access parameters. In addition to the general parameters on this page (described on its help screen), this page allows you to specify security-related parameters for the Gateway. For Kerberos connections, these are:

   - **Connection Security Level** — Choose the kind of protection that you would like Kerberos to attempt to provide this connection. (Note that this must match or exceed the type of security specified for the CSP service in the previous step.)
   - **User Name** — The name of the Kerberos principal that represents the Gateway. (This must be the same principal as was used in the first step of this process.)
   - **Password** — Do not specify a value for this. (This field is used when configuring the Gateway for use with Caché login.)
   - **Product** — Caché or Ensemble, depending on which product you are using.
   - **Service Principal Name** — The name of the principal that represents the Caché server. This is typically a standard Kerberos principal name, of the form "cache/machine.domain", where `cache` is a fixed string indicating that the service is for Caché, *machine* is the machine name, and *domain* is the domain name, such as "intersystems.com".
   - **Key Table** — When connecting to an instance of Caché on Windows, leave this field blank; for other operating systems, provide the name of the keytab file containing the permanent key belonging to the CSP Gateway, including the full path.

   After entering all these values, click the **Save Configuration** button to save them.

The CSP service is now ready to configured. This means that it can now provide the necessary underlying infrastructure to support a web application.

When creating a secured web application, the application developer needs to:

1. Choose an authentication method.
2. Configure the roles for the application.
3. If required, make sure the browser-to-Web server connection uses SSL.

2.5 Configuring for Operating-System–Based Authentication

Operating-system–based authentication (sometimes called "OS-based authentication") is only available for local processes, namely:

- Callin (**`%Service_Callin`**)
- Console (**`%Service_Console`**)

- Terminal (**%Service_Terminal**)

To set up the use of this type of authentication, the procedure is:

1. On the **Authentication/CSP Session Options** page (**System Administration** > **Security** > **System Security** > **Authentication/CSP Session Options**), select **Allow Operating System authentication**.
2. On to the **Services** page (**System Administration** > **Security** > **Services**) and select the service from the **Name** column. This displays the **Edit Service** page for the service.
3. On the **Edit Service** page, choose operating-system–based (the **Operating System** check box).

   Click **Save** to use the settings.

This type of authentication requires no other configuration actions.

Note: On Windows, when logged in using a domain account, OS-based and Kerberos authentication are the same.

2.5.1 A Note on %Service_Console

Since the console (**%Service_Console**) is a Windows-based service and Windows domain logins typically use Kerberos, console's OS-based authentication provides authentication for local logins.

2.5.2 A Note on %Service_Callin

With callin (**%Service_Callin**), OS-based authentication is only available from an OS-level prompt. When using callin programmatically, OS-based authentication is not supported — only unauthenticated access is available.

2.6 Configuring for Authentication with Caché Login

The services available for authentication with Caché login are:

- **%Service_Binding**
- **%Service_CSP**
- **%Service_CacheDirect**
- **%Service_CallIn**
- **%Service_ComPort**
- **%Service_Console**
- **%Service_Telnet**
- **%Service_Terminal**

For a service to use Caché login, you must configure it as follows:

1. On the **Authentication/CSP Sessions Options** page (**System Administration** > **Security** > **System Security** > **Authentication/CSP Session Options**), enable authentication with Caché login by selecting **Allow Password authentication**).
2. For the particular service, go to the **Services** page (**System Administration** > **Security** > **Services**) and select that service, such as **%Service_Bindings**, in the **Name** column; this displays the **Edit Service** page for the service.
3. On this page, choose Caché login, listed simply as **Password** from the list of authentication types.

4. Click **Save** to save this setting.
5. In addition to this basic procedure, certain services require further configuration. This is described in the following sections:
   - Web
   - ODBC
   - Telnet and Caché Direct

2.6.1 Web

For web access, you can optionally require that the CSP Gateway authenticate itself to the Caché server through Caché login. To perform this configuration, the procedure is:

1. From the Management Portal home page, go to the **Web Gateway Management** page (**System Administration** > **Configuration** > **CSP Gateway Management**).
2. On the Web Gateway management page, there are a set of choices on the left. Under **Configuration**, click **Server Access**. This displays the **Server Access** page.
3. On the **Server Access** page, you can add a new configuration or edit an existing one. To add a new configuration, click the **Add Server** button; to edit an existing one, select it from the list on the left, select the **Edit Server** radio button, and click **Submit**. This displays the page for editing or configuring server access parameters. In addition to the general parameters on this page (described on its help screen), this page allows you to specify security-related parameters for the Gateway. For Caché login connections, these are:
   - **Connection Security Level** — Choose **Password** from the drop-down list to use Caché login.
   - **User Name** — The user name under which the Gateway service runs (the installation process creates the CSPSystem user for this purpose). This user (CSPSystem or any other) should have no expiration date; that is, its Expiration Date property should have a value of 0.
   - **Password** — The password associated with the user account just entered.
   - **Product** — Caché or Ensemble, depending on which product you are using.
   - **Service Principal Name** — Do not specify a value for this. (This field is used when configuring the Gateway for use with Kerberos.)
   - **Key Table** — Do not specify a value for this. (This field is used when configuring the Gateway for use with Kerberos.)

   After entering all these values, click the **Save Configuration** button to save them.

It is important to remember that the authentication requirements for the Gateway are not directly related to those for an application that uses the Gateway. For example, you can require Caché login as the authentication mechanism for a web application, while configuring the Gateway to use Kerberos authentication — or no authentication at all. In fact, choosing a particular authentication mechanism for the Gateway itself makes no technical requirement for the web application, and vice versa. At the same time, some pairings are more likely to occur than others. If a web application uses Kerberos authentication, then using any other form of authentication for the Gateway means that Kerberos authentication information will be flowing through an unencrypted channel, thereby potentially reducing its effectiveness.

With a web application that uses Caché login, the username and password of the end-user are passed from the browser to the Web server, which then hands them to the co-located CSP Gateway. Since the Gateway has its own connection to the Caché server, it then passes the username and password to the Caché server. To establish its connection to the Caché server, the Gateway uses the CSPSystem account, which is one of the Caché predefined accounts.

By default, all these transactions are unencrypted. You can use SSL to encrypt messages from the browser to the Web server. You can use Kerberos to encrypt messages from the Gateway to the Caché server as described in the section "Setting Up a Secure Channel for a CSP Connection"; if you are not using Kerberos, you may prefer to physically secure the connection between the host machines, such as by co-locating the Gateway and Caché server machines in a locked area with a direct physical connection between them.

2.6.2 ODBC

Caché supports Caché login for ODBC connections among all its supported platforms. This requires client-side configuration. The ways of configuring client behavior vary by platform:

- On non-Windows platforms, use the Caché ODBC initialization file to specify name-value pairs that provide connection information . This file is described generally in *Using Caché ODBC*. The file has the following variables relevant to Caché login:
  - *Authentication Method* — Specifies how the ODBC client authenticates to the DSN. 0 specifies Caché login; 1 specifies Kerberos.
  - *UID* — Specifies the name for the default user account for connecting to the DSN. At runtime, depending on application behavior, the end-user may be permitted to override this value with a different user account.
  - *Password* — Specifies the password associated with the default user account. If the end-user has been permitted to override the UID value, the application will accept a value for the newly specified user's password.
- On a Windows client, you can specify connection information either through a GUI or programmatically:
  - Through a GUI, there is an ODBC DSN configuration dialog. Caché provides options on the **System DSN** tab. This screen has associated help that describes its fields. The path from the Windows Start menu to display this screen varies by version of Windows; it may be listed in the **Windows Control Panel**, under **Administrative Tools**, on the screen for **Data Sources (ODBC)**.
  - Programmatically, the **SQLDriverConnect** function is available, which accepts a set of name-value pairs. **SQLDriverConnect** is a C call that is part of the ODBC API and is documented at the Microsoft Web site. Its name-value pairs are the same as those for the initialization file available on non-Windows platforms, except that the password is identified with the *PWD* keyword.

2.6.3 Telnet and Caché Direct

When establishing a connection using Caché Direct and the Caché Telnet server for Windows, the client uses configuration information that has been stored as part of a Caché remote server. To configure a remote server, go to the client machine. On that machine, the procedure is:

1. Click on the Caché cube and select **Preferred Server** from the menu (the **Preferred Server** choice also displays the name of the current preferred server).
2. From the submenu that appears, choose **Add/Edit**.
3. To create a new remote server, click the **Add** button; to configure an already-existing server, choose the Caché server to which you are connecting and click the **Edit** button.
4. This displays the **Add Connection** dialog. In the **Authentication Method** area on that dialog, click **Password** for Caché login.
5. If you are editing the values for an already-existing server, there should be no need to change or add values for the more general fields in this dialog, as they are determined by the server that you chose to edit.

If you are adding a new server, the fields to complete are described in the section "Define a Remote Server Connection" of the "Connecting to Remote Servers" chapter of the *Caché System Administration Guide*.

6. Click **OK** to save the specified values and dismiss the dialog.

Important: When connecting to a non-Windows machine using telnet, there is no Caché telnet server available — you simply use the telnet server that comes with the operating system. Once you have established the connection to the server machine, you can then connect to Caché using the **%Service_Terminal** service.

2.7 Configuring Two-Factor Authentication

In addition to the authentication mechanism in use, Caché supports the use of *two-factor authentication*. This means that Caché authentication can require the end-user to possess two separate elements or "factors." From the end-user's perspective, the first factor is something that you know — for example, a password; the second factor is something that you have — for example, a smart phone. Caché performs two-factor authentication on its end-users using either of two mechanisms:

- *SMS text authentication* — Caché sends a security code to the end-user's phone via SMS. The end-user enters that code when prompted.
- *Time-based one-time password (TOTP)* — The end-user initially receives a secret key from Caché. That key is a *shared secret* between Caché and the end-user's application (such as an app on a mobile phone) or physical authentication device; both use the key and other information to generate a TOTP that serves as a verification code and that the end-user enters at a Caché prompt. The TOTP expires after 60 seconds and the end-user can only use it a single time, which is why it is called *time-based* and *one-time*.

This section covers the following topics:

2.7.1 Overview of Setting Up Two-Factor Authentication

The major steps to setting up two-factor authentication are:

1. Enable and configure two-factor authentication for the instance as a whole. You can configure the instance to use SMS text authentication, TOTP authentication, or both. For details about TOTP authentication, see the "Two-Factor TOTP Overview" section.
2. For SMS text authentication, configure the mobile phone service provider(s), if necessary. This includes:
   - Adding any mobile phone service providers if any are required and are not included in the list of default providers.
   - Changing configuration information as necessary for any existing providers (default or added).
3. Configure the service, as appropriate:
   - **%Service_Bindings** — Enable two-factor authentication for the service and continue to the next step.

- **%Service_Console** and **%Service_Terminal** — Simply enable two-factor authentication for the service. This is all that is required.
- **%Service_CSP** — There is no central means of enabling two-factor authentication for **%Service_CSP**. Continue to the next step.

You can enable either or both types of authentication for each service. For more information about services, see the "Services" chapter.

4. Configure client/server applications and web applications, as appropriate:
   a. For client/server applications (those that use **%Service_Bindings**), add the appropriate calls into the client application to support it; this is a programming task that varies according to the client-side component in use (for example, ODBC, JDBC, C++, Java, .NET, Python, or Perl).

      Important: Two-factor authentication is designed to receive a response from a human end-user in real time. If what the end-user considers a single session actually consists of multiple, sequential sessions, then the repeated prompting for the second factor may result in an unexpectedly difficult user experience. With client/server applications, the underlying protocol often causes clients to establish, disconnect, and reestablish connections repeatedly; such activity makes the use of two-factor authentication less desirable for this type of application.

   b. For web applications (those that use **%Service_CSP**), configure each application to support it.

   Note: For the Caché Terminal, which uses the **%Service_Console** service on Windows and the **%Service_Terminal** service on other operating systems, there is no configuration required other than server-side setup; since Caché controls the prompting in these, it simply follows the standard prompt (regardless of the authentication mechanism) with the two-factor authentication prompt and processes end-user input accordingly.

5. If you are using delegated authentication, modify the **ZAUTHENTICATE.mac** routine as required. See "Using Delegated Authentication" for more information.
6. Configure each end-user to enable SMS text authentication or TOTP authentication. An end-user can be configured to use both mechanisms, but cannot have both mechanisms enabled simultaneously.

2.7.1.1 Two-Factor TOTP Overview

Two-factor authentication using a time-based one-time password (TOTP) authentication works as follows:

1. As a requirement, each end-user must have either an authentication device or an application that generates such passwords. For example, end-users can use one of the following apps for mobile phones:
   - Google Authenticator, for Android, BlackBerry, or iPhone
   - Duo Mobile, for Android or iPhone
   - Amazon AWS MFA, for Android
   - Authenticator, for Windows Phone 7
2. When you configure an end-user for two-factor TOTP authentication, the system generates a secret key, which is displayed as a base-32 encoded randomized bit string. Caché and the end-user share this secret key (which is why it is known as a *shared secret*). Both Caché and the end-user's authentication device or application use it to generate the TOTP itself, which serves as a verification code. The TOTP, which the end-user enters into a **Verification code** field or prompt, is a string of six digits, and a new one is generated at a regular interval (thirty seconds, by default).

3. At login time, after the end-user provides Caché with a password, Caché then additionally prompts for the TOTP. The end-user provides the TOTP, and then completes the login process.

The end-user can get the secret key from Caché in several ways:

- When you configure the end-user's account to support two-factor TOTP authentication, the **Edit User** page for the end-user displays the end-user's secret key, as well as the name of the issuer and the end-user's account name. It also displays a QR code that includes all this information (a QR code is a machine-readable code such as the one pictured below). The end-user can then enter the information into an authentication device or an application by scanning the code or entering the information manually.
- If you choose to show the end-user their secret key during the login to a web application or the Terminal session (using **%Service_Console** or **%Service_Terminal**), you can enable this behavior by selecting the **Display Time-Based One-time Password QR Code on next login** field on the **Edit User** page. The Terminal session will then display the end-user's issuer, account, and secret key. A web application will display the end-user's issuer, account, and secret key, along with a QR code; here, the end-user can then scan the code or enter the information manually.

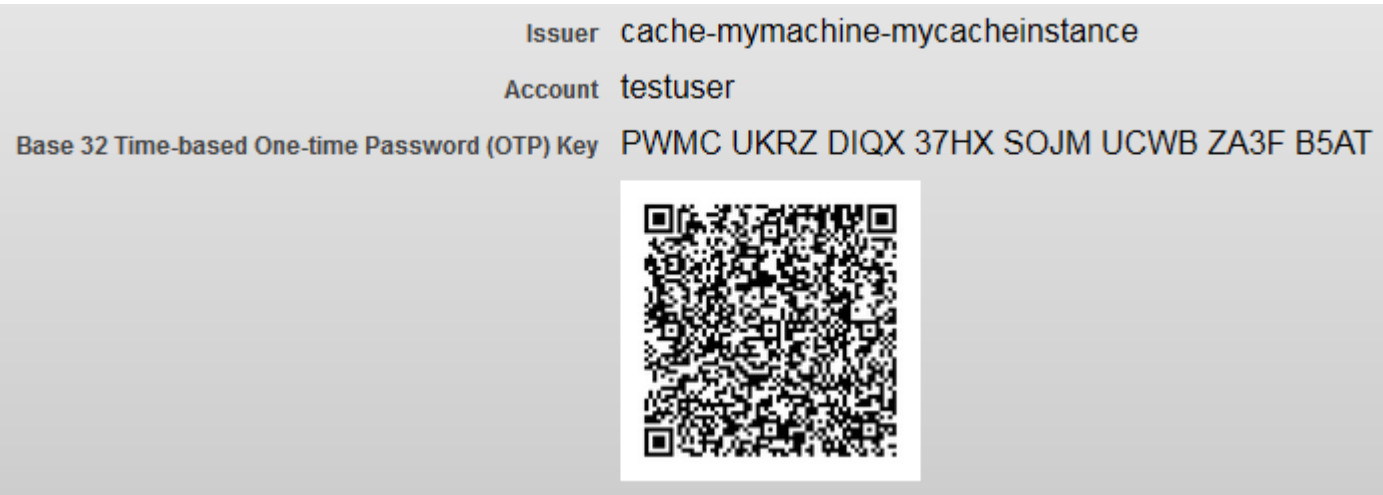
  Important: InterSystems does not recommend this option. See the following caution for more details.

CAUTION: The following are critical security concerns when using two-factor TOTP authentication:

- Do *not* transmit the secret key or QR code in an unsecured environment. Out-of-band transmission is preferable to transmission even on a secure network. (The secret key gives an end-user the means to log into Caché or a Caché application. If you and your end-users do not ensure the secret key's safety, then an attacker may gain access to it, which renders it useless for security.)
- When configuring two-factor TOTP authentication for your organization, InterSystems strongly recommends that you provide the secret key to each end-user in person or by phone, or that you have the end-user scan the QR code in the physical presence of an administrator. This provides the opportunity to authenticate the individual who obtains the secret key.

  Delivering the secret key over the network increases the possibility of exposing it. This includes displaying the secret key to the end-user when they first log into a web application, console, or the Terminal; this also includes displaying the QR code to the end-user when they first log into a web application.

Figure 2–3: A TOTP Issuer, Account, Key, and QR Code



Note: If you are using two-factor TOTP authentication and wish to generate QR codes, Java 1.7 or higher must be running on the Caché server. Without Java, Caché can use two-factor TOTP authentication, but the end-user enters the values for the issuer, account, and key manually on the authentication device or in the application.

2.7.2 Configuring Two-Factor Authentication for the Server

The steps in configuring two-factor authentication for the Caché server are:

1. Enable and configure two-factor authentication for the instance as a whole. You can configure the instance to use SMS text authentication, TOTP authentication, or both.
2. For SMS text authentication, configure the mobile phone service provider(s), if necessary. This includes:
   - Adding any mobile phone service providers if any are required and are not included in the list of default providers.
   - Changing configuration information as necessary for any existing providers (default or added).

2.7.2.1 Enabling and Configuring Two-Factor Authentication Settings for an Instance

When setting up two-factor authentication for a Caché instance (server), you can enable one or both of:

- **Two-factor time-based one-time password authentication** (TOTP authentication)
- **Two-factor SMS text authentication**

To enable either form of two-factor authentication, the procedure is:

1. From the Management Portal home page, go to the **Authentication/CSP Session Options** page (**System Administration** > **Security** > **System Security** > **Authentication/CSP Session Options**).
2. To enable two-factor TOTP authentication, on the **Authentication/CSP Session Options** page, select the **Allow Two-Factor Time-Based One-Time Password Authentication** check box. This displays the **Two-Factor Time-Based One-Time Password Issuer** field; here, enter a string to identify this instance of Caché.
3. To enable two-factor SMS text authentication, on the **Authentication/CSP Session Options** page, select the **Allow Two-Factor SMS Text Authentication** check box. This displays the following fields:
   - **Two-Factor Timeout (secs)** — Optional timeout in seconds for entering the one-time security token.
   - **DNS name of SMTP server** — The DNS (Domain Name Service) name of the SMTP (Simple Mail Transfer Protocol) server that this instance of Caché is using to send SMS text messages, such as smtp.example.com (required).
   - **From (address)** — Address to appear in the "From" field of message (required).
   - **SMTP username** — Optional username for SMTP authentication (if the SMTP server requires it).
   - **SMTP Password** and **SMTP Password (confirm)** — Optional password (entered and confirmed) for SMTP authentication (if the SMTP server requires it).
4. Click **Save**.
5. If the instance is supporting SMS text authentication, configure mobile phone service providers as required. These procedures are described in the next section.

After completing this process for the instance itself, you may need to perform other configuration, such as for the instance's services, web applications, and client/server applications; you *will* need to configure the instance's users. "The Overview of Setting Up Two-Factor Authentication" provides general direction about this.

2.7.2.2 Configuring Mobile Phone Service Providers

The topics related to configuring mobile phone service providers are:

- Creating or Editing a Mobile Phone Service Provider
- Deleting a Mobile Phone Service Provider
- Predefined Mobile Phone Service Providers

Creating or Editing a Mobile Phone Service Provider

To create or edit a mobile phone service provider, the procedure is:

1. From the Management Portal home page, go to the **Mobile Phone Service Providers** page (**System Administration** > **Security** > **Mobile Phone**):
   - To create a new provider, click **Create New Provider**.
   - To edit an existing provider, click **Edit** on the provider's row in the table of providers.

   This displays the **Edit Phone Provider** page for the selected mobile phone service provider.
2. On the **Edit Phone Provider** page, enter or change the value for each of the following fields:
   - **Service Provider** — The name of the mobile phone service provider (typically, its company name).
   - **SMS Gateway** — The address of the server that the mobile phone service provider uses to dispatch SMS (short message service) messages.

Deleting a Mobile Phone Service Provider

To delete a mobile phone service provider, the procedure is:

1. From the Management Portal home page, go to the **Mobile Phone Service Providers** page (**System Administration** > **Security** > **Mobile Phone**).
2. On the **Mobile Phone Service Providers** page, in the row of the provider, click **Delete**.
3. When prompted to confirm the deletion, click **OK**.

Predefined Mobile Phone Service Providers

Caché ships with a predefined list of mobile phone service providers, each with its SMS (short message service) gateway preset. These are:

- AT&T Wireless — txt.att.net
- Alltel — message.alltel.com
- Cellular One — mobile.celloneusa.com
- Nextel — messaging.nextel.com
- Sprint PCS — messaging.sprintpcs.com
- T-Mobile — tmomail.net
- Verizon — vtext.com

2.7.3 Enabling or Disabling Two-Factor Authentication for a Service

Important: For **%Service_CSP**, there is no central location for enabling or disabling two-factor authentication. Enable or disable it for each application as described in "Configuring Web Applications for Two-factor Authentication."

To enable or disable two-factor authentication for **%Service_Bindings**, **%Service_Console**, and **%Service_Terminal**, procedure is:

1. From the Management Portal home page, go to the **Services** page (**System Administration** > **Security** > **Services**).
2. On the **Services** page, click the name of the service for which you wish to enable either form of two-factor authentication. This displays the **Edit Service** page for the service.
3. On the service's **Edit Service** page, select or clear the **Two-factor SMS** check box, **Two-factor Time-based One-time Password** check box, or both. Note that each of these check boxes only appear if two-factor authentication is enabled for the instance.
4. Click **Save**.

2.7.4 Configuring Web Applications for Two-Factor Authentication

Once you have enabled two-factor authentication for an instance, you must enable it for all web applications that will use it. The procedure to enable it for an application is:

1. From the Management Portal home page, go to the **Web Applications** page (**System Administration** > **Security** > **Applications** > **Web Applications**).
2. On the **Web Applications** page, for the application you wish to enable two-factor authentication, click the name of the application, which displays its **Edit** page.
3. On the **Edit** page, in the **Security Settings** section of the page, select or clear the **Two-factor SMS** check box, **Two-factor Time-based One-time Password** check box, or both. Note that each of these check boxes only appear if two-factor authentication is enabled for the instance.

For general information about the **Edit Web Application** page, see the "CSP Application Options" section of the "CSP Architecture" chapter of the *Using Caché Server Pages (CSP)* book.

Note: A web application cannot simultaneously support both two-factor authentication and web services.

2.7.5 Configuring an End-User for Two-Factor Authentication

To configure an end-user to receive a one-time security token for two-factor authentication, the procedure is:

1. From the Management Portal home page, go to the **Users** page (**System Administration** > **Security** > **Users**):
2. For an existing user, click the name of the user to edit; for a new user, begin creating the user by clicking **Create New User** (for details about creating a new user, see the "Creating a New User" section of the "Users" chapter). Either of these actions displays the **Edit** page for the end-user.
3. On the **Edit User** page, select **SMS text enabled** or **Time-based One-time Password enabled**, as appropriate.
4. If you select **SMS Text**, you must complete the following fields:
   - **Mobile phone service provider** — The company that provides mobile phone service for the user. Either select a provider from those listed or, if the provider does not appear in the list, click **Create new provider** to add a new provider for the Caché instance. (Clicking **Create a new provider** displays the **Create a New Mobile Phone Provider**

window, which has fields for the **Service Provider** and the **SMS Gateway**, the purpose of which are identical to those described in the section Creating or Editing a Mobile Phone Service Provider.)

- **Mobile phone number** — The user's mobile phone number. This is the second factor, and is where the user receives the text message containing the one-time security token.

5. If you select **Time-based One-time Password enabled**, the page displays the following fields and information:

   - **Display Time-Based One-time Password QR Code on next login** — Whether or not to display a QR code when the user next logs in. If selected, Caché displays the code at the next login and prompts the user to scan it into the authentication device or application, and then to provide the displayed token to complete the authentication process. By default, this option is not selected. InterSystems recommends that you do *not* use this option.
   - **Generate a new Time-based One-time Password Key** — Creates and displays both a new shared secret for the end-user and a new QR code.

     Important: If you generate a new time-based one-time password key for a user, the current key in the user's authenticator application will no longer work. Before logging in, the user must enter the new key into the authenticator, either by scanning the QR code or by manually entering it. (This does not affect existing sessions.)

   - **Issuer** — The identifier for the Caché instance, which you established when configuring two-factor TOTP authentication for the instance.
   - **Account** — The identifier for the Caché account, which is the account's username.
   - **Base-32 Time-Based One-Time Password (OTP) Key** — The secret key that the end-user enters into the authentication device or application.
   - **QR Code** — A scannable code that contains the values of the issuer, account, and secret key.

6. Click **Save** to save these values for the user.

If a service uses two-factor authentication and an end-user has two-factor authentication enabled, then authentication requires:

- For SMS text authentication, a mobile phone that is able to receive text messages on that phone.
- For TOTP authentication, an application or authentication device that can generate verification codes.

Otherwise, the end-user cannot authenticate:

- For SMS text authentication, the end-user must have a mobile phone and be able to receive text messages on that phone. This is the phone number at which the user receives a text message containing the one-time security token as an SMS text.
- For TOTP authentication, the user must have an authentication device or application that can either scan a QR code or that can accept the secret key and other information required to generate each TOTP (which serves as a verification code).

2.7.6 Configuring Bindings Clients for Two-Factor Authentication

Client/server connections use **`%Service_Bindings`**. For these connections, the code required to use two-factor authentication varies by programming language. (Note that Console, the Terminal, and web applications do not require any client-side configuration.) Supported languages include:

- C++
- Java and JDBC

- .NET
- ODBC
- Perl
- Python

Client-side code performs three operations:

1. After establishing a connection to the Caché server, it checks if two-factor authentication is enabled on the server. Typically, this uses a method of the client's connection object.
2. It gets the one-time security token from the user. This generally involves user-interface code that is not specifically related to Caché.
3. It provides the one-time security token to the Caché server. This also typically uses a connection object method.

Note: When a user logs in through **%Service_Bindings**, Caché does not present a QR code to scan. The user must have previously set up the authentication device or application.

Important: Studio, which connects to the Caché server using **%Service_Bindings**, does not support two-factor authentication.

2.7.6.1 C++

With C++, support for two-factor authentication uses two methods of the tcp_conn class:

- `bool tcp_conn::is_two_factor_enabled()`

  This method checks if two-factor authentication is enabled on the server. It returns a boolean; `true` means that two-factor authentication is enabled.

- `bool tcp_conn::send_two_factor_token(const wchar_t* token, Conn_err* err)`

  This method provides the two-factor authentication token to the server. It returns a boolean; `true` means that the user has been authenticated. It takes two arguments:

  – *token*, the two-factor authentication token that the user has received. Note that the client application is responsible for obtaining the value of the token from the user.

  – *err*, an error that the method throws if the user does not successfully authenticate.

Important: If two-factor authentication is enabled on the server and the client code does not implement two-factor authentication calls, then the server will drop the connection with the client.

The following example uses an instance of a connection called *conn*:

1. It uses that instance's methods to check if two-factor authentication is enabled.
2. It attempts to provide the token to the server and performs error processing if this fails. Note that the sample code here assumes that the application code has stored the one-time security token in a variable of type std::string; it then uses the **c_str** method of the string class to extract the one-time security token as a null-terminated string to pass to the server.

```
// Given a connection called "conn"
if (conn->is_two_factor_enabled()) {
  // Prompt the user for the one-time security token.
  // Store the token in the "token" variable of type std::string.
  Conn_err err;
  if (!conn->send_two_factor_token(token.c_str(), &err;))
      // Process the error from a invalid authentication token here.
}
```

Note: The light C++ binding does not support two-factor authentication.

2.7.6.2 Java and JDBC

With Java, support for two-factor authentication uses two methods of the CacheConnection class:

- `public boolean isTwoFactorEnabled() throws Exception`

  This method checks if two-factor authentication is enabled on the server. It returns a boolean; `true` means that two-factor authentication is enabled.

- `public void sendTwoFactorToken(String token) throws Exception`

  This method provides the one-time security token to the server. It takes one argument, *token*, the one-time security token that the user has received.

The following example uses an instance of a connection called *conn*:

1. It uses that instance's methods to check if two-factor authentication is enabled.
2. It attempts to provide the token to the server and performs error processing if this fails.

Important: If two-factor authentication is enabled on the server and the client code does not implement two-factor authentication calls, then the server will drop the connection with the client.

```
// Given a connection called "conn"
if (conn.isTwoFactorEnabled()) {
  // Prompt the user for the two-factor authentication token.
  // Store the token in the "token" variable.
  try {
    conn.sendTwoFactorToken(token);
  }
  catch (Exception ex) {
    // Process the error from a invalid authentication token here.
  }
}
```

2.7.6.3 .NET

For .NET, Caché supports connections with two-factor authentication with the managed provider and with ADO.NET. Support for two-factor authentication uses two methods of the tcp_conn class:

- `bool CacheConnection.isTwoFactorEnabledOpen()`

  This method opens a connection to the Caché server and checks if two-factor authentication is enabled there. It returns a boolean; `true` means that two-factor authentication is enabled.

- `void CacheConnection.sendTwoFactorToken(token)`

  This method provides the one-time security token to the server. It has no return value. It takes one argument, *token*, the one-time security token that the user has received. If there is a problem with either the token (such as if it is not valid) or the connection, then the method throws an exception.

Important: A client application makes a call to **isTwoFactorEnabledOpen** *instead of* a call to **CacheConnection.Open**. The **isTwoFactorEnabledOpen** method requires a subsequent call to **sendTwoFactorToken**.

Also, if two-factor authentication is enabled on the server and the client code does not implement two-factor authentication calls, then the server will drop the connection with the client.

The following example uses an instance of a connection called *conn*:

1. It uses that instance's methods to check if two-factor authentication is enabled.
2. It attempts to provide the token to the server and performs error processing if this fails.

```
// Given a connection called "conn"
try {
  if (conn.isTwoFactorEnabledOpen()) {
    // Prompt the user for the two-factor authentication token.
    // Store the token in the "token" variable.
    conn.sendTwoFactorToken(token);
  }
}
catch (Exception ex) {
  // Process exception
}
```

2.7.6.4 ODBC

With ODBC, support for two-factor authentication uses two standard ODBC function calls (which are documented in the Microsoft ODBC API Reference):

- `SQLRETURN rc = SQLGetConnectAttr(conn, 1002, &attr, sizeof(attr), &stringLengthPtr);`

  The SQLGetConnectAttr function, part of the Microsoft ODBC API, returns the current value of a specified connection attribute. The Caché ODBC client uses this function to determine if the server supports two-factor authentication. The value of the first argument is a handle to the connection from the client to the server; the value of the second argument is `1002`, the ODBC attribute that specifies whether or not two-factor authentication is supported; the values of the subsequent arguments are for the string containing the value of attribute 1002, as well as relevant variable sizes.

- `SQLRETURN rc = SQLSetConnectAttr(conn, 1002, unicodeToken, SQL_NTS);`

  The SQLSetConnectAttr function, also part of the Microsoft ODBC API, sets the value of a specified connection attribute. The Caché ODBC client uses this function to send the value of the two-factor authentication token to the server. The values of the four arguments are, respectively:

  – The connection from the client to the server.
  – `1002`, the ODBC attribute that specifies whether or not two-factor authentication is supported.
  – The value of the one-time security token.
  – `SQLNTS`, which indicates that the one-time security token is stored in a string.

Important: If two-factor authentication is enabled on the server and the client code does not implement two-factor authentication calls, then the server will drop the connection with the client.

The following example uses an instance of a connection called *conn*:

1. It uses **SQLGetConnectAttr** to check if two-factor authentication is enabled.
2. It attempts to provide the token to the server with the **SQLSetConnectAttr** call and performs error processing if this fails. If **SQLSetConnectAttr** fails, the server drops the connection, so you need to reestablish the connection before you can attempt authentication again.

```
// Given a connection called "conn"
SQLINTEGER stringLengthPtr;
SQLINTEGER attr;
SQLRETURN rc = SQLGetConnectAttr(conn, 1002, &attr, sizeof(attr), &stringLengthPtr);
if attr {
  // Prompt the user for the two-factor authentication token.
  wstring token;
  SQLRETURN rc = SQLSetConnectAttr(conn, 1002, token, SQL_NTS);
    if !rc {
      // Process the error from a invalid authentication token.
    }
}
```

2.7.6.5 Perl

With Perl, support for two-factor authentication uses two methods of the Intersys::PERLBIND::Connection class:

- `is_two_factor_enabled()`

  This method checks if two-factor authentication is enabled on the server. It returns a integer; 1 means that two-factor authentication is enabled and 0 means that it is disabled.

- `send_two_factor_token($token)`

  This method provides the two-factor authentication token to the server. It returns a integer; 1 indicates success and 0 indicates failure. Its argument, *$token*, is the two-factor authentication token that the user has received and then entered at the client prompt. Note that the client application is responsible for obtaining the value of the token from the user.

Important: If two-factor authentication is enabled on the server and the client code does not implement two-factor authentication calls, then the server will drop the connection with the client.

The following example uses an instance of a connection called *conn*:

1. It uses that instance's methods to check if two-factor authentication is enabled.
2. It attempts to provide the token to the server and performs error processing if this fails.

```
// Given a connection called "conn"
if ($conn->is_two_factor_enabled()) {
  # Prompt the user for the one-time security token.
  # Store the token in the "token" variable of type std::string.
  if (!$conn->send_two_factor_token($token)) {
    # Process the error from a invalid authentication token here.
  } else {
    # two-factor authentication succeeded
}
else {
  # Handle if two-factor authentication is not enabled on the server.
}
```

Caché comes with sample programs that demonstrate two-factor authentication with Perl. These programs are in the *install-dir*\dev\perl\ directory, and are samples\two_factor.pl. For more information on Perl sample programs for use with Caché, see the "Sample Programs" section of the "Caché Perl Binding" chapter of *Using Perl with Caché*.

2.7.6.6 Python

With Python, support for two-factor authentication uses two methods of the intersys.pythonbind.connection class:

- `is_two_factor_enabled()`

  This method checks if two-factor authentication is enabled on the server. It returns a boolean; `true` means that two-factor authentication is enabled.

- `send_two_factor_token(token)`

This method provides the two-factor authentication token to the server. It returns a boolean; `true` means that the user has been authenticated. It takes one argument, *token*, the two-factor authentication token that the user has received. Note that the client application is responsible for obtaining the value of the token from the user.

Note: Python leaves carriage returns in input. This means that, when processing the one-time security token, it is necessary to strip any carriage returns out of it.

Important: If two-factor authentication is enabled on the server and the client code does not implement two-factor authentication calls, then the server will drop the connection with the client.

The following example uses an instance of a connection called *conn*:

1. It uses that instance's methods to check if two-factor authentication is enabled.
2. It attempts to provide the token to the server and performs error processing if this fails.

```
// Given a connection called "conn"
if conn.is_two_factor_enabled():
  # Prompt the user for the one-time security token.
  # Store the token in the "token" variable of type std::string.
  # Make sure that the carriage returns are stripped from the string.
  if !conn.send_two_factor_token(token):
    # Process the error from a invalid authentication token here.
  else:
    # two-factor authentication succeeded
else:
  # Handle if two-factor authentication is not enabled on the server.
```

Caché comes with sample programs that demonstrate two-factor authentication with Python. These programs are in the *install-dir*\dev\Python\ directory; they are samples\two_factor.py for Python 2.6 and samples3\two_factor.py for Python 3.0. For more information on Python sample programs for use with Caché, see the "Sample Programs" section of the "Caché Python Binding" chapter of *Using Python with Caché*.

2.8 Other Topics

Areas covered in this section are:

- System Variables and Authentication
- Using Multiple Authentication Mechanisms
- Cascading Authentication
- Establishing Connections with the UnknownUser Account
- Programmatic Logins
- The JOB Command and Establishing a New User Identity

2.8.1 System Variables and Authentication

After authentication, two variables have values:

- *$USERNAME* contains the username
- *$ROLES* contains a comma-delimited list of the roles held by the user

The *$ROLES* variable allows you to manage roles programmatically. For more information on this, see the section "Programmatically Managing Roles" in the "Roles" chapter.

2.8.2 Using Multiple Authentication Mechanisms

The one situation in which InterSystems recommends the use of multiple authentication mechanisms is when moving from a less rigorous mechanism to a more rigorous one. For example, if an instance has been using no authentication and plans to make a transition to Kerberos, the following scenario might occur:

1. For the transition period, configure all supported services to allow both unauthenticated and Kerberos-authenticated access. Users can then connect using either mechanism.
2. If appropriate, install new client software (which uses Kerberos for authentication).
3. Once the list of Caché users has been synchronized with that in the Kerberos database, shut off unauthenticated access for all services.

The use of multiple authentication mechanisms is often in conjunction with cascading authentication, described in the next section.

2.8.3 Cascading Authentication

While Caché supports for a number of different authentication mechanisms, InterSystems recommends that you do not use any other password-based authentication mechanism along with Kerberos. Also, there are limited sets of circumstances when it is advisable for an instance to have multiple authentication mechanisms in use.

If a service supports multiple authentication mechanisms, Caché uses what is called "cascading authentication" to manage user access. With cascading authentication, Caché attempts to authenticate users via the specified mechanisms in the following order:

- Kerberos cache (includes Kerberos with or without integrity-checking or encryption)
- OS-based
- LDAP (with checking the LDAP credentials cache second)
- Delegated
- Caché login
- Unauthenticated

Note: If a service specifies Kerberos prompting and this fails, there is no cascading authentication. If a service specifies both Kerberos prompting and Kerberos cache, then Caché uses Kerberos cache only.

For example, if a service supports authentication through:

1. Kerberos cache
2. OS-based
3. Unauthenticated

If a user attempts to connect to Caché, then there is a check if the user has a Kerberos ticket-granting ticket; if there is such a ticket, there is an attempt to obtain a service ticket for Caché. If this succeeds, the user gets in. If either there is no initial TGT or a Caché service cannot be obtained, authentication fails and, so, cascades downward.

If the user has an OS-based identity that is in the Caché list of users, then the user gets in. If the user's OS-based identity is not in the Caché list of users, then authentication fails and cascades downward again.

When the final option in cascading authentication is unauthenticated access, then all users who reach this level gain access to Caché.

Note: If an instance supports cascading authentication and a user is authenticated with the second or subsequent authentication mechanism, then there have been login failures with any mechanisms attempted prior to the successful one. If the %System/%Login/LoginFailure audit event is enabled, these login failures will appear in the instance's audit log.

2.8.4 Establishing Connections with the UnknownUser Account

If Caché login and unauthenticated mode are both enabled, then a user can simply press **Enter** at the **Username** and **Password** prompts to connect to the service in unauthenticated mode, using the UnknownUser account. If only Caché login is enabled, then pressing **Enter** at the **Username** and **Password** prompts denies access to the service; Caché treats this as a user attempting to log in as the UnknownUser account and providing the wrong password.

2.8.5 Programmatic Logins

In some situations, it may be necessary for a user to log in after execution of an application has begun. For example, an application may offer some functionality for unauthenticated users and later request the user to log in before some protected functionality is provided.

An application can call the Caché log in functionality through the **Login** method of the $SYSTEM.Security class with the following syntax:

```
set success = $SYSTEM.Security.Login(username,password)
```

where

- *success* is a boolean where 1 indicates success and 0 indicates failure
- *username* is a string holding the name of the account logging in
- *password* is a string holding the password for the *username* account

If the username and password are valid and the user account is enabled and its expiration date has not been reached, then the user is logged in, *$USERNAME* and *$ROLES* are updated accordingly, and the function returns 1. Otherwise, *$USERNAME* and *$ROLES* are unchanged and the function returns 0.

No checking of privileges occurs as a result of executing **$SYSTEM.Security.Login**. As a result, it is possible that the process has lost privileges that were previously held.

There is also a one-argument form of **$SYSTEM.Security.Login**:

```
set success = $SYSTEM.Security.Login(username)
```

It behaves exactly the same as the two-argument form except that no password checking is performed. The single-argument form of **$SYSTEM.Security.Login** is useful when applications have performed their own authentication and want to set the Caché user identity accordingly. It can also be used in situations where a process is executing on behalf of a specific user but is not started by that user.

Note: The single-argument **Login** method is a restricted system capability as described in the section "Special Capabilities" in the "Resources" chapter.

2.8.6 The JOB Command and Establishing a New User Identity

When a process is created using the JOB command, it inherits the security characteristics (that is, the values of *$USERNAME* and *$ROLES*) of the process that created it. Note that all roles held by the parent process, User as well as Added, are inherited.

In some cases, it is desirable for the newly created process to have *$USERNAME* and *$ROLES* values that are different from its parent's values. For example, a task manager might be created to start certain tasks at certain times on behalf of certain users. While the task manager itself would likely have significant privileges, the tasks should run with the privileges of the users on whose behalf they are executing, not with the task manager's privileges.

The following pseudocode illustrates how this can be done:

```
WHILE ConditionToTest {
    IF SomeThingToStart {
        DO Start(Routine, User)
    }
}

Start(Routine, User) {
    NEW $ROLES    // Preserve $USERNAME and $ROLES

    // Try to change username and roles
    IF $SYSTEM.Security.Login(User) {
        JOB ...
        QUIT $TEST
    }
    QUIT 0       // Login call failed
}
```

3

Assets and Resources

Once a user has authenticated, the available resources are determined by the authorization facilities in Caché security. Authorization protects Caché components from inappropriate use, and involves the following concepts:

1. An *asset* is something that is protected. For instance, a Caché database is an asset, the ability to connect to Caché using SQL is an asset, and the ability to perform a backup is an asset.
2. Assets are protected via *resources*. Sometimes there is a one-to-one correspondence between assets and resources, that is a single asset (such as a database) is protected by one resource. In other cases, multiple assets are protected by a single resource, in order to simplify security management. For instance, a variety of system management functions are protected by a single resource.
3. A *privilege* grants permission to do something with one or more assets protected by a resource, such as being able to *read* the *orders database*. A privilege is written as a resource name followed by a permission separated by a colon, such as:

   ```
   %DB_Sales:Read
   ```

   For more information on privileges, see the chapter "Privileges and Permissions."

Caché includes a set of resources for assets that it protects — that is, to which it provides access for users based on the rights that they hold. You can also define your own resources.

This chapter addresses issues related to resources and the assets that they protect. Topics include:

- About Resources
- System Resources
- Database Resources
- Application Resources
- Creating or Editing a Resource
- Using Custom Resources with the Management Portal

3.1 About Resources

There are multiple resource types:

- System Resources — For controlling the ability to perform various actions for a Caché instance. For more information on these resources, see the "System Resources" section of this chapter.

These resources are **%Admin_Journal**, **%Admin_Manage**, **%Admin_Operate**, **%Admin_Secure**, **%Admin_Task**, **%Development**, and **%System_Callout**.

- Database Resources — For controlling read and write access to Caché databases. For more information on these resources, see the "Database Resources" section of this chapter.

  The database resources for a newly installed Caché instance are **%DB_CACHE**, **%DB_CACHEAUDIT**, **%DB_CACHELIB**, **%DB_CACHESYS**, **%DB_CACHETEMP**, **%DB_DOCBOOK**, **%DB_SAMPLES**, **%DB_USER**, and **%DB_%DEFAULT**.

- Service Resources — For controlling the ability to connect to Caché using various Caché connection technologies. For more information on these resources and the functionality that they control, see the chapter "Services."

  Not all services have associated privileges — only those services for which Caché provides user-based access; other services, such as database shadowing, are not user-based and, as a result, do not have associated security resources. For more information on managing services, see the chapter "Services."

  The service resources are **%Service_CSP**, **%Service_CacheDirect**, **%Service_Callin**, **%Service_ComPort**, **%Service_Console**, **%Service_Login**, **%Service_Object**, **%Service_SQL**, **%Service_Telnet**, and **%Service_Terminal**.

- DeepSee Resources — For controlling the ability to use various aspects of DeepSee. For more information on these resources, see the "Setting Up Security" chapter in the *DeepSee II Implementation Guide*.

- Application Resources — Either for controlling the whole of a user-defined application or for perform authorization checks anywhere in user code. For information on these resources generally, see the "Application Resources" section of this chapter. For information on creating these resources, see the "Creating or Editing a Resource" section of this chapter.

3.2 System Resources

Caché comes with a set of built-in resources that govern actions in relation to the installed Caché instance. (These were formerly known collectively as the administrative and development resources.) System resources include:

- Administrative Resources
- The %Development Resource
- The %System_Callout Resource
- The %Secure_Break Resource

System resources also include the resources associated with resource-based services. For more details on services, see the "Services" chapter.

3.2.1 Administrative Resources

If you receive privileges involving an administrative resource, then you can perform designated Caché systems administration tasks. Their associated permission is Use — Read and Write are not relevant for them.

%Admin_Journal

Having the Use permission on this resource allows users to set and clear the no-journaling process flag via the **DISABLE^%SYS.NOJRN** and **ENABLE^%SYS.NOJRN** entry points, respectively, in programmer mode in the Terminal. This resource allows you to establish users who can perform this action without having to give them the Use permission on the **%Admin_Manage** resource, which might give them more privileges than necessary or desired.

%Admin_Manage

Most visibly, this resource controls access to various pages in the Management Portal. Specifically, it controls the ability to:

- Create, modify, and delete Caché configurations.
- Create, modify, and delete backup definitions.
- Add databases, modify database characteristics, and delete databases.
- Modify namespace map.
- Perform database and journal restores.
- Set and clear the no-journaling process flag via the **ENABLE^%SYS.NOJRN** and **DISABLE^%SYS.NOJRN** entry points, respectively, in programmer mode in the Terminal. Note that if you wish for a user to be able to perform this task without other managerial privileges, use the **%Admin_Journal** resource.

%Admin_Operate

Most visibly, this resource controls access to various pages in the Management Portal. Specifically, it controls the ability to:

- Start and stop Caché.
- Examine and terminate processes.
- Mount and dismount databases.
- Perform integrity checks.
- Start, stop, and switch journals.
- Perform database backups.
- Examine and delete locks.
- Examine logs.
- Start and stop services.

The **%Admin_Operate:Use** privilege is required to mount a database, either explicitly (such as when using a Caché utility) or implicitly (such as when making a global reference to an un-mounted database).

%Admin_Secure

Most visibly, this resource controls access to various pages in the Management Portal. Specifically, it controls the ability to:

- Create, modify, delete users.
- Create, modify, delete roles.
- Create, modify, delete application definitions and application resources.
- Modify audit settings.
- Modify services.

%Admin_Tasks

Most visibly, this resource controls the ability to create, modify, or run a task, such as through the Management Portal's Task Manager (**System Operation** > **Task Manager**).

Note that users holding privileges on **%Admin_*** resources can carry out administrative functions without having any database privileges (**%DB_<database-name>:R/W**). For example, a user (presumably a system operator) holding the **%Admin_Operate:Use** privilege can perform backups without holding privileges on any of the databases being backed up. This is as it should be, since there is no reason for an operator to have access to the contents of databases other than through applications such as the Caché database backup system.

3.2.2 The %Development Resource

The **%Development** resource controls access to Caché development facilities and various pages in the Management Portal. Specifically, it controls the ability to:

- Enter direct mode.
- Use Studio. The **%Development:Use** privilege is checked whenever the Studio connects to a server.
- Use the global, routine, class, table, or SQL capabilities of the Caché system manager utility. (This privilege is also required to call any APIs that provide programmatic access to this functionality.)
- Use the debugging facilities of Caché, including the **BREAK** and **ZBREAK** commands and the debug option of the process display in the Caché system manager utility.

The **%Development:Use** privilege works in conjunction with database privileges to control developer access to Caché as follows:

- For Studio, the **%Development:Use** privilege is checked whenever the Studio connects to a server. In order to connect, the user must have the **%Development:Use** privilege for the server and be able to read the default global database for the namespace (that is, have the **%DB_<database-name>:R** privilege for it). In order to open a routine, class or other definition, the user must have the Read privilege for the database in which it is stored (which may or may not be the default routine database). In order to compile or save a definition, the user must have the Write privilege for that database.
- For the global, routine, or class capabilities of the Caché system manager utility, the user must have the Read or Write privileges for the database to access or modify globals.
- For the SQL capabilities of the Caché system manager utility, the user must have the appropriate SQL privileges for the tables, views, stored procedures, or other SQL assets. If you have some form of SQL access to a table in a database, you are also granted Read or Write access to that database.

To debug a Caché application, you need no specific database privileges. If you hold the **%Development:Use** privilege for the system, you can set a breakpoint in any routine stored in any database on that system. However, you must have the Read privilege for a database to:

- View routine source via the debugger
- Execute a routine

3.2.3 The %System_Callout Resource

The **%System_Callout** resource controls access to various tools that perform actions outside of Caché. These include:

- In ObjectScript, using the $ZF(-100) function, which supports invoking operating system commands from within ObjectScript code. Also see "Issuing Operating System Commands" in *Using the Callout Gateway*, which includes detailed instructions for adding the **%System_Callout:Use** privilege.
- At the Terminal of the Terminal, using "!" and "$" as control characters for operating system access. For details, see the $ZF(-100) documentation.
- In Zen Reports, using the built-in PDF rendering engine.

- In local interprocess communication with ObjectScript, opening an interprocess communications device in Q mode. For details, see the "OPEN-only Command Keywords for Interprocess Communications Pipes" table in the "Local Interprocess Communication" chapter of the *Caché I/O Device Guide*.
- In the MultiValue shell, using the SH and DOS commands.

Note: **%System_Callout** also controls interactions with the deprecated $ZF(-1) and $ZF(-2) functions.

3.2.4 The %Secure_Break Resource

The **%Secure_Break** resource enforces the use of the secure debug shell, which restricts programmer access at a <BREAK> prompt. For more information on the secure debug shell, see the "The Secure Shell" section in the "System Management and Security" chapter.

3.3 Database Resources

Database resources control access to the contents of Caché databases. The name of the database resource that governs access to a database is stored in the label block of that database.

All database resource names must start with the string "%DB_" and, for custom resources, the first character after the underscore should not be the percent sign. The default database resource name is **%DB_<database-name>**. You can change the resource name assigned to a database by using the Management Portal.

3.3.1 Database Resource Privileges

The available database privileges are:

Table 3–1: Database Privileges

Permission	Enables
Read	Data access and routine execution
Write	Modification and deletion of data (including executable code)

Read and Write permissions provide access to all contents of a database, including source and executable code as well as data. Caché security management utilities automatically grant the Read permission for any database resource where there is Write access.

Database privileges do not enable protection of individual items, such as routines or globals, within a database. Rather, the same protection is applied to all items of a resource within a database. You can establish higher granularity of protection by storing globals and routines in separate databases. Caché namespace mapping allows you to do this without any application-level modifications.

Note: SQL security grants table-level access, specifying which particular action can be performed, such as **SELECT** or **UPDATE**. For more information on SQL and security, see the chapter "SQL Security."

3.3.2 Shared Database Resources

Often, there is a one-to-one correspondence between databases and the resources used to protect them. For instance, protection for the CACHESYS database is specified using the **%DB_CACHESYS** resource and protection for the SAMPLES database

is specified using the **%DB_SAMPLES** resource. However, this is not a requirement and, when several databases share the same security definitions they can share the same security resource.

Consider a sales application with three databases. Rather than define access for each of these individually, the system manager has the choice option to:

1. Create a new database resource, such as **%DB_SALES**.
2. Assign this resource to the three databases.
3. Specify suitable access to **%DB_SALES** which then governs access to all three databases.

3.3.3 Default Database Resource

When mounting an existing database that has no database resource name, Caché assigns the default resource, **%DB_%DEFAULT**, to the database. By default, **%DB_%DEFAULT** has the following permissions:

Table 3–2: %DB_%DEFAULT Privileges

Role	Permissions
%Developer	Read, Write
%Manager	Read, Write

While you can change the privileges associated with **%DB_%DEFAULT** resource, you cannot delete the **%DB_%DEFAULT** resource itself, since it must be available if an unnamed database is mounted.

3.3.4 Unknown or Non-Valid Resource Names

With one exception (see below), if you attempt to mount a database that has an unknown or invalid database resource name, the attempt fails. (This might occur if a database were moved from one Caché instance to another.) An automatic mount attempt fails with an error and an explicit mount attempt prompts you with the choice of creating the resource named in the database label or changing the database to use a valid resource.

The one exception to this rule is that a user who is a member of the **%All** role can mount a database that does not have a resource (such as if its resource was deleted or the database was previously on a different system).

3.3.5 Namespaces

Users and applications interact with Caché databases through namespaces. While there are no privileges associated with namespaces, access to a namespace is granted or denied based on the privileges associated with the underlying databases. More specifically, to access a namespace, you must hold the Read privilege on the default globals database associated with that namespace. This requirement is checked when:

- A process attempts to change to a different namespace, such as by using the **$NAMESPACE** special variable, the **ZNSPACE** command, or the **%CD** utility
- There is an attempt to connect to Caché using any service that connects to a namespace, such as an SQL connection or an object connection

Note: This requirement is not checked when a global or routine reference is made, implicitly or explicitly, to a namespace.

The fact that namespace privileges depend on privileges for the underlying databases can lead to unexpected behavior. For example, suppose that namespace NSCust refers to data in three databases: DBCust1, DBCust2, and DBCust3. Suppose also

that the role **AverageUser** has the privileges **%DB_DBCust1:R** and **%DB_DBCust3:R**. Because the role has no privilege associated with DBCust2, any attempt to access data in that database fails (including if it is through the namespace).

3.3.6 Databases that Ship with Caché

A number of databases ship with Caché. Some of these have special characteristics, where described in the following sections. These include:

- CACHESYS
- SAMPLES

3.3.6.1 CACHESYS, the Manager's Database

Caché ships with a database that provides a repository for administrative routines and globals. This is the CACHESYS database, and is sometimes known as the *manager's database.*

Within this database, there are groups of globals and routines whose names begin with the percent sign (these are often known as "percent globals" or "percent routines"). These globals and routines have a special role in the management of a Caché site and have special rules that apply to them:

- All users have Read permission for percent routines and percent globals.

  Note that via mappings, it is possible to change where these items are stored, but that has no effect on their visibility. Percent globals and percent globals are always visible in all namespaces.
- All percent routines have Write permission for *all* globals located in the same database (percent as well as non-percent). For instance, percent routines in the CACHESYS database have Write access to globals stored in that database, but not to globals in any other database. Simultaneously, percent routines in any other database have implicit Write access to globals stored in that same database but *not* to percent globals in CACHESYS. This implicit Write permission is only available during normal routine execution. It is disabled if the routine has been modified and it is not available in **XECUTE** commands or through argument indirection.
- You can control Write access to percent globals from non-percent routines with the **Enable writing to percent globals** field on the **System-wide Security Parameters** page (**System Administration** > **Security** > **System Security** > **System-wide Security Parameters**); this page is described in the System-wide Security Parameters section of the "System Management and Security" chapter.

CAUTION: Do not move, replace, or delete the CACHESYS database.

Special Capabilities

There are special capabilities available to code located in the CACHESYS database. These capabilities are sometimes called "restricted system capabilities." They are:

- Invoking protected **VIEW** commands and **$VIEW** functions.
- Using protected class methods.
- Modifying the roles of a process with a `SET $ROLES = ...` call.
- Invoking the single-argument form of the **$SYSTEM.Security.Login** function (which is the **Login** method of the %SYSTEM.Security class).
- Invoking the two-argument form of the **$SYSTEM.Security.ChangePassword** function (which is the **ChangePassword** method of the %SYSTEM.Security class). (Note that the new password must conform to the general password constraints described in "Properties of Users" section of the "Users" chapter and the instance-specific password constraints described in the "Password Strength and Password Policies" section of the "System Management and Security" chapter.

Note: You need no database privileges to read or write database blocks with the **VIEW** command.

The only code that can perform these actions is:

- Routines stored in the CACHESYS database, but only during "normal" routine execution. They are disabled if a **ZINSERT** into the current routine has modified the routine, and they are also not available in **XECUTE** commands or through argument indirection.
- Processes with the Write permission on the **%DB_CACHESYS** resource.

3.3.6.2 The SAMPLES Database

Caché ships with a database of example data, called the SAMPLES database. By default, all users are granted SQL access to this database. Further, if a user has any SQL system privileges in the namespace (or if a user has any SQL privileges for any table, view, or procedure in a namespace), then the user is granted the **%SQL** role and the database role. For this reason, all users can connect to the SAMPLES database.

3.4 Application Resources

Caché supports several forms of custom authorization, all of which depend on user-defined resources, known as Application resources. These include:

- Supplementary authorization checking for a Portal page — for more information, see the section "Using Custom Resources with the Management Portal."
- Authorization checking at a particular point in an application — for more information, see the next section, "Creating or Editing a Resource."
- Authorization for the whole of an application

For the whole of an application, Caché allows you to create an application definition associated with a user-defined application (which itself is defined as a named entity composed of executable code). Application resources then allow you to perform authorization checking for the application. There are three types of applications:

- Web application definitions — These are associated with a specific CSP or Zen application.
- Privileged Routine application definitions — These are associated with one or more routines.
- Client application definitions — These are associated with one or more Caché Direct executables.

Application resources provide a means of controlling access to applications. To use this feature, create a custom resource as described in the next section, "Creating or Editing a Resource" and use it in association with the application as described in the "Creating and Editing an Application: The General Tab" section of the "Applications" chapter.

For example, if a Zen application has an associated resource, then users can only run the application if they have Use permission on the resource. If an application uses other resource-regulated entities, such as databases, then users must also have appropriate permissions on those resources in order to operate the application effectively. For more information on applications, consult the "Applications" chapter.

3.5 Creating or Editing a Resource

To create a new resource, on the **Resources** page (**System Administration** > **Security** > **Resources**), click **Create New Resource**.

To edit an existing resource, on the **Resources** page (**System Administration** > **Security** > **Resources**), click the **Edit** button to the right of the resource you wish to edit.

This displays the **Edit Resource** page. The **Edit Resource** page has fields for the following:

- Resource Name — The string by which the resource is identified. For more information on resource names, see the section "Resource Naming Conventions." When creating a resource, this is an editable field; when editing an existing resource, this is a non-editable, displayed string.
- Description — Optional text related to the resource.
- Public Permission —
  - Read — When selected, specifies that all users can view this resource.
  - Write — When selected, specifies that all users can view or change this resource.
  - Use — When selected, specifies that all users can run or otherwise employ this resource.

Once you have added a resource, it appears in the table of resources and is of type Application. You can then use it as part of application-specific authorization. See the section "Checking Privileges" in the "Privileges and Permissions" chapter for more information.

3.5.1 Resource Naming Conventions

The names of Caché resources begin with a percent sign character. The names of application-defined resources should not begin with a percent sign character.

Resource names are not case-sensitive. This means that:

- Names are defined using mixed case and the name is preserved exactly as it is entered.
- Names that differ only by case are prohibited.
- When a name is looked up, Caché ignores differences in case.

For example, if there is a resource named "Accounting". An attempt to create another resource named "ACCOUNTING" will fail. References to the resource using name values of "Accounting", "accounting", and "ACCOUNTING" will all succeed.

3.6 Using Custom Resources with the Management Portal

By default, the **%Admin_Manage**, **%Admin_Operate**, **%Admin_Secure**, and **%Development** system resources control access to the Management Portal. As a supplement to these that allows for more granular Portal security, you can associate an additional custom resource with each Portal page. If a Portal page has an associated custom resource, then the user must hold both the system and custom resource for the page in order to view that page.

For example, access to the **Lock Table** page requires the **%Operator** role. You can also associate a custom resource (for example, called **MyLockTable**) with the **Lock Table** page; once you have created this association, a user must both be a member of the **%Operator** role and also have the **MyLockTable:Use** privilege in order to view the **Lock Table** page. With this arrangement, the **%Operator** role grants access to fewer pages than in an instance with default settings, and you can define a new role that can view the **Lock Table** page and all the other pages to which **%Operator** role grants access. You can also create multiple custom resources, so that various roles would have access to various subsets of what a predefined role would have available by default.

This section describes:

- Defining and Applying a Custom Resource to a Page
- Removing a Custom Resource from a Page

Important: Because there can be complex interactions among the various pages, resources, and roles, system administrators should plan carefully before implementing custom resources for the Management Portal.

3.6.1 Defining and Applying a Custom Resource to a Page

To define and apply a custom resource, the procedure is:

1. Log in as a user who holds the **%Admin_Secure:Use** privilege or is a member of the **%All** role.
2. Create the custom resource. To do this, on the **Resources** page (**System Administration** > **Security** > **Resources**), click **Create New Resource**. When creating the resource, make sure that you properly set its public permissions according to the instance's needs.
3. Associate the privilege to use the custom resource with a role. For an existing role, on the **Roles** page (**System Administration** > **Security** > **Roles**), simply add the privilege to the role; alternately, (also on the **Roles** page), create a new role and then add the privilege to it immediately after creating it. Either way, the privilege consists of the custom resource and the Use permission.
4. Assign the custom resource to the page. To do this:
   a. Use the finder feature of the Portal to select the page. Note that clicking on the name of the page takes you directly to that page; click inside the box (but not on the name itself) to display the page's action pane.
   b. At the very bottom of the page's action pane, click **Assign**. This displays the **Assign Custom Resource** dialog.
   c. In that dialog, select the desired resource from the **Custom Resource Name** list and click **OK**.

3.6.2 Removing a Custom Resource from a Page

To disassociate a custom resource from a page, the procedure is:

1. Log in as a user who holds the **%Admin_Secure:Use** privilege or is a member of the **%All** role.
2. Use the finder feature of the Portal to select the page. Note that clicking on the name of the page takes you directly to that page; click inside the box (but not on the name itself) to display the page's action pane.
3. At the very bottom of the page's action pane, click **Assign**. This displays the **Assign Custom Resource** dialog.
4. In that dialog, select the empty item from the **Custom Resource Name** list and click **OK**.

4

Privileges and Permissions

Permissions allow you to perform some action, such as reading or writing data, or using a tool. Permissions are associated with resources, forming privileges. This model allows a user to perform a particular action in relation to a particular resource.

4.1 How Privileges Work

A privilege associates a resource with a permission, so that a role that holds the privilege can perform a particular action, such as read or write to a database or use an application. The possible permissions are:

- Read — View (but not change) the contents of a resource, such as in a database
- Write — View or change the contents of a resource, such as in a database
- Use — Run or otherwise employ an executable program or tool, such as an application or a Caché service

The meaning of each permission depends on the resource with which it is used. Permission names can appear as the full word or the first letter of the word; their names are not case-sensitive.

4.2 Public Permissions

For each resource, permissions can be designated as Public. Effectively, this is equivalent to all users holding that permission on the resource. For example, if the **`%Service_CacheDirect:Use`** privilege is Public, then any user can connect to Caché using the Caché Direct technology. Similarly, if the **`%DB_SALES:Read`** privilege is Public, then any user can read from any database protected by the %DB_SALES resource. This does not, however, enable all users to write those databases because (in this example) the **`%DB_SALES:Write`** privilege is not Public.

The following database privileges are Public by default:

Table 4–1: Default Public Privileges

Resource	Permission
%DB_CACHE	Read
%DB_CACHELIB	Read
%DB_CACHETEMP	Read, Write
%DB_DOCBOOK	Read

4.3 Checking Privileges

Caché provides a method, **$SYSTEM.Security.Check**, to check on privileges held by the current process. Its one-argument form lists what privileges the process holds on a particular resource; its two-argument form returns whether or not the process holds privileges for a particular resource.

The one-argument form returns a comma-delimited list of the permissions held by the process on a resource. For example:

```
$SYSTEM.Security.Check("%DB_Samples")
```

returns `READ,WRITE` if the process holds Read and Write permissions for %DB_Samples. The permission names are always returned as full words in uppercase letters. The function returns an empty string if the process holds no permissions on the resource.

The two-argument form returns True or False (1 or 0) to indicate whether the process holds a specific privilege. For example:

```
$SYSTEM.Security.Check("%DB_Samples", "WRITE")
```

returns 1 if the process holds the Write permission on the %DB_Samples resource.

You can also call the function with a list of permissions, such as:

```
$SYSTEM.Security.Check("%DB_Samples", "WRITE,READ")
```

It returns 1 if the process holds all of the requested permissions and 0 otherwise. You can also simply use the first letter of the privileges to be checked:

```
$SYSTEM.Security.Check("%DB_Samples", "W,R")
```

The method has the following general behaviors:

- The method always returns 1 for a public resource privilege, whether or not the process explicitly holds that privilege.
- Permission names are not case-sensitive.
- Permission names can be fully spelled out, as in the example above, or abbreviated by their first character. Also, permission names are not case-sensitive. Thus, “WRITE,READ”, “W,R”, and “R,Write” are equivalent.

4.4 When Changes in Privileges Take Effect

Caché maintains a persistent database of the security settings. When Caché starts, it extracts this information and places it into a segment of shared memory that allows quick access to the consolidated settings. While a process is executing, it maintains its own per-process cache of the privileges it has been granted. This is updated as new privileges are needed (and authorized).

Editing roles, privileges, and so on makes changes to the persistent copy of the information. This becomes visible to users or applications the next time they are subsequently authenticated.

5

Roles

A role is a named collection of privileges. Roles are useful because multiple users often need the same set of privileges. For example, all users of an application or all developers working on a particular project might need a common set of privileges. By using a role, such sets of privileges can be defined once (which makes future modification much easier) and shared by the relevant users. Privileges are assigned exclusively to roles; privileges are not assigned directly to users. To assign some privileges to a single user, create a role for that purpose.

Major topics related to roles include:

- About roles
- Roles, users, members, and assignments
- Creating roles
- Managing roles
- Predefined roles
- Login roles and added roles
- Programmatically managing roles

Note: For SQL access to data in tables, Caché supports row-level security. For information on setting this up, see the section "Adding Row-Level Security" in the chapter "Other Options for Persistent Classes" in the book *Using Caché Objects*.

5.1 About Roles

Every role has the following properties:

Table 5–1: Role Properties

Property Name	Property Description
Name	Unique role identifier. See the section "Naming Conventions" for more information on valid names.
Description	Any text.
Privileges	Resource-permission pair(s) associated with the role. A role can hold zero or more privileges.
Members	Users or roles that have been assigned to the role (listed on the **Members** tab of the **Edit Role** page).

These are displayed on the **General** tab of the **Edit Role** page, which is accessible by selecting Edit in the row for any role in the table on the Roles page (**System Administration** > **Security** > **Roles**).

Each role also may have members that are assigned to it or other roles to which it is assigned. These relations are described in the next section.

5.2 Roles, Users, Members, and Assignments

A role is a container that holds one or more privileges. If a user is associated with a role, then that user is able to exercise the role's privileges. The terminology for the association of a user and role is:

- The user *is assigned to* the role.
- The user *is a member of* the role.
- The role *includes* the user.

These phrases are all equivalent in meaning to each other.

Each user can be assigned to multiple roles and each role can have multiple users as its members. Similarly, each role can also be assigned to multiple roles and can also have multiple roles as its members. A role can have both users and roles as its members.

Suppose one role is assigned to another role. In this case, if role A is assigned to role B, then role A is described as a "member" of role B; this is equivalent to saying that role A is assigned to role B or that role B includes role A.

If one role is assigned to another, that first role holds the privileges associated with the second role. This is analogous to the relationship of assigning a user to role, whereby the user then holds the privileges associated with the role. Hence, if a user is a member of one role and that role is a member of another role, then the user holds privileges associated with both the roles.

For example, suppose a university has three roles available for its students: **UndergraduateStudent**, **GraduateStudent**, and **GeneralStudent**. Each student is assigned to either **UndergraduateStudent** or **GraduateStudent**, and these two roles are both assigned to **GeneralStudent**. If Elizabeth is assigned to **GraduateStudent**, she holds the privileges associated with both **GraduateStudent** and **GeneralStudent**; if James is assigned to **UndergraduateStudent**, he holds the privileges associated with both **UndergraduateStudent** and **GeneralStudent**.

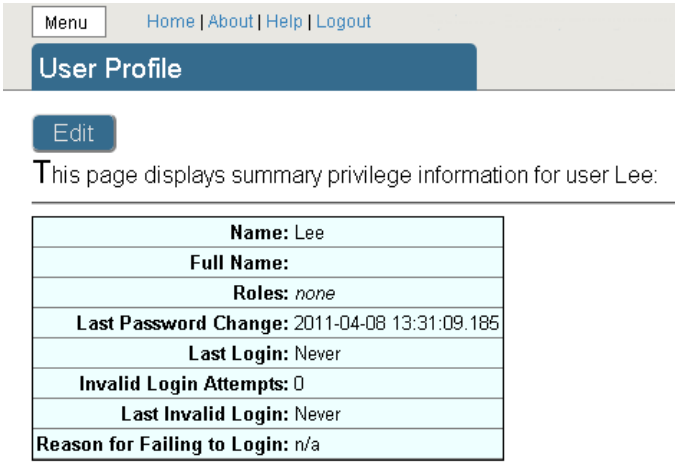
A role's members are listed on the **Edit Role** page's **Members** tab; on this tab, you can also assign new members to a role. If a role has been assigned to other roles, these are listed on the **Assigned To** tab of the **Edit Role** page; you can also assign a role to additional roles on that tab.

5.2.1 An Example of Multiple Role Assignment

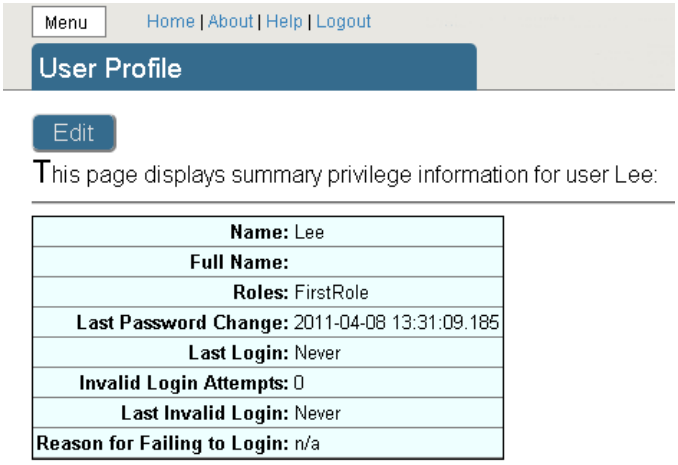
This section provides an example of how users and roles interact in Caché.

Suppose there is a user named Lee, a role named **FirstRole**, and a role named **SecondRole**. **FirstRole** protects a resource called **FirstResource** and **SecondRole** protects a resource called **SecondResource**.

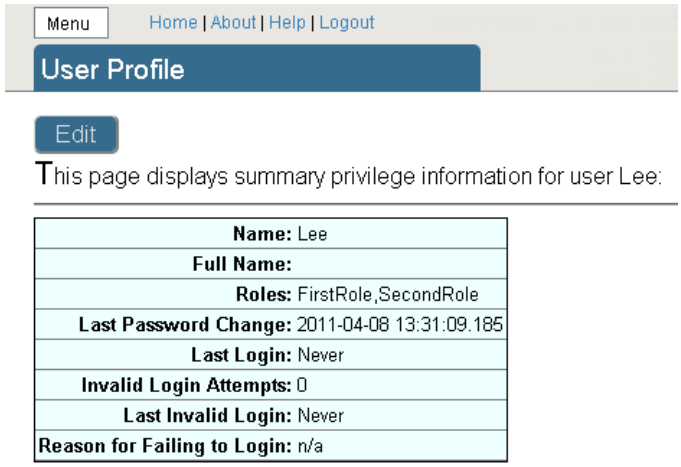
When first created, Lee is not a member of any roles. This is reflected in Lee's profile:



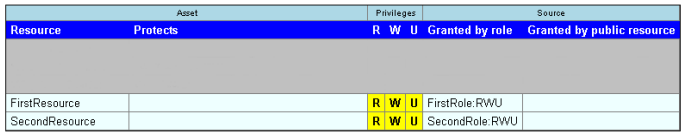
When Lee is assigned to the role **FirstRole**, this changes Lee's profile:



When the role FirstRole is assigned to the role **SecondRole**, this also changes Lee's profile:



The list of Lee's privileges specifies which privileges originate with which roles:



5.3 Creating Roles

You can define roles for use by developers, operators, system managers and other classes of users. Once created, there are various features available to edit a role.

To create a new role:

1. From the Management Portal home page, go to the **Roles** page (**System Administration** > **Security** > **Roles**).
2. On the **Roles** page, click **Create New Role**. This displays the **Edit Role** page.
3. On the **Edit Role** page, the **General** tab is visible. Here, enter values for the following properties:
   - **Name** (required) — Specifies the name of the new role. See the following section, Naming Conventions, for naming rules.
   - **Description** (optional) — Specifies descriptive information about the role.

   The role's resources are listed, but empty, as a role cannot receive resources until it has been created, except under the conditions described in the next step.
4. As a shortcut, if you wish to create multiple roles with similar characteristics, you can use the **Copy from** field on the **Role** page to begin the process of creating a new role. When you select an existing role from this field's drop-down menu, all its privileges appear in the list of resources; you can then add or delete privileges as desired, and modify its Description property.
5. Click **Save** to create the role.

Once a role exists, you can edit it as described in the section "Managing Roles." For example, the **Resources** table allows you to add new privileges to the role; do this by clicking **Add**.

Note: InterSystems recommends that you do not modify predefined roles.

5.3.1 Naming Conventions

The name of a user-defined role is subject to the following rules in its use of characters:

- It can include all alphanumeric characters.
- It can include symbols, except for the following prohibited characters: “,” (comma), “:” (colon), and “/” (slash).
- It cannot begin with “%” (the percent-sign character), which is reserved for Caché predefined roles.
- It can include Unicode characters.

Also, a role name is not case-sensitive. This means that:

- For names that are defined using mixed case, the name is preserved exactly as it is entered.
- Names that differ only by case are prohibited.
- When a name is looked up, Caché ignores differences in case.

A role name can be up to 64 characters long.

For example, if there is a role named **`BasicUser`**, any attempt to create another resource named **`BASICUSER`** will fail. References to the resource using name values of **`BasicUser`**, **`basicuser`**, and **`BASICUSER`** will all succeed.

5.4 Managing Roles

Once you have created a role, you modify it in a number of different ways, each of which is described in one of the following sections. The actions fall into several categories:

- General tasks. This includes:
  - Viewing Existing Roles
  - Deleting a Role
- Creating, modifying, and removing a role’s privileges. This includes:
  - Giving a Role New Privileges
  - Modifying Privileges for a Role
  - Removing Privileges from a Role
- Creating and removing assignments among roles and users. This includes:
  - Assigning Users or Roles to the Current Role
  - Removing Users or Roles from the Current Role
  - Assigning the Current Role to Another Role
  - Removing the Current Role from Another Role
- Modifying a Role’s SQL-related Options

Note: Changing a user’s roles or changing a role’s privileges does not affect the assigned privileges associated with the user’s existing processes. For new privileges to become active, the user must log out and log in again, restart the process, or perform an equivalent action.

5.4.1 Viewing Existing Roles

To view a list of the currently existing roles, see the **Roles** page in the Portal (**System Administration** > **Security** > **Roles**). This page displays information on the following fields:

- Name — The role's name (cannot be edited)
- Description — Any text that has been provided to describe the role
- Created By — What user created the role

For each role, you can

- Edit the role's properties, which includes all actions for privilege management, assignment management, and SQL-related options.
- Delete the role

5.4.2 Deleting a Role

To delete a role:

1. On the **Roles** page (**System Administration** > **Security** > **Roles**), for the role you wish to delete, click **Delete** in that role's row.
2. Caché displays a confirmation dialog. Click **OK** to delete the role and **Cancel** otherwise.

5.4.3 Giving New Privileges to a Role

To give a role new privileges:

1. On the **Edit Role** page (**System Administration** > **Security** > **Roles** > **Edit Role**) for an existing role, click **Add** in the **Privileges** table.
2. This displays a page listing all resources. To select a resource to assign to the role, click it. You can select multiple resources simultaneously using the **Ctrl** or **Shift** keys.
3. To add the selected resources to the role, click **Save.** This gives the role all possible permissions on the resource; you can then modify the available permissions for the resource (such as changing permissions on a database from Read-Write to just Read).

5.4.4 Modifying Privileges for a Role

To modify the privileges that a role holds:

1. From the Management Portal home page, go to the **Roles** page (**System Administration** > **Security** > **Roles**).
2. On the **Roles** page, click **Edit** for the role you wish to modify. This displays the **Edit Role** page.
3. On the **Edit Role** page, in the **Resources** table, click **Edit** for the resource whose privileges you wish to modify.
4. This displays the page for editing the permissions on the selected resource. Check or clear the boxes for each permission as appropriate.

   Note: This page does not let you clear all permissions for an individual resource. This is because eliminating all a role's permissions for a resource is equivalent to deleting the role's privileges for the resource.

5. Click **Save** to save the privileges in their new state.

These changes should be reflected in the **Resources** table on the **Role** page.

5.4.5 Removing Privileges from a Role

To remove privileges from a role:

1. From the Management Portal home page, go to the **Roles** page (**System Administration** > **Security** > **Roles**).
2. On the **Roles** page, click **Edit** for the role you wish to modify. This displays the **Edit Role** page.
3. On the **Edit Role** page, in the **Resources** table, click **Delete**. This removes the privileges for the resource from the role.
4. Click **Save** to save the privileges in their new state.

5.4.6 Assigning Users or Roles to the Current Role

A role can have users or other roles as members that are assigned to it. If a user is assigned to a role, then that user holds the privileges associated with that role. If one role is assigned to another role, then a user assigned to the first role holds the privileges associated with both roles.

The role being edited is known as the "current" role. The users and roles that are assigned to the current role are listed on the **Members** tab of the **Edit Role** page (these users and roles are known as its *members*).

To assign a user or role to the current role, the procedure is:

1. From the Management Portal home page, go to the **Roles** page (**System Administration** > **Security** > **Roles**).
2. On the **Roles** page, click **Edit** for the role you wish to modify. This displays the **Edit Role** page.
3. On the **Edit Role** page, select the **Members** tab.
4. On the **Members** tab, choose either the Users or Roles option to specify whether to assign users or roles to the role. (Users is the default.)
5. The list of users or roles that can be assigned to the current role appears in the **Available** list. You can move them to and from the **Selected** list using the arrow buttons between the lists.
6. When you have the final list of users or roles to add, click **Assign** or **Assign with Grant Option**. Clicking **Assign** simply assigns the new members (users or roles) to the role being edited. Clicking **Assign with Grant Option** also gives the new members the ability to assign other users or roles to the current role when using SQL commands.

5.4.7 Removing Users or Roles from the Current Role

If a user or role has been assigned to the current role, it is known as a member of that role. The procedure to remove a member from a role is:

1. From the Management Portal home page, go to the **Roles** page (**System Administration** > **Security** > **Roles**).
2. On the **Roles** page, click **Edit** for the role you wish to modify. This displays the **Edit Role** page.
3. On the **Edit Role** page, select the **Members** tab.
4. On the **Members** tab, there is a table of users and roles assigned to the current role. For the specified members, click the  button in the right-most column of the member's row.
5. A prompt appears to confirm the removal. Click **OK**.

The user or role has now been removed from the current role.

5.4.8 Assigning the Current Role to Another Role

One role can be assigned to another role. If one role is assigned to another role, then a user assigned to the first role holds the privileges associated with both roles.

The role being edited is known as the "current" role. The roles to which the current is assigned are listed on the **Assigned To** tab of the **Edit Role** page.

To assign the current role to another role, the procedure is:

1. From the Management Portal home page, go to the **Roles** page (**System Administration** > **Security** > **Roles**).
2. On the **Roles** page, click **Edit** for the role you wish to modify. This displays the **Edit Role** page.
3. On the **Edit Role** page, select the **Assigned To** tab.
4. The list of roles that the current role can be assigned to appears in the **Available** list. You can move them to and from the **Selected** list using the arrow buttons between the lists.
5. When you have the final list of roles, click **Assign** or **Assign with Grant Option**. Clicking **Assign** simply assigns the current role to the selected roles. Clicking **Assign with Grant Option** also gives the current role the ability to assign other users or roles to the selected role(s) when using SQL commands.

5.4.9 Removing the Current Role from Another Role

If the current role has been assigned to another role, it is known as a member of that role. The procedure to remove the current role from another role is:

1. From the Management Portal home page, go to the **Roles** page (**System Administration** > **Security** > **Roles**).
2. On the **Roles** page, click **Edit** for the role you wish to modify. This displays the **Edit Role** page.
3. On the **Edit Role** page, select the **Assigned To** tab.
4. On the **Assigned To** tab, there is a table of roles to which the current role is assigned. To remove the current role from one of these roles, select the  button in the right-most column of that role's row.
5. A prompt appears to confirm the removal. Click **OK**.

The current role has now been removed from that role.

5.4.10 Modifying a Role's SQL-Related Options

For every role, you can grant or remove the following SQL-related characteristics:

- General SQL Privileges
- Privileges for Tables
- Privileges on Views
- Privileges for Stored Procedures

5.4.10.1 General SQL Privileges

On the **SQL Privileges** tab of the **Edit Role** page, you can add or remove SQL privileges for a role:

- To add a privilege to a role, first move the privilege from the **Available** list to the **Selected** list (either double-click it or select it and then click the single right-arrow); click **Assign** to give the privilege to the role. To also add the privilege of being able to grant the added privilege to other roles, select the relevant check box below the **Available** list.
- To add all privileges to a role, click the double-arrow pointing from the **Available** list to the **Selected** list; click **Assign** to give the privileges to the role. To also add the privileges of being able to grant the added privileges to other roles, select the relevant check box below the **Available** list.
- To remove a privilege from a role, click **Remove** to the right of privilege name.
- To remove all privileges from a role, click **Remove All** below the table listing the currently assigned privileges.

The following privileges are available:

- %ALTER_TABLE — For a given namespace, allow the member of the role to run the ALTER TABLE command.
- %ALTER_VIEW — For a given namespace, allow the member of the role to run the ALTER VIEW command.
- %CREATE_FUNCTION — For a given namespace, allow the member of the role to run the CREATE FUNCTION command.
- %CREATE_METHOD — For a given namespace, allow the member of the role to run the CREATE METHOD command.
- %CREATE_PROCEDURE — For a given namespace, allow the member of the role to run the CREATE PROCEDURE command.
- %CREATE_QUERY — For a given namespace, allow the member of the role to run the CREATE QUERY command.
- %CREATE_TABLE — For a given namespace, allow the member of the role to run the CREATE TABLE command.
- %CREATE_TRIGGER — For a given namespace, allow the member of the role to run the CREATE TRIGGER command.
- %CREATE_VIEW — For a given namespace, allow the member of the role to run the CREATE VIEW command.
- %DROP_FUNCTION — For a given namespace, allow the member of the role to run the DROP FUNCTION command.
- %DROP_METHOD — For a given namespace, allow the member of the role to run the DROP METHOD command.
- %DROP_PROCEDURE — For a given namespace, allow the member of the role to run the DROP PROCEDURE command.
- %DROP_QUERY — For a given namespace, allow the member of the role to run the DROP QUERY command.
- %DROP_TABLE — For a given namespace, allow the member of the role to run the DROP TABLE command.
- %DROP_TRIGGER — For a given namespace, allow the member of the role to run the DROP TRIGGER command.
- %DROP_VIEW — For a given namespace, allow the member of the role to run the DROP VIEWcommand.

5.4.10.2 Privileges for Tables

On the **SQL Tables** tab of the **Edit Role** page, you can add or remove table-related SQL privileges for a role:

1. Choose the relevant namespace from the drop-down near the top of the page. A list of the namespace's tables appears.
2. To change privileges for a table, click **Edit** in that table's row. This displays a window for altering privileges.
3. In this window, you can check or clear any of the following items:
   - ALTER
   - DELETE
   - INSERT

- REFERENCES
- SELECT
- UPDATE
- Give the GRANT option to the role

4. After making your selection(s), click **Apply** to establish the new privileges for the table.

If a role has privileges for a table, it is listed in a table on this page. To revoke the role's privileges for a table, click **Revoke** at the far right of the role's row. Clicking this displays a message containing the namespace and the full name of the table (including the schema), such as the "SAMPLES Sample.Company" namespace and table.

5.4.10.3 Privileges on Views

On the **SQL Views** tab of the **Edit Role** page, you can add or remove view-related SQL privileges for a role.

To add privileges for the view:

1. Choose the relevant namespace from the drop-down near the top of the page. A list of the namespace's views will appear.
2. To change privileges for a view, click **Edit** in that view's row. This displays a window for altering privileges.
3. In this window, you can check or clear any of the following items:
   - ALTER
   - DELETE
   - INSERT
   - REFERENCES
   - SELECT
   - UPDATE
   - Give the GRANT option to the role
4. After making your selection(s), click **Apply** to establish the new privileges for the table.

If a role has privileges for a view, it is listed in a table on this page. To revoke the role's privileges for a view, click **Revoke** at the far right of the role's row. Clicking this displays a message containing the namespace and the full name of the view (including the schema).

5.4.10.4 Privileges for Stored Procedures

On the **SQL Procedures** tab of the **Edit Role** page, you can add or remove a role's SQL privileges related to stored procedures.

To add privileges for a stored procedure:

1. Choose the relevant namespace from the drop-down near the top of the page. A list of the namespace's stored procedures then appears.
2. Below this window, click **Add**, which displays the **Grant procedure privilege...** dialog.
3. In this dialog, near the top, select the schema from the drop-down that contains the procedure that you wish to add. This displays a list of the schema's procedures in the **Available** window on the left part of the page.
4. Move one or more procedures into the **Selected** window. Make sure the **EXECUTE** box is checked, so that the role has the privilege to execute the stored procedure.

5. Optionally, you can grant the roles the ability to grant this privilege on other roles; to do this, click the **Grant privilege** box near the bottom of the page.
6. Click **Apply** to grant the privilege(s) to the role.

If a role has privileges for a stored procedure, it is listed in a table on this page. To revoke the role's privileges for a stored procedure, click **Revoke** at the far right of the role's row. Clicking this displays a message containing the namespace and the full name of the stored procedure (including the schema).

5.5 Predefined Roles

Caché includes a number of predefined roles. These include:

- **%All** — The ability to perform all operations.
- **%Developer** — The privileges typically associated with application development. These are roughly the privileges associated with the Portal's System Exploration menu. They include the ability to use the WebStress and UnitTest pages in the Management Portal, as well as the documentation class reference (sometimes known as Documatic).
- **%Manager** — The privileges typically associated with system management. These are roughly the privileges associated with the Portal's System Administration and System Operation menus.
- **%Operator** — The privileges typically associated with system operation. These are roughly the privileges associated with the Portal's System Operation menu.
- **%SQL** — The privileges typically associated with SQL-related tasks.
- **%SecureBreak** — The **%Secure_Break:Use** privilege, which enforces use of the secure debug shell. For more information on the secure debug shell, see the "The Secure Shell" section in the "System Management and Security" chapter.

Note: InterSystems recommends that you do not modify predefined roles. Rather, create a new role based on the predefined role and modify the role that you have created.

The following table has a column for each role. Each row of the table lists a system-defined resource and the privilege, if any, that the role holds on it.

Table 5–2: Predefined Roles and Their Privileges

Resource	%Developer	%Manager	%Operator	%SQL	%SecureBreak
%Admin_Manage		Use			
%Admin_Operate		Use	Use		
%Admin_Secure		Use			
%Admin_Task		Use			
%DB_CACHE	Read	Read	Read		
%DB_CACHEAUDIT		Read			
%DB_CACHELIB	Read	Read, Write			
%DB_CACHESYS		Read, Write	Read, Write		
%DB_CACHETEMP	Read, Write	Read, Write	Read, Write		

Resource	%Developer	%Manager	%Operator	%SQL	%SecureBreak
%DB_DOCBOOK	Read	Read, Write	Read		
%DB_SAMPLES	Read, Write	Read, Write			
%DB_USER	Read, Write	Read, Write			
%DB_%DEFAULT	Read, Write	Read, Write			
%Development	Use	Use			
%Secure_Break					Use
%Service_Console					
%Service_CSP	Use	Use	Use		
%Service_Object	Use	Use			
%Service_SQL	Use	Use		Use	
%Service_Telnet	Use	Use			
%Service_Terminal	Use	Use			

The definitions of these predefined roles are set during a new Caché installation and are not modified during an upgrade installation. With the exception of **%All**, the use of predefined roles is optional.

The **%Admin_Secure** resource is designed to make all the necessary security assets available or restricted as a single unit. This makes it easy to separate these resources for use by the security administrator.

Note: The **%Operator** role does not hold the **%Admin_Task:Use** privilege by default; if you wish for members of that role to be able to manage tasks, include **%Admin_Task:Use** among the role's privileges. Further, any custom roles based on **%Operator** must add the **%DB_CACHESYS:RW** privilege in order to use the Portal's **Operator** menu. They may also add the **%Admin_Task:Use** privilege so that they can manage tasks.

5.5.1 %All

The predefined role, **%All**, always holds all privileges for all resources on the system. This is why, for example, a user belonging to the **%All** role can still mount a database for which there is no resource available. (One exception is the restrictive **%Secure_Break:Use** privilege, which must always be explicitly granted.)

This role cannot be deleted or modified, and there must always be at least one user account holding the **%All** role. If there is only one such account, it cannot be deleted or disabled. This is designed to protect a sole Caché system administrator from being inadvertently locked out of the system.

Important: A user who is assigned to the **%All** role does not automatically have access to rows in a table that are protected with row-level security. The application must explicitly provide the **%All** role with access to such a row. For detailed information about how to do this, see the section "Adding Row-Level Security" in the chapter "Other Options for Persistent Classes" in the book *Using Caché Objects*.

5.5.2 Default Database Resource Roles

When you create a database resource, the system automatically creates a role with the name **%DB_<database-resource-name>** that has Read and Write permissions for that resource.

5.6 Login Roles and Added Roles

Each Caché process has, at any point in time, a set of roles that determine the current privileges for that process. The set of roles includes both *login roles*, which come from the definition of the user (received at login time) and *added roles*, which come from the currently running application (received by application role escalation). From a security standpoint, the origin of a role is immaterial: a process either has a required privilege or it does not.

When an application is started, each role currently held by the process is looked up in a table and any associated application roles are added.

For example, suppose there is an order entry application with two classes of users: normal users, who are assigned the **OrderEntryUser** role, and managers, who are assigned the **OrderEntryManager** role. Both of these roles allow someone to run the order entry application (that is, both are assigned the **%Application_OrderEntry:Use** privilege.) But, when the application does role escalation, different roles are used (**OrderEntryAppNormal** versus **OrderEntryAppSpecial** and **OrderEntryAppReporting**) to enable the application to perform different functions on behalf of these user classes.

Matching Role	Added Roles
OrderEntryUser	**OrderEntryAppNormal**
OrderEntryManager	**OrderEntryAppSpecial**, **OrderEntryAppReporting**

During the matching sequence, each role held by the process is considered, even if a match has already been found. In other words, multiple roles may match and multiple sets of new roles may be added. However, this process is not recursive: roles added as a result of the matching process are not considered for further matches.

Note: There is no way to restrict a user's roles to fewer than the login roles.

5.6.1 A Note on Added Roles and Access in the Management Portal

When a user goes to a new Portal page, the Portal resets the process to have only the user's login roles. The Portal then checks if the page's application requires a resource; if it does, then the Portal checks if the user has the appropriate permissions on that resource. If the user's privileges do not include the required privileges, the page will not be available.

If the user does have the required privileges, the Portal then adds any application roles and any applicable target roles. The Portal then checks if any links on the page require custom resources; if the user has the appropriate resource(s), the Portal displays those links.

5.7 Programmatically Managing Roles

Certain routines can directly modify the application roles of a running process by setting the *$ROLES* system variable, such as

```
SET $ROLES = "Payroll"
```

$ROLES contains a comma-separated list of the role names assigned to the current process. The union of all the privileges granted to all roles in the list determines the privileges that the process possesses. *$ROLES* initially contains the roles assigned at authentication (that is, login roles).

This command can only be invoked either from a routine that is part of the CACHESYS database or if the current privileges held include Write permission for the CACHESYS database (**%DB_CACHESYS:W**).

Note that setting *$ROLES* only alters a process's added roles, not its login roles. Thus if a process currently has the login roles Employee and Manager and the added role **Payroll**, after the statement

```
SET $ROLES = "Accounting"
```

$ROLES has the value "Employee,Manager,Payroll,Accounting".

A role can be added to the process's current roles by concatenating it to the current roles, with a call such as:

```
SET $ROLES = $ROLES _ ",Payroll"
```

The statement

```
SET $ROLES = ""
```

removes all added roles.

The **NEW** command can be used with *$ROLES* to stack the current set of roles (Login and Added) and the current value of *$USERNAME*. This allows code to modify the list and, whether control leaves the containing block normally or abnormally, the changes are undone upon exit.

With the exception of a null string argument, `SET $ROLES = <role_name>` is a system capability. `NEW $ROLES` and `SET $ROLES = ""` can be executed by any code.

6

Users

This chapter includes the following sections:

6.1 Properties of Users

Each Caché user account has a number of properties. The following are listed on the **General** tab for the user:

Table 6–1: User Account Properties

Property Name	Property Description
Name	Unique user identifier that is up to 128 characters long. This can include any character except "@" or "*". A name is not case-sensitive. All usernames can include Unicode characters.
Full Name	The user's displayable name.
Comment	Any text.
Password	New password value. This value is never visible, regardless of the privileges of user viewing this page; a user either with the **`%Admin_Secure:Use`** privilege or assigned to the **`%All`** role can change another user's password, such as if that user's password has been forgotten or lost. A password can be up to 128 characters long and is case-sensitive. All paswords can include Unicode characters.
Confirm Password	Confirmation of new password value.
Change Password on Next Login	A check box specifying whether or not the user is required to change the password at the next login.

Property Name	Property Description
Password Never Expires	A check box specifying whether or not the system-wide password expiration limit applies to this user. If selected, the user's password does not expire, even if it has been unchanged for longer than the system limit. To set the password expiration limit, see the System-wide Security Parameters page.
User Enabled	A check box specifying whether or not the account is currently enabled.
Expiration Date	The last date on which the account can be used.
Account Never Expires	A check box specifying whether or not the system-wide account inactivity limit applies to this user. If selected, the user's account does not expire, even if it has been inactive for longer than the system limit. To set the inactivity limit, see the System-wide Security Parameters page.
Startup Namespace	The namespace in which to begin execution following login from a terminal-type service. This property overrides any namespace value provided via the command invoking Caché.
Startup Tag^Routine	The routine to execute automatically following login from a terminal-type service. This property overrides any routine value provided via the command invoking Caché.
Mobile Phone Service Provider	For two-factor authentication, the user's mobile phone service provider.
Mobile Phone Number	For two-factor authentication, the mobile phone number at which the user receives a text message containing the second authentication token (factor).
Type (only displayed on the **Users** page)	The kind of user, which is determined by the authentication and role-assignment mechanisms in use. Values can be `Caché password user`, `Delegated user`, `Kerberos user`, `LDAP user`, or `OS user`. For more information on user types, see the following section, "About User Types."

6.1.1 About User Types

Among a user's properties is the user's *Type*. There are multiple possible types:

- Caché password user — This type is authenticated through Caché login, Kerberos (without delegated authorization), or the operating system (without delegated authorization). The Caché tools for editing or otherwise altering users are for use with Caché password users.
- Delegated user — This type is authenticated through a user-defined authentication mechanism. The Caché tools are available only for viewing this type of user's properties; the user's properties must be edited through external means.
- Kerberos user — This type is authenticated using Kerberos when delegated authorization is in use; with delegated authorization, Caché tools are available only for viewing this type of user's properties; the user's properties must be edited through external means and specified by the **ZAUTHORIZE** routine, as described in the chapter "Delegated Authorization." If a user is authenticated through Kerberos without using delegated authorization, then the user is of type Caché Password User.
- LDAP user — This type is authenticated through LDAP. The Caché tools are available only for viewing this type of user's properties; the user's properties must be edited through external means.
- OS user — This type is authenticated through the operating system (OS) when delegated authorization is in use; with delegated authorization, Caché tools are available only for viewing this type of user's properties; the user's properties

must be edited through external means and specified by the **ZAUTHORIZE** routine, as described in the chapter "Delegated Authorization." If a user is authenticated through the operating system without using delegated authorization, then the user is of type Caché Password User.

Important: A user can only have one type. A user of one type cannot log in using authentication mechanisms associated with another type.

For more information about the relationship among user types, authentication, and role assignment, see the "Authentication-Authorization Matrix" in the "Authentication" chapter.

6.2 Creating and Editing Users

To either create or edit a user, the Management Portal provides the **Edit User** page, which is off the **Users** page (**System Administration** > **Security** > **Users**) for each user being created or edited. This page differs for creating and editing users as follows:

- When creating a new user, the Name field accepts a value.
- When editing an existing user, the Name field is not writable.

6.2.1 Creating a New User

To create a new user:

1. From the Management Portal home page, go to the **Users** page (**System Administration** > **Security** > **Users**).
2. On the **Users** page, select **Create New User**. This displays the **General** tab of the **Edit User** page for creating and configuring users.
3. On the **Edit User** page, set values for the following user properties, which are described in the Properties of Users section:
   - **Name** (required) — Unique user identifier.
   - **Copy from** (optional) — The name of another user, the settings of which are to serve as the basis for the new user. See the next step for details.
   - **Full Name** (optional) — The account's displayable name.
   - **Comment** (optional) — Any text.
   - **Password** (required) — New password value.
   - **Confirm Password** (required) — Confirmation of new password value.
   - **Change Password on Next Login** (optional) — Whether or not a password change is required at the next login.
   - **Password Never Expires** (optional) — Whether or not the user's password is valid past the system-wide password expiration limit. For information about the password expiration limit, see the System-wide Security Parameters page.
   - **User Enabled** (required) — Whether or not the account is available for use.
   - **Expiration Date** (optional) — The last date on which the account is available for use.
   - **Account Never Expires** (optional) — Whether or not the user's account remains active past the system-wide inactivity limit. For information about the inactivity limit, see the System-wide Security Parameters page.

- **Startup Namespace** (optional) — The namespace in which to begin execution following login from a terminal-type service.
- **Startup Tag^Routine** (optional) — The routine to execute automatically following login from a terminal-type service.
- **Mobile phone service provider** — For two-factor authentication, the user's mobile phone service provider. (If the user's mobile phone service provide does not appear in the list, you can add a new provide by clicking **Create new provider**; this displays fields for adding a new mobile phone service provider, which will then be visible.
- **Mobile phone number** — For two-factor authentication, the mobile phone number at which the user receives the text message containing the second authentication token (factor).

4. As a shortcut, if you wish to create multiple users with similar characteristics, you can use the **Copy from** field on the **Edit User** screen to begin the process of creating a new user. When you select an existing user from this field's drop-down menu, the following properties of the selected user receive values for the user being created:
   - Full Name
   - Expiration Date
   - Default Namespace
   - Default Tag^Routine
5. Click the **Save** button to create the new user.

Once you have created a user account, you can then edit its characteristics.

6.2.2 Editing an Existing User

Once you have created a user account, you can alter any of its basic properties (on the **General** tab of the **Edit User** page):

1. From the Management Portal home page, go to the **Users** page (**System Administration** > **Security** > **Users**).
2. On the **Users** page, there is a table of users. To edit an existing user, select the name of the user from the table. This displays the **General** tab of the **Edit User** page for creating and configuring users.
3. On the **Edit User** page, you can alter values for the following properties, which are described in the Properties of Users section:
   - **Full Name**
   - **Comment**
   - **Password** — If you use this page to set a new password, the password must conform to the pattern (types of characters and length) specified in the **Password Pattern** field on the **System Security Settings** page (**System Administration** > **Security** > **System Security** > **System-wide Security Parameters**).
   - **Change password on next login**
   - **Password Never Expires**
   - **User Enabled**
   - **Expiration Date**
   - **Account Never Expires**
   - **Startup Namespace** — This refers to the Terminal Namespace property.
   - **Startup Tag^Routine** — This refers to the Terminal Routine property.
   - **Mobile phone service provider**

- **Mobile phone number**

4. Click the **Save** button to save the new values for the user.

You can also modify other aspects of the user account on this page's other tabs:

- Roles — Lists the roles that the user currently holds. You can also give a user new roles (or take them away) on this page.
- SQL Properties — This includes:
  - **SQL Privileges** — Lists all the SQL privileges that a user currently holds, on a per-namespace basis. You can also assign or revoke SQL privileges on this page.
  - **SQL Tables** — Lists, by namespace, the tables for which a user has been granted privileges (%ALTER, DELETE, INSERT, REFERENCES, SELECT, and UPDATE). You can also assign or revoke SQL table privileges on this page.
  - **SQL Views** — Lists, by namespace, the views for which a user has been granted privileges (%ALTER, DELETE, INSERT, REFERENCES, SELECT, and UPDATE). You can also assign or revoke SQL view privileges on this page.
  - **SQL Procedures** — Lists, by namespace, the stored procedures which a user can run. You can also assign or revoke the right to run procedures on this page.

Note: A change to a user account only takes effect after the user logs out and then logs back in.

6.2.2.1 Modifying a User's Roles

On the **Roles** tab of the **Edit User** page, you can assign a user to a role or remove it from a role:

- To assign a user to a role, first move the role from the **Available** list to the **Selected** list (either double-click it or select it and then click the single right-arrow); click the **Assign** button to assign the user to the role.
- To assign a user to all roles, click the double-arrow pointing from the **Available** list to the **Selected** list; click the **Assign** button to assign the user to all the roles.

  Note: If you assign a user to all roles, this includes the predefined %SecureBreak role, which limits (and does not expand) the user's abilities. If a user is assigned to the **`%SecureBreak`** role, this enables the Caché secure debug shell, which restricts the commands that the user may issue. This may also have unexpected consequences in other areas.

- To remove a user from a role, click the  button to the right of role name.
- To remove a user from all roles, click **Remove All** below the table listing the currently assigned roles. (This button is only present if a user is assigned to two or more roles.)

6.2.2.2 Modifying a User's SQL-Related Options

For every user, you can grant or remove the following SQL-related characteristics:

- General privileges
- Privileges for tables
- Privileges on views
- Privileges for stored procedures

General SQL Privileges

On the **SQL Privileges** tab of the **Edit User** page, you can add or remove SQL privilege for a user:

- To add a privilege to a user, first move the privilege from the **Available** list to the **Selected** list (either double-click it or select it and then click the single right-arrow); click the **Assign** button to give the privilege to the user. To also add the privilege of being able to grant the added privilege to other users, click the relevant button below the **Available** list.
- To add all privileges to a user, click the double-arrow pointing from the **Available** list to the **Selected** list; click the **Assign** button to give the privileges to the user. To also add the privileges of being able to grant the added privileges to other users, click the relevant button below the **Available** list.
- To remove a privilege from a user, click the **Remove** link to the right of privilege name.
- To remove all privileges from a user, click the **Remove All** button below the table listing the currently assigned privileges.

The following privileges are available:

- %ALTER _TABLE — For a given namespace, allow the user to run the ALTER TABLE command.
- %ALTER_VIEW — For a given namespace, allow the user to run the ALTER VIEW command.
- %CREATE_FUNCTION — For a given namespace, allow the user to run the CREATE FUNCTION command.
- %CREATE_METHOD — For a given namespace, allow the user to run the CREATE METHOD command.
- %CREATE_PROCEDURE — For a given namespace, allow the user to run the CREATE PROCEDURE command.
- %CREATE_QUERY — For a given namespace, allow the user to run the CREATE QUERY command.
- %CREATE_TABLE — For a given namespace, allow the user to run the CREATE TABLE command.
- %CREATE_TRIGGER — For a given namespace, allow the user to run the CREATE TRIGGER command.
- %CREATE_VIEW — For a given namespace, allow the user to run the CREATE VIEW command.
- %DROP_FUNCTION — For a given namespace, allow the user to run the DROP FUNCTION command.
- %DROP_METHOD — For a given namespace, allow the user to run the DROP METHOD command.
- %DROP_PROCEDURE — For a given namespace, allow the user to run the DROP PROCEDURE command.
- %DROP_QUERY — For a given namespace, allow the user to run the DROP QUERY command.
- %DROP_TABLE — For a given namespace, allow the user to run the DROP TABLE command.
- %DROP_TRIGGER — For a given namespace, allow the user to run the DROP TRIGGER command.
- %DROP_VIEW — For a given namespace, allow the user to run the DROP VIEW command.

Privileges for Tables

On the **SQL Tables** tab of the **Edit User** page, you can add or remove table-related SQL privileges for a user:

1. Choose the relevant namespace from the drop-down near the top of the page. A list of the namespace's tables will appear.
2. To change privileges for a table, select the **Edit** button in that table's row. This displays a window for altering privileges.
3. In this window, you can check or uncheck any of the following items:
   - ALTER
   - SELECT
   - INSERT
   - UPDATE

- DELETE
- REFERENCES

4. After making your selection(s), click the **Apply** button to establish the new privileges for the table.

Privileges on Views

On the **SQL Views** tab of the **Edit User** page, you can add or remove view-related SQL privileges for a user.

To add privileges for the view:

1. Choose the relevant namespace from the drop-down near the top of the page. A list of the namespace's views will appear.
2. To change privileges for a view, select the **Edit** button in that view's row. This displays a window for altering privileges.
3. In this window, you can check or uncheck any of the following items:
   - ALTER
   - SELECT
   - INSERT
   - UPDATE
   - DELETE
   - REFERENCES
4. After making your selection(s), click the **Apply** button to establish the new privileges for the table.

Privileges for Stored Procedures

On the **SQL Procedures** tab of the **Edit User** page, you can add or remove a user's SQL privileges related to stored procedures.

To add privileges for a stored procedure:

1. Choose the relevant namespace from the drop-down near the top of the page. A list of the namespace's stored procedures will appear.
2. Below this window, click the **Add** button, which displays the **Grant procedure privilege...** dialog.
3. In this dialog, near the top, select the Schema from the drop-down that contains the procedure that you wish to add. This displays a list of the schema's procedures in the **Available** window on the left part of the page.
4. Move one or more procedures into the **Selected** window. Make sure the **EXECUTE** box is checked, so that the user has the privilege to execute the stored procedure.
5. Optionally, you can grant the users the ability to grant this privilege on other users; to do this, click the **Grant privilege** box near the bottom of the page.
6. Click the **Apply** button to grant the privilege(s) to the user.

To remove a user's stored procedure privileges:

1. Choose the relevant namespace from the drop-down near the top of the page. A list of the namespace's stored procedures will appear.
2. To change privileges for a stored procedure, select the **Edit** button in that table's row. This displays a page for altering privileges.
3. On the page that appears, uncheck the **EXECUTE** check box and the **GRANT privilege** check box as appropriate.
4. Click the **Apply** button to change the privilege(s) for the user.

6.3 Viewing and Managing Existing Users

To view a list of the currently existing users, see the **Users** page in the Portal (**System Administration** > **Security** > **Users**). This page displays information on the following fields (as described in more detail in the Properties of Users section):

- User — A unique identifier for the user
- Full Name — The user's displayable name
- Enabled — Whether or not the user is currently enabled
- Namespace (default namespace) — The initial namespace for a terminal-type connection
- Routine (default routine) — The initial routine executed for a terminal-type connection
- Type — The kind of user, which is determined by the authentication and role-assignment mechanisms in use

For each user, you can

- Edit the user's properties
- Delete the user
- View the user profile

6.3.1 Deleting a User

To delete a user:

1. From the Management Portal home page, go to the **Users** page (**System Administration** > **Security** > **Users**).
2. On the **Users** page, for the user you wish to delete, select the **Delete** button in that user's row.
3. Caché displays a confirmation dialog. Select **OK** to delete the user and **Cancel** otherwise.

6.3.2 Viewing a User Profile

A user profile provides security information about a user account, such as the roles to which the user is assigned and the time of the user's last login. To view a user's profile, the procedure is:

1. From the Management Portal home page, go to the **Users** page (**System Administration** > **Security** > **Users**).
2. On the **Users** page, in the row for the user, click **Profile**. This displays the user's profile.

Alternately, if the **Edit User** page is visible for a user, click **Profile** in the upper-left corner of the page.

The following properties are listed as part of the user profile.

Table 6–2: User Profile Properties

Property Name	Property Description
Name	Unique user identifier. This can include any characters except the @, which is used to identify a domain. This is editable on the **Edit User** page.
Full Name	The user's displayable name. This is editable on the **Edit User** page.
Roles	A comma-separated list of roles assigned to user. These are editable on the **Roles** tab of the **Edit User** page.

Property Name	Property Description
Last Password Change	The date and time of user's most recent password change.
Last Login	The date and time of most recent successful login or 0 if there has not yet been a successful login. Read-only.
Last Login Device	The IP address of the host from which the user last logged in.
Invalid Login Attempts	The number of invalid login attempts since the most recent successful login. Read-only.
Last Invalid Login	The date and time of most recent invalid login attempt. Read-only.
Last Invalid Login Device	The IP address of the host from which the user last unsuccessfully attempted to log in.
Last Reason for Failing to Login	The error thrown for the most recent invalid login attempt. Read-only.
Time account was created	The date and time at which the user account was created. Read-only.
Username who created account	The account name associated with the user who created the account. Read-only.
Time account was last modified	The date and time at which the account was last modified. Read-only.
Username who last modified account	The account name associated with the user who last modified the account. Read-only.
Information last modified in account	A list of properties that were last modified for the account. Read-only.

6.4 Predefined User Accounts

Every instance of Caché automatically includes the following accounts:

Table 6–3: Predefined Users

Username	Assigned Roles	Purpose
Admin	%Manager	Default administrator account. This account exists for all instances of all InterSystems products to support instance administration. InterSystems recommends that you change the password for this account from its initial value prior to going into production.
CSPSystem	(None)	Default account representing the CSP Gateway when it connects to Caché via Caché login for Normal and Locked-down instances. InterSystems recommends that you change the password for this account from its initial value prior to going into production.
SuperUser	%All	Default account with all privileges available. This account exists for all instances of all InterSystems products to provide complete access to all aspects of the product. InterSystems recommends that you change the password for this account from its initial value prior to going into production.
UnknownUser	%All (Minimal security) or None (Normal or Locked-Down security)	Default account for a non-logged in user
_PUBLIC	(None)	Set of privileges given to all users (not a login account).
_SYSTEM	%All	Default SQL account. This account exists for all instances of all InterSystems products to provide SQL access. InterSystems recommends that you disable this account for production systems.
_Ensemble	%All	Internal Ensemble user (not a login account). Only on Ensemble instances.

There is also an account called a "privileged user account," which is created during Normal and Locked Down installations and for which you supply a username and password.

It is not possible to delete the following accounts:

- _Ensemble
- _PUBLIC
- _SYSTEM
- UnknownUser

You can delete other predefined user accounts, subject to the requirement that there be at least one user with the **%All** role.

CAUTION: If the _SYSTEM account is available with its default password of "SYS" on deployed systems, this is a security vulnerability. To address this issue, you can disable the account or change its password. InterSystems recommends disabling the account.

6.4.1 Default Predefined Account Behavior

The predefined accounts have different defaults and behavior depending on whether an installation uses Minimal security, Normal security, or Locked Down security.

6.4.1.1 Minimal Security Defaults

For an installation with Minimal security, all the created accounts except _PUBLIC have an initial default password of "SYS". With the exception of UnknownUser, you should change the account passwords after installation in order to prevent unauthorized access to your Caché instance.

The _PUBLIC account has no password by default and should never be given a password, since it is never enabled.

6.4.1.2 Normal Security Defaults

For an installation with Normal security, all the created accounts except _PUBLIC receive the same password as is chosen for the privileged user account. It is recommended that you change these passwords after installation, so that each account has its own password.

The _PUBLIC account has no password by default and should never be given a password, since it is never enabled.

6.4.1.3 Locked Down Security Defaults

For an installation with Locked Down security, all the created accounts except _PUBLIC receive the same password as is chosen for the privileged user account. It is recommended that you change these passwords after installation, so that each account has its own password.

The _PUBLIC account has no password by default and should never be given a password, since it is never enabled. In Locked-Down installations, the _SYSTEM account is also disabled.

6.4.2 Notes on Various Accounts

6.4.2.1 The UnknownUser Account

For certain applications, or certain parts of an application, unauthenticated users may have a legitimate reason to use Caché, such as for a retail system to display availability of products, prior to the user initiating a purchase. For this type of situation, Caché supports the UnknownUser account. When an unauthenticated user connects, a special name, UnknownUser, is assigned to $USERNAME and the roles defined for that user are assigned to $ROLES.

Unauthenticated access is *not* used when authentication fails. For example, suppose that a user attempts to connect to Caché via terminal and supplies a username and password that fails authentication. Here, the user is not connected to Caché, even if unauthenticated access is permitted; on the other hand, if unauthenticated access is permitted and that same user connects to Caché without supplying a username (such as by pressing the Enter key at the username prompt), then that user is connected as UnknownUser, the unauthenticated user. Similarly, if an ODBC client attempts to connect with null strings for username and password, that connection will be accepted if unauthenticated access is permitted for the service; if the same ODBC client provides a non-empty username and password values that fail authentication, that client is not connected even if unauthenticated access is permitted.

6.4.2.2 The _PUBLIC Account

The predefined user, _PUBLIC, is a special account that does not exist for logins. Rather, it holds a set of roles. These roles are specified as the default roles for any user who connects to the system. This ensures a minimum set of roles for any user. For example, if you associate the %Operator role with the _PUBLIC user, then the value of *$Roles* for any user will always include %Operator.

6.5 Validating User Accounts

If you need to validate user accounts in application code, you can do this by creating a simple routine that attempts to log the user in with the one-argument form of the **$SYSTEM.Security.Login** method. If the login succeeds, the user is valid; if the login fails, the user is not valid. When the routine exits (regardless of the login's success or failure), the current user will be the user who invoked the routine.

Here is a sample routine to perform this task, called **ValidateUser**:

```
ValidateUser(TestUser) {
    Write "Validating ",TestUser,"...",!
    New $Roles
    Set sc = $SYSTEM.Security.Login(TestUser)
    If sc = 1 {
        Write $Username," is a valid user.",!
        Write $Username," belongs to the following login roles: ",$Roles,!
    } Else {
        Write TestUser," is not a valid user.",!
    }
    Quit sc
}
```

This routine takes as its single argument, a string that is the name of the user to be validated. It then performs the following actions:

1. The call of `New $Roles` stacks both the *$Roles* variable and the *$Username* variable. For more information on *$Roles*, see the $Roles reference page.
2. It then invokes the one-argument form of the **$SYSTEM.Security.Login** method, which attempts to log the user in and does not require the user's password. If the login succeeds, the method returns 1; this determines the information that the routine displays and the routine's return value.
3. When the routine exits, this implicitly logs out the user, if there has been a successful login.

Important: This routine uses the one-argument form of the **$SYSTEM.Security.Login** method. To successfully invoke the one-argument form of **$SYSTEM.Security.Login**, a user must have the **`CACHESYS:Write`** and **`%Service_Login:Use`** privileges. For more information on **$SYSTEM.Security.Login**, see the reference page for the %SYSTEM.Security class.

Here is a sample routine that demonstrates invoking **ValidateUser**, called **VUTest**. It is hard-coded to test two users, one called ValidUser and one called NonexistentUser:

```
VUTest() {
    Write $Username," is the current user.",!,!

    Set sc = $$^ValidateUser("ValidUser")
    Write !

    Write "Exited validation code. ",$Username," is the current user.",!,!

    Set sc = $$^ValidateUser("NonexistentUser")
    Write !

    Write "Testing complete.",!
    Write $Username," is the current user."
    Quit 1
}
```

Suppose the **VUTest** routine were created in the User namespace of a Caché instance, the PrivilegedUser account were a member of the **`%All`** role, and only the ValidUser were to exist. Here are the results of invoking **VUTest** at a Terminal prompt:

```
Username: PrivilegedUser
Password: ***********
USER>d ^VUTest
PrivilegedUser is the current user.

Validating ValidUser...
ValidUser is a valid user.
ValidUser belongs to the following login roles: %Manager

Exited validation code. PrivilegedUser is the current user.

Validating NonexistentUser...
NonexistentUser is not a valid user.

Testing complete.
PrivilegedUser is the current user.
USER>
```

7

Services

There are various pathways for connecting to a Caché instance for users, applications, and even other Caché instances. These pathways are managed by Caché services, which serve as gatekeepers for connecting to Caché. Because Caché services are the primary means by which users and computers connect to Caché, their management is an essential part of security administration.

Topics in this chapter are:

7.1 Available Services

The Services page (**System Administration** > **Security** > **Services**) provides a list of services that Caché provides.

There are two groups of services:

- Resource-based Services — These are services that provide user access to Caché. This kind of service needs the authentication and authorization infrastructure of Caché security, so it has an associated *resource* and uses the various available authentication mechanisms.
- Basic Services — These are services that provide connections between a Caché server and a Caché application. These do not have associated resources, so they provide little more than the *basic* security functionality of being turned on or off. Enabling or disabling them controls all forms of access.

The following lists the available services, what each controls, and what kind of service it is:

- **`%Service_Bindings`** — SQL or Objects; use of Studio [resource-based]
- **`%Service_CSP`** — Web application pages (CSP or Zen) [resource-based]
- **`%Service_CacheDirect`** — Cache Direct [resource-based]
- **`%Service_CallIn`** — The CallIn interface [resource-based]
- **`%Service_ComPort`** — COMM ports attached to a Windows system [resource-based]

- **%Service_Console** — Terminal from a Windows console (analogous to **%Service_Terminal** for MacOS, UNIX®, and Linux) [resource-based]
- **%Service_DataCheck** — The DataCheck [basic]
- **%Service_ECP** — Enterprise Cache Protocol (ECP) [basic]
- **%Service_Login** — Use of **$SYSTEM.Security.Login** [resource-based]
- **%Service_MSMActivate** — MSM Activate protocol [basic]
- **%Service_Mirror** — Caché database mirroring [basic]
- **%Service_Monitor** — SNMP and remote Monitor commands [basic]
- **%Service_Shadow** — Access to this instance from shadow destinations [basic]
- **%Service_Telnet** — Telnet sessions on a Windows server [resource-based]
- **%Service_Terminal** — Terminal from a Mac, UNIX®, and Linux console (analogous to **%Service_Console** for Windows) [resource-based]
- **%Service_Weblink** — WebLink [basic]

The table of services includes a column for each service property.

7.1.1 Notes on Individual Services

7.1.1.1 %Service_Bindings

For the **%Service_Bindings** service, there are a pair of resources that manage access: the **%Service_Object** resource and the **%Service_SQL** resource. Once a user has authenticated, these two resources control whether data is accessible to the user as either objects or SQL respectively. (If a user has table-level SQL privileges on data, then Caché automatically grants an authenticated user the **%Service_SQL:Use** privilege for the duration of the connection.)

This service also controls access to Studio. For more information about Studio and security, see the section "Integration with Caché Security" in the "Introduction to Studio" chapter of *Using Studio*.

7.1.1.2 %Service_CacheDirect

This service authenticates connections to Caché through the Caché Direct server. The Caché Direct server is available on any supported platform; clients for this service can only be on Windows.

%Service_CacheDirect manages access for two types of client-side applications:

- Client applications that use the Caché Direct client software. These have all authentication mechanisms available.
- Client applications that use the legacy CacheObject.dll library. These have no security features available; for these legacy applications, the **%Service_CacheDirect** service must enable the Unauthenticated mechanism only.

Since legacy applications can only support unauthenticated access and both types of client applications use the same service, Kerberos authentication is not available for Caché Direct clients if Caché is configured to accept connections from a legacy application; similarly, if a Caché instance is configured to accept Kerberos-authenticated connections from Caché Direct clients, legacy clients cannot connect to it.

Note: CacheObject.dll is supported for legacy applications only. New development should use either the Caché Direct client or the CacheActiveX.dll and the **%Service_Bindings** service.

7.1.1.3 %Service_Console and %Service_Terminal

These two services both provide console or terminal-style access to Caché. This functionality is analogous for both Windows and non-Windows systems; **%Service_Console** provides this functionality for Windows and **%Service_Terminal** provides this functionality for UNIX®, Linux, and Mac.

CAUTION: Terminal or console access is one of the most sensitive aspects of Caché security. If an attacker gains access to Caché in one of these ways, it can be possible to read or destroy sensitive data.

7.1.1.4 %Service_CSP

This service manages connections that serve up CSP pages. Specifically, it manages connections between the CSP Gateway and the Caché server. If there is no unauthenticated access for the service and the CSP Gateway has no valid authentication information, then there is no access to the server via CSP. Hence, if you disable unauthenticated access through this service, then you must ensure that the CSP Gateway has the information it needs to authenticate to the Caché server. For Caché login (password) access, this is a valid username-password pair; for Kerberos access, this is a valid service principal name and key table location. To specify these values use the CSP Gateway Web Management interface; for a standard installation, the URL for this is http://localhost:57772/csp/bin/systems/module.cxw, where localhost represents 127.0.0.1 for IPv4 and ::1 for IPv6.

Because **%Service_CSP** regulates the use of the Portal and its subapplications, disabling **%Service_CSP** does not disable any system applications (/csp/broker, /csp/docbook, /csp/documatic, /csp/sys, /csp/sys/exp, /csp/sys/mgr, /csp/sys/op, /csp/sys/sec, /isc/studio/rules, and /isc/studio/templates). For more information on system applications, see the section System Applications in the "Applications" chapter.

Important: If you inadvertently lock yourself out of the Portal, you can use emergency access emergency access mode to reach the Portal and correct the problem; this is described in the section Emergency Access in the chapter "System Management and Security."

7.1.1.5 %Service_DataCheck

This service regulates the use of the DataCheck utility, which provides a mechanism to compare the state of data on two systems. For more details, see the "Data Consistency on Multiple Systems" chapter of the *Caché Data Integrity Guide*, and, for security issues, particularly the section "Enabling the DataCheck Service."

7.1.1.6 %Service_ECP

A resource does not govern the use of ECP. Rather, you either enable or disable the service (this makes ECP what is called a "basic service"). This means that all the instances in an ECP configuration need to be within the secured Caché perimeter.

See the "Specifying ECP Privileges and Roles" section of the "Configuring Distributed Systems" chapter of the *Caché Distributed Data Management Guide* for details on how privileges work within an ECP configuration.

7.1.1.7 %Service_Login

This service controls the ability to explicitly invoke the **Login** method of the %SYSTEM.Security class. Calls to this method are of the form:

```
Set Success = $SYSTEM.Security.Login(username, password)
```

where *username* is the user being logged in and *password* is that user's password.

7.1.1.8 %Service_Mirror

This service regulates the use of the Caché database mirroring, which provides automatic failover between two systems. For more details about mirroring generally, see the "Mirroring" chapter of the *Caché High Availability Guide*; for more details about security for mirroring (though the use of SSL/TLS), see the "Configuring Caché to Use SSL/TLS with Mirroring" section in the "Configuring Caché to Use SSL/TLS with Mirroring" chapter.

7.1.1.9 %Service_Shadow

This service regulates the use of a Caché instance as a shadow source. For more details, see "Configuring the Source Database Server" in the "Shadowing" chapter of the *Caché Data Integrity Guide*.

7.1.1.10 Legacy Services

Caché includes support for a number of legacy services. These are all basic services, and can simply be enabled or disabled. They include **`%Service_MSMActivate`** and **`%Service_Weblink`**.

7.2 Service Properties

Each service has a set of properties that control its behavior. These can include:

- **Service Name** — Specifies the identifier for the service.
- **Description** — Provides an optional description of the service.
- **Service Enabled** — Controls whether a service is on or off. When enabled, a service allows connections to Caché, subject to user authentication and authorization; when disabled, a service does not permit any connections to Caché.

  At system start up, each service has the same state (enabled or disabled) that it had when Caché was shut down. Note that enabling or disabling a service is not simply a security setting. It determines whether or not a certain capability is provided by Caché and may, for instance, determine whether certain daemon processes are started or memory structures are allocated.
- **Allowed Authentication Methods** — Specifies the available authentication mechanisms for connecting to the service, including either of the two-factor authentication mechanisms; if multiple mechanisms are selected, the user or client can attempt to connect using any of these. The mechanisms available depend on what is selected on the **Authentication/CSP Session Options** page (**System Administration** > **Security** > **System Security** > **Authentication/CSP Session Options**). If a service supports multiple authentication mechanisms, these are used according to the Caché rules of cascading authentication.

  If either two-factor authentication mechanism is enabled, it has a check box. If visible, these are:

  – **Two-factor Time-based One-time Password** — A Caché user's mobile phone or an authentication device serves as a second authentication "factor"; Caché and the phone or device share a secret key. This key is used to generate a time-based one-time password (TOTP), which the user must enter at a prompt as part of the authentication process.

  – **Two-factor SMS** — A Caché user's mobile phone serves as a second authentication "factor"; Caché sends a eight-digit security token to the phone, which the user must enter at a prompt as part of the authentication process.

  For more details, see the section "Configuring Two-factor Authentication" in the "Authentication" chapter.

  Note: If two-factor authentication is enabled for an instance, this check box appears on the **Edit Service** page for all its services. However, two-factor authentication is only available for **`%Service_Bindings`**, **`%Service_Console`**, and **`%Service_Terminal`** (and only when it is enabled for the instance).

- **Allowed Incoming Connections** — Specifies a list of IP addresses or machine names from which the service accepts connections; if a service has no associated addresses or machine names, then it accepts connections from any machine. This capability can be very useful with multi-tier configurations; for example, with the CSP service, it can be used to limit the set of Web servers that can connect to Caché. The Allowed Incoming Connections facility for ECP has additional features described in the Restricting ECP Application Server Access section of the *Caché Distributed Data Management Guide*.

For a resource-based service, the service can be specified as public. Public services are available to all authenticated users, while non-public services are available only to those users with Use permission on the service's resource. This value is displayed on the main Services page (**System Administration** > **Security** > **Services**) and is set on the **Edit Resource** page for the service's resource. Possible values are:

- N/A — The service has no associated resource; this means that service can simply be turned on or off.
- NO — Access is available to any user holding a role that has the Use permission on the service's resource. This is checked after authentication.
- YES — Access is available to any user.

Note: A change to a service only takes effect after the service is restarted.

7.3 Services and Authentication

Basic services do not support authentication for Caché security. They are simply turned on and off. For those services, enabling the service ensures that it accepts all connections. For these services, the assumption is made that all instances or machines using the service are within a secure perimeter and can only be accessed by valid users. This includes **%Service_ECP**, **%Service_MSMActivate**, **%Service_Monitor**, **%Service_Shadow**, and **%Service_Weblink**.

To enable an authentication mechanism for a resource-based service, you must first enable it for the Caché instance on the **Authentication/CSP Session Options** page (**System Administration** > **Security** > **System Security** > **Authentication/CSP Session Options**). Resource-based services support authentication mechanisms as listed in the table below. If a service has more than one authentication mechanism enabled, Caché supports cascading authentication.

Table 7–1: Services with Authentication Mechanisms

Service Name	KRB Cache	KRB Login	Del	LDAP	OS	Caché	Un
%Service_Bindings	N	Y	Y	Y	N	Y	Y
%Service_CSP	N	Y	Y	Y	N	Y	Y
%Service_CacheDirect	N	Y	Y	Y	N	Y	Y
%Service_CallIn	N	N	Y	Y	Y	N	Y
%Service_ComPort	N	N	Y	Y	N	Y	Y
%Service_Console	Y	Y	Y	Y	Y	Y	Y
%Service_Login	N	N	Y	Y	Y	Y	Y
%Service_Telnet	N	Y	Y	Y	N	Y	Y
%Service_Terminal	Y	Y	Y	Y	Y	Y	Y

Key:

- KRB Cache — Kerberos Cache
- KRB Login — Kerberos Login
- Del — Delegated authentication
- LDAP — LDAP authentication
- OS — Operating-System–based authentication
- Caché — Caché Login
- Un — Unauthenticated access

For each resource-based service, if there are multiple enabled authentication mechanisms, then Caché attempts to authenticate users going from the strongest enabled form of authentication to allowing unauthenticated access (if that is enabled). This is process is described in the section Cascading Authentication in the "Authentication" chapter.

7.4 Services and Their Resources

For resource-based services, the properties of the service itself govern access to Caché; at the same time, the properties of the service's resource govern access to and behavior of the service. For all resource-based services except **%Service_Bindings**, the service's associated resource has the same name as the service itself; hence the **%Service_CSP** resource manages access for the **%Service_CSP** service. (The **%Service_SQL** and **%Service_Object** resources manage access for **%Service_Bindings**.)

A resource itself has only two related properties: whether or not it is public and, if it is public, what its public permissions are; for a service resource, the only relevant permission is Use. If it is public, then all users have Use permission on the service. For more information on resources, see the chapter "Resources."

Independent of privileges for other resources, service privileges provide little to the user. For example, for the **%Service_CacheDirect:Use** privilege to be useful, the user must also hold the **%DB_<database-name>:Write** privilege for the database where updates are to occur.

8

Applications

Applications are the primary way that most users interact with Caché. Because of this, application security can provide a key set of tools for regulating user actions. Application security uses Caché authorization tools to ensure that only the appropriate users can use an application. An application can also escalate its users' privileges.

This chapter covers the following topics:

- Applications, Their Properties, and Their Privileges
- Application Types
- Creating and Editing Applications
- System Applications

8.1 Applications, Their Properties, and Their Privileges

From a security standpoint, an application:

- is an entity that allows users to perform actions
- is associated with one or more resources
- has properties that govern its behavior
- can enhance its users' privileges while they are running it
- can include programmatic privilege checks

All these characteristics and capabilities are part of the Caché authorization tools. If an application is formally defined with Caché, then it can use this functionality. There are three kinds of applications:

- web applications, which are applications built with CSP or Zen
- privileged routine applications, which are applications typically built from Caché classes, in ObjectScript routines, or both
- client applications, which are applications built using Caché Direct

This section covers:

- Applications and Their Properties
- Associating Applications with Resources

- Applications and Privilege Escalation
- Checking for Privileges Programmatically

8.1.1 Applications and Their Properties

Applications allow you to specify a set of permitted actions, such as reading from and writing to databases or using other assets. To do this, Caché supports what is called an *application definition*, which is a set of information that represents the application within Caché. (The relationship of an application definition to an application is analogous to that of the relationship of a resource to an asset.) By establishing an application definition, you can control and manage the application.

Important: Applications and application definitions are frequently referred to interchangeably. The distinction is only important in settings where the executable code or user experience of that code differs from the representation of that code within Caché. The former is the application itself and the latter is the application definition.

Each application, through its application definition, has the following properties:

Name

The name of the application. This must start with an alphabetic character and must be followed by alphabetic, numeric, or underscore characters. The application definition's name is independent of the name of any resource.

Description

A description of the application.

Enabled

A switch that specifies if the application is available for use. If the application is not enabled, no one can run the application — not even a member of the **%All** role. For more details on how this property governs each kind of application, see the appropriate section: web applications, privileged routine applications, or client applications.

Resource

A resource that manages application behavior. The resource has different effects for different application types: for web applications and client applications, it controls whether or not users have access to the application; for privileged routine applications, it controls the application's privilege escalation. If a web or client application has no resource, then any user can run the application; if a privileged routine application has no resource, then the application escalates privileges for any user.

Each application definition can only be associated with a single resource. For more details on how this property affects each kind of application, see the appropriate section: web applications, privileged routine applications, or client applications. For more information on how applications interact with resources, see "Associating Applications with Resources."

Application Roles

One or more roles to which application users are assigned. While running the application, the user is assigned to its application roles by appending these roles to the list of roles in the *$Roles* variable. For more information on using application roles, see the section "Applications and Privilege Escalation".

Matching Roles

One or more roles that cause the user be assigned to some additional roles (called "target roles") while running the application. If users are assigned to a matching role, then, while using the application, they are also assigned to any target roles. This is done by appending these roles to the list of roles in the *$Roles* variable. For example,

if **%Admin_Manage** is a matching role, then being a member of that role might cause the application user to also become a member of the target role of **%Admin_Secure**. For more information on using matching roles, see the section "Applications and Privilege Escalation".

All applications have these properties. Each of the three application types also has its own other, unique characteristics.

8.1.2 Associating Applications with Resources

When an application (and therefore its application definition) is a single, unitary whole, it has a single resource — a one-to-one relationship. You can also specify that multiple applications (and therefore multiple application definitions) are associated with a single resource — a many-to-one relationship. For any number of applications, the situation reduces to some combination of these two conditions.

A more complex case is when an application is composed of separate parts, each of which is known as a *sub-application.* An application made up of a group of sub-applications is designed to behave as a single, unitary whole, while allowing different sub-applications to require different kinds of or levels of security. In this situation, it is useful to give each sub-application its own application definition and to associate it with a separate resource. This way, each sub-application can have its own independent security-related behavior. While, from the application perspective, there are multiple sub-applications that compose the larger application, from the Caché security perspective, there are simply multiple, individual applications — each with its own application definition — and users pass among them without knowing it. Again, this reduces to the one-to-one and many-to-one cases, except that the multiple application definitions represent what appears to the end-user as a single application. Because the users have already authenticated and by that process have established their roles, then passing from one sub-application to another requires no authentication.

For example, suppose there is an expense-reporting application where all employees can enter expense reports, but only accounting officers can generate checks. In this case, the application might appear as a single whole and the functionality to generate checks would be grayed out for all employees except the accounting officers. To accomplish this, there would be two separate sub-applications, one for entering reports and another for generating checks. All users would be able to use the former and only accounting officers would be able to use the latter. For them, there would be no visible difference between what might simply appear as two separate screens in a single application.

8.1.3 Applications and Privilege Escalation

Since you can use application resources to escalate a user's roles, they provide a mechanism for meeting authorization needs that shift dynamically. To perform privilege escalation for an application:

1. Given an existing application, create a resource and then associate the application with the resource.
2. Create one or more roles that hold the Use permission for the resource.
3. Determine the list of privileges that the application requires in order to run. If the application has sub-applications, there may be more than one such list.
4. Associate each list of privileges with a particular role. Establish each role as an *application role* for the application or sub-application.
5. Establish any *matching roles* for the application or sub-application. Each matching role has one or more *target roles* associated with it.
6. When a user successfully invokes an application, Caché performs two actions:
   - For the duration of application use, it assigns the user to any application roles. (For privileged routine applications, this depends on successfully invoking the **AddRoles** method, as described in the section "Privileged Routine Applications.")
   - If the user is assigned to any matching role, the application assigns the user to any target roles for the duration of application use. (Again, for privileged routine applications, this depends on successfully invoking the **AddRoles** method, as described in the section "Privileged Routine Applications.")

For example, suppose that an application has its own resource, called **AppRsrc**. Two roles hold the **AppRsrc:Use** privilege; these are **AppUser** and **AppOperator**. **AppOperator** is also a matching role, where the target role is **%Manager**. In this scenario, when a user belonging to the **AppUser** role invokes the application, the value of *$Roles* does not change; when a user belonging to **AppOperator** invokes the application, the value of *$Roles* expands to include **%Manager**. If the application has an application role of **AppExtra**, then a user belonging to the **AppUser** role receives the **AppExtra** role when invoking the application. In the first scenario (matching role only), belonging to the **AppOperator** role causes privilege escalation; in the second scenario (matching role and application role), belonging to either role results in privilege escalation.

8.1.3.1 User-Based and Application-Based Security

The Caché security model allows for flexible privilege assignment that can be user-based, application-based, or both. The use of an application can be limited to specific users or open to any users. For those users authorized to use the application, there can be several behaviors:

- The application can run with the user's privileges alone.
- The application can escalate privileges for only some users (using matching and target roles).
- The application can escalate privileges for all users (using application roles).
- The application can escalate some privileges for all users and only escalate other privileges for certain users (using a combination of matching/target roles and application roles).

Hence, you have control of whether application use is limited to specific users or open to any users; simultaneously, you also have control of whether an application runs with the user's privileges or with its own privileges. This enables Caché to provide a very flexible model:

Table 8–1: Protection/Escalation Matrix for Secured Applications

Privilege Level / Protection Level	Public Application	Restricted Application
With User-Dependent Privileges	1. Any user can run the application. Application runs with user privileges.	2. Only specified users can run the application. Application runs with user privileges.
With Privilege Escalation	3. Any user can run the application. Application runs with (expanded) application privileges through application roles and matching roles.	4. Only specified users can run the application. Application runs with (expanded) application privileges through application roles and matching roles.

Each of the scenarios described in the previous table is commonly used for a different authorization model:

1. Public Application with User-Dependent Privileges

This describes an application available to any authenticated user; when run, the application grants no additional privileges. For example, for a company's contact database, any user belonging to the company-wide role can get the office phone number and email address for any employee; managers hold greater privileges, which entitle them to view employee home phone numbers; HR staff hold even greater privileges, which entitle them to view and update full records. The application is accessible to all employees, and its behavior depends on privileges that each user already has when invoking it — the application itself grants no roles.

2. Restricted Application with User-Dependent Privileges

This describes an application available only to a user who belongs to a specified role; when run, the application grants no additional privileges. For example, a company may have a payroll application for its hourly employees, which displays the

number of hours worked, pay rate, and so on. To run the application, a user has to be a member of either the **HourlyEmployee** role or the **HourlyManager** role. Once running, the application checks which role was present: members of **HourlyEmployee** can see and *not* edit their own data, while members of **HourlyManager** can see and edit data for their own reports. An employee who is a member of the **HourlyEmployee** role can run the application to check the accuracy of personal data; any other employee (such as one on a salary and who is not a member of the required role) cannot even run the application.

3. Public Application with Privilege Escalation

This describes an application available to any authenticated user; when run, the application escalates privileges based on the roles to which the user belongs. (The application can also escalate privileges only for certain roles.) For example, suppose a university has an application where students can review and update their records. Here, any student is an authenticated user and can edit his or her own contact information. To support this functionality, the application includes code for editing an entry; this code checks that the entry being edited matches the authenticated user and, if so, escalates its own privileges to update the record, and then restores the privileges to their previous state. If one student attempts to update another's record, then the check fails, there is no privilege escalation, and the update does not occur. The application might also check if the user is a member of the registrar's office role, in which case it would be possible to update information more widely.

4. Restricted Application with Privilege Escalation

This describes an application available only to a user who belongs to a specified role; when run, the application escalates privileges based on the roles to which the user belongs. (The application can also escalate privileges only for certain roles.) For example, a hospital's emergency room might have an application that grants the attending doctor special, wider privileges for viewing the records of patients currently admitted for emergency care. Because of the potentially critical nature of emergency-room cases, the doctor needs to be able to view more information in this setting than while simply making rounds; hence, the privileges are escalated.

8.1.4 Checking for Privileges Programmatically

An application can also include code to check if its users have privileges required to perform a particular action. To do this, use the **$SYSTEM.Security.Check** method. The syntax of this call is:

```
Set status = $SYSTEM.Security.Check(app_resource, app_permission)
```

where

- *app_resource* is the resource for which the user must hold a permission
- *app_permission* is the permission that must be held.
- *status* is the method's return value of TRUE or FALSE (1 or 0).

For example, if an application requires a user to have Write permission on the **Application_Order_Customer** resource, then the **Check** call would be:

```
Set status = $SYSTEM.Security.Check("Application_Order_Customer", "WRITE")
```

Note: No privilege is required to call **$SYSTEM.Security.Check**.

8.2 Application Types

There are three types of applications:

- Web Applications
- Privileged Routine Applications

- Client Applications

8.2.1 Web Applications

These are applications built with CSP and Zen (which itself uses CSP). They connect to Caché using the **%Service_CSP** service.

For web applications, security information is maintained as part of the CSP session. That is, the values of *$USERNAME* and *$ROLES* are preserved across page requests. (More specifically, when processing begins for a page, *$ROLES* contains the user's roles as well as roles defined for the application. It does not contain roles that have been dynamically added during processing of a previous page via `SET $ROLES` or **$SYSTEM.Security.AddRoles**. This is true for both stateless and "state-full" sessions.

With Web applications (whether they are built with Caché Server Pages, Zen, or with some other technology), the client (that is, the browser) typically does not send a username and password to the server when it connects. Instead, the user requests a page and the server responds with a login page that must be completed before the rest of the application can be accessed. If two-factor authentication is enabled, then, once the user has provided a username and password, the server displays a page for entering the security code; if authentication succeeds, the user has access to the application.

Note: With two-factor authentication, the server always displays the page for entering the one-time security token — even if the username-password pair is not valid. After the user enters the one-time security token, the server displays a message that access is denied, and provides a minimum of information that could be used against the system.

CSP security processing occurs as follows:

1. As each page request is received, its application is determined from the URL. If the application is not enabled, there is no connection.
2. If the application is the same as the application for the last page processed for the CSP session, then there is already a connection, so no further security checking is required.
3. If the Use permission for **%Service_CSP** is not public and the user does not hold this permission, there is no connection.
4. If the application or the CSP service require authentication and the user has not already been authenticated, then Caché checks if the request includes *CacheUserName* and *CachePassword* parameters:
   a. If *CacheUserName* and *CachePassword* are present, the CSP server attempts to log in; if the login succeeds, it checks if the user has the Use permission for the application resource. If either of these fail, there is no connection.
   b. If *CacheUserName* and *CachePassword* are not present, CSP displays an application-specific login page, if one is defined in the web application configuration. (This is the only page in a secure application that can be used prior to login.) If there is no application-specific login page, the username and password fail authentication, or the user does not have the Use permission on the application resource, there is no connection.

To edit a web application, the Portal provides the following pages:

- Creating and Editing an Application: The General Tab
- Editing an Application: The Application Roles Tab
- Editing an Application: The Matching Roles Tab

8.2.2 Privileged Routine Applications

A *privileged routine application* grants the privilege to escalate roles to one or more classes or routines for the users of those classes or routines. The classes or routines in a privileged routine application are written in ObjectScript, Caché Basic, or MVBasic. To use a privileged routine application:

1. As an administrative task, create an application definition in the Management Portal, as described in the "Creating and Editing an Application: The General Tab" section.
2. As an administrative task, add classes or routines to it, as described in the "Editing an Application: The Routines/Classes Tab" section.
3. As a development task, edit the application definition's classes or routines in your development environment to escalate roles, as described in the "Escalating Roles in a Privileged Routine Application: The AddRoles Method" section.

The Portal provides the following pages to edit a privileged routine application (which includes the first two mentioned above):

- Creating and Editing an Application: The General Tab
- Editing an Application: The Application Roles Tab
- Editing an Application: The Matching Roles Tab
- Editing an Application: The Routines/Classes Tab

8.2.2.1 Escalating Roles in a Privileged Routine Application: The AddRoles Method

To escalate roles in a privileged routine application, invoke the **AddRoles** method of the %SYSTEM.Security class. To call **AddRoles**, the syntax is:

```
Set sc = $SYSTEM.Security.AddRoles("AppDefName")
```

where *AppDefName* is the name of the application definition and *sc* is a status code. If a class or routine is part of an application definition and the user is appropriately privileged, then calling **AddRoles** from that class or routine escalates privileges to include any application roles (as described in "Editing an Application: The Application Roles Tab") and any relevant matching roles (as described in "Editing an Application: The Matching Roles Tab").

Important: If a routine does not use curly braces to delimit code in its entry points, then control can pass from one entry point to another, possibly resulting in overprivileged users and unintended levels of access. For more information on structuring routines, see the "User-Defined Code" chapter of *Using Caché ObjectScript*.

Processing of the call to **AddRoles** occurs as follows:

1. If the call is not from a privileged class or routine, then the call fails.
2. If the required resource specified in the application definition is not public and the user invoking the method or routine does not have Use permission on this resource, then the call fails.
3. Otherwise, the call succeeds.

Tip: To cause the user to give up any application roles and to revert to login roles when control passes out of scope for the routine that escalates privileges, include the following command *prior* to the call to **AddRoles**:

```
New $Roles
```

For more information on these topics, see the "Programmatically Managing Roles" section of the "Roles" chapter.

8.2.2.2 An Example of Using a Privileged Routine Application

Suppose there is an application that uses a database called DB1. This application's users hold the **%DB_DB1** role only, so they all have privileges for DB1. Some of the application's users also require temporary access to another database, DB2. Those users get access to DB2 through the **PRAEscalate** method ("PRA" for "Privileged Routine Application") of the

PRATestClass class, which escalates their privileges; specifically, **PRAEscalate** adds the **%DB_DB2** role, which provides access to DB2.

To enable the **PRAEscalate** method to add the **%DB_DB2** role for the appropriate users, the following security items must exist:

- A resource called **PRATestResource**, which is not public.
- A role called **PRA_DB2**, which has only one privilege: **PRATestResource:Use**.
- The **%DB_DB2** role, which was created when the DB2 database was created.
- A privileged routine application called PRATestApp. Related to PRATestApp:
  - Users must have the **PRATestResource:Use** privilege to run the PRATestApp application, Therefore, users who require access to the DB2 database must have the **PRA_DB2** role (which grants the **PRATestResource:Use** privilege).
  - The PRATestClass class is part of the PRATestApp application. (To include the class in the application, do so on the **Routines/Classes** tab of the **Edit** page for PRATestApp.)
  - The **%DB_DB2** role is an application role for PRATestApp. (To specify an application role, do so on the **Application Roles** tab of the **Edit** page for PRATestApp.)

Given this setup and two users, PRATestBasicUser and PRATestDB2User:

- PRATestBasicUser is a member of **%DB_DB1** only. Therefore, the PRATestApp application does not escalate PRATestBasicUser's roles, and the user *cannot* use the part of the application that requires access to DB2.
- PRATestDB2User is a member of the **%DB_DB1** and **PRA_DB2** roles. Therefore, the PRATestApp application does escalate PRATestBasicUser's roles, and the user *can* use the part of the application that requires access to DB2.

Here is the code of **PRAEscalate**:

```
Method PRAEscalate()
 {
   Write "This method is a part of the privileged routine application ",!
   Write "called PRATestApp.",!
   Write "The user invoking this routine is ",$Username,!
   Write "The current value of $Roles is ",$Roles,!,!
   Write "Calling the AddRoles method...",!,!
   New $Roles
   Set sc = $SYSTEM.Security.AddRoles("PRATestApp")
   If sc = 1
   {
      Write "Application roles have been added.",!
      Write "$Roles now is ",$Roles,!,!
   } Else {
      Write "The call to AddRoles has failed.",!
      Do $system.Status.DecomposeStatus(sc,.Err)
      Write Err(Err),!
   }
 }
```

Here is the terminal session where PRATestDB2User runs this routine:

```
Username: PRATestDB2User
Password: ********
USER>set x = ##class(PRATestClass).PRATest()
This method is a part of the privileged routine application
called PRATestApp.
The user invoking this routine is PRATestDB2User
The current value of $Roles is %DB_DB1, PRA_DB2

Calling the AddRoles method...

Application roles have been added.
The current value of $Roles is %DB_DB1, %DB_DB2, PRA_DB2
Removing %DB_DB2 from $Roles...
$Roles now is %DB_DB1, PRA_DB2

USER>
```

Here is the terminal session where PRATestBasicUser runs this routine:

```
Username: PRATestBasicUser
Password: ********
USER>set x = ##class(PRATestClass).PRATestMethod()
This method is a part of the privileged routine application
called PRATestApp.
The user invoking this routine is PRATestUser
The current value of $Roles is %DB_DB1

Calling the AddRoles method...

The call to AddRoles has failed.
ERROR #862: User is restricted from running privileged application PRATestApp
-- cannot execute.

USER>
```

8.2.3 Client Applications

These are applications that use Caché Direct to connect to Caché.

Important: Client application security is available only for applications using Caché Direct. Any client/server application using other technology, such as ActiveX, cannot use application security. For these applications, use the authentication tools described in chapter "Authentication."

Caché enables executables built using Caché Direct to be identified to the system. When such an executable attempts to connect to the server, Caché performs the following processing:

1. If the **%Service_CacheDirect** is not enabled, there is no connection. If **%Service_CacheDirect** is enabled, the attempt to connect proceeds.
2. If the Use permission for the **%Service_CacheDirect** service is not public and the user does not hold the permission on the service, then the connection is rejected.
3. If the Use permission for a resource specified in the application definition is not public and the user does not hold the permission on the application, then the connection is rejected.
4. If the process has Read or Write permission on the namespace to which the connection is being made, the connection is accepted; otherwise, the connection is rejected.
5. Once the connection is accepted, application roles are added. Similarly, if the user is a member of any matching roles, then the appropriate target roles are added.

To edit a client application, the Portal provides the following pages:

- Creating and Editing an Application: The General Tab
- Editing an Application: The Application Roles Tab
- Editing an Application: The Matching Roles Tab

8.3 Creating and Editing Applications

This section describes several topics:

- Creating and Editing an Application: The General Tab
- Editing an Application: The Application Roles Tab
- Editing an Application: The Matching Roles Tab
- Editing an Application: The Routines/Classes Tab

8.3.1 Creating and Editing an Application: The General Tab

For creating and editing an application, the **General** tab holds fields that specify information needed for basic operation of the application.

8.3.1.1 Creating an Application

To create an application, the procedure is:

1. In the Management Portal menu, select **System Administration** > **Security** > **Applications**, which displays the different application types.
2. Choose **Web Applications**, **Privileged Routine Applications**, or **Client Applications**. This displays the page for the selected application type.
3. In the upper-left corner of the applications page, click the button to create a new application. Depending on the application you are attempting to create, select **Create New Web Application**, **Create New Privileged Routine Application**, or **Create New Client Application**. This displays the application editing page for the selected application type. You can then edit the application as if it already existed using the information in the next section.

8.3.1.2 Editing an Application's General Attributes

You can create or modify settings for how you want Caché to process a specific CSP application on the **Edit Web Application** page of the Management Portal as follows:

1. In the Management Portal menu, select **System Administration** > **Security** > **Applications** > **Web Applications**.

   This lists configured web applications. The **Type** column identifies an application as a user application (`CSP`) or a system application (`CSP,System`; a CSP-based utility included with Caché).
2. Select an application, click **Edit**, and enter or change the information.
3. When finished with edits, restart Caché for the new settings to take effect.

The **General** tab displays the following options:

Table 8–2: Edit Web Application Settings — General Tab

Field	Description
Name	Enter a name for the application. The name must include a leading slash (/), such as in the /csp/acme application.
Description	Enter a description.
Namespace	The Caché namespace in which pages for this application are run.

Field	Description
Namespace Default Application	Sets this application as the default application for this namespace. A call to %System.CSP.GetDefaultApp returns this application as the default for the namespace. Caché import functions use this to deal with cases such as importing a CSP page from an XML file where the current namespace does not have the CSP application that the CSP file was exported from. CSP imports the CSP file into the default CSP application for the namespace.
Application	Controls whether an application is available. When selected, an application is available, subject to user authentication and authorization; when unchecked, it is not available.
CSP/Zen	Enables CSP to serve CSP and Zen pages. Uncheck to disable. For more information on Zen, see the book Using Zen.
DeepSee	Enables the %-class required for DeepSee. Uncheck to disable.
iKnow	Enables the %-class required for iKnow. Uncheck to disable.
Inbound Web Services	Enables CSP to serve SOAP requests. Uncheck to disable. For more information on SOAP and Caché web services, see the book Creating Web Services and Web Clients in Caché.
Permitted Classes	Specify classes that may be run in this application in three ways: 1) ObjectScript match pattern. Example: `1"myclass".3N` allows myclass123.cls to run in this application, but not myclassxy.cls. 2) ObjectScript expression that evaluates to a boolean, prefixed with `@`. The requested class name is passed as a variable named `class`. Example: `@class = "PermittedClasses.PermittedPage"` 3) Call to a class method (can also use *@syntax*). Example: `##class(MyPackage).CheckClassIsPermitted(class)` See also "Enabling Application Access to %CSP Pages".
Resource Required	Specifies a resource for which users must have the Use permission (enabled as part of a privilege in a role) in order to run the application. For information on resources and permissions, see the "About Resources" section in the *Caché Security Administration Guide*.
Group by ID	Enter a group name for this application to share authentication privileges with all other applications with this group name. All applications with this group name stay in authentication sync. If you log out of any of these applications, you are logged out of all of them. If you then try to return to a page of any of these applications, you need to log in again. Once logged in, however, you can go from one application to another without logging in again. (The only exception is that if any of these applications are unauthenticated, they are not treated as part of the authentication cluster.) Note that **Group by ID** is attached to an application, not a namespace. So applications with the same Group by ID share authentication regardless of namespace. For more specifics, see the section "By-ID Groups".

Field	Description
Allowed Authentication Methods	Specifies the available authentication mechanisms for connecting to the application. The options displayed here are determined by what is checked on the **Authentication Options** page (**Management Portal** >**System Administration** > **Security** > **System Security** > **Authentication/CSP Session Options**). If an application supports multiple authentication mechanisms, authentication occurs as follows: • If more than one option is enabled *including* **Unauthenticated**, the user has the choice of logging in without providing a username and password. • If all options are enabled *and* the user enters a username and password, then Caché attempts to perform cascading authentication. • If the selected options are Kerberos and Password and Unauthenticated is *not* selected, then the user must provide a username and password. Caché attempts to perform authentication first using Kerberos and then Caché password login. If either authentication succeeds, the user is authenticated; if both fail, the user is denied access to the application. For more information on authorization, see the chapter Authentication in the *Caché Security Administration Guide*.
Session Timeout	The default session timeout in seconds. You can override this value using the AppTimeout property of the %CSP.Session object. Note that if a session changes CSP applications during its life span, its timeout value will not be updated according to the default timeout defined in the application that the session moved into. For example, if a session starts out in CSP Application A, with a default timeout of 900 seconds, and then moves into CSP Application B, which has a default timeout of 1800 seconds, the session will still timeout after 900 seconds. If you want an application change to result in the session timeout being updated to that of the new application, use a session event class, override the **OnApplicationChange** callback method, and add code to handle the update of the AppTimeout property of the *%session* object.
Event Class	Specifies the default name of the CSP class (a subclass of %CSP.SessionEvents) whose methods are invoked for CSP application events, such as a timeout or session termination. You can override this value using the EventClass property of the %CSP.Session object. Note: Use only a class name without an extension (such as .cls or .zen) as a value of this setting, for example MyApplication.SessionEvents.
Use Cookie for Session	Whether you want CSP to track which session a browser is in by using cookies or a URL-rewriting technique (placing a value in each URL). (In order for an application to actually use cookies, regardless of this setting, the application has to be written to use cookies.) Choices are always use cookies, never use cookies, or, the default, automatically detect whether a user has disabled cookies. The specific choices are: a) (default) Use cookies (**Always**), b) Do not use cookies (**Never**), c) Use cookies unless the client browser has disabled cookies (**Autodetect**). This option does not set *whether* an application uses cookies (this is dependent on how the application is written); it merely controls *how CSP manages sessions*. If the user has disabled cookies, the application uses URL rewriting.

Field	Description
Session Cookie Path	Scope of the session cookie. This determines which URLs the browser uses to send the session cookie back to Caché. If your application name is `myapp`, it defaults to /myapp/ meaning it only sends the cookie for pages under /myapp/. If you restrict this to only what is required by your application, it prevents this session cookie being used by other CSP applications on this machine, or from being seen by any other application on this web server. On the other hand, browsers and cookies are case-sensitive. Setting the session cookie to '/' can prevent license or session problems if, for example, an application name changes from capital to lowercase letters.
Dispatch Class	Identifies the corresponding custom subclass of %CSP.REST for implementing a REST service. See Creating Rest Services for more information.
Serve Files	
Serve Files Timeout	Length of time static files should be cached by the browser in seconds. Default is 3600.
CSP Files Physical Path	The directory on the Caché server in which CSP source files are stored. The path is relative to the *install-dir*/csp/ directory on the Caché server system.
Package Name	The name of an optional package prefix used by the CSP compiler. This name is prepended to the package names used for classes created from CSP files. If this field is not specified, the default value of `csp` is used.
Default SuperClass	The name of the default superclass used by the CSP compiler for classes created from CSP files. The default is %CSP.Page.
Recurse	Specifies whether to include subdirectories within this application or not. If *UPath* is the URL Path and *PPath* is the Physical Path, then with *Recurse* set to **Yes**, UPath/xxx/yyy looks for CSP files in PPath/xxx/yyy. If *Recurse* is set to **No**, only files directly contained in *UPath* are used.
Auto Compile	**Auto Compile** works with **Lock CSP Name** to determine when an application is compiled.
Lock CSP Name	If two Web applications both point to the same namespace and **Lock CSP Name** is set to **Yes** (true) for both Web applications, then any CSP page in that namespace is displayed only through the Web application where it was last compiled. You can determine which Web application applies to a CSP page by looking at the page class's CSPURL parameter. For example: `Parameter CSPURL = "/csp/samples/zipcode.csp";` If you set **Lock CSP Name** to **Yes**, set **Auto Compile** to **No**. If **Auto Compile** is also set to **Yes**, then a change to any CSP page in the namespace triggers a recompilation of that page (including the CSPURL parameter) when it's next requested. A change to either Web application definition also triggers recompilation of any pages in the namespace when next requested. In these cases, this allows the *next* request for a page to use *either* of the Web applications, and from then on, the page is displayed only through the last Web application used to request it. For Zen pages in this scenario, if you set **Lock CSP Name** to **Yes**, set the CSPURL parameter of each ZEN page. See %CSP.Page for details. The **AutoCompile** setting doesn't affect ZEN pages.

Field	Description
Login Page	The name can be the name of a CSP page, a Zen page, or a CSP-enabled class which may be prefixed with the full CSP application path. All of the following are acceptable: mylogin.csp, /csp/user/mylogin.csp, MyApp.LoginPage.zen, /csp/user/MyApp.LoginPage.cls. In most cases, the login page is loaded before the user has logged in to Cache, so the requesting process runs under the CSPSystem user (or whatever user connects the CSP Gateway to Caché). As a result, the CSPSystem user must have sufficient privileges to load and run the code in the login page, which generally requires READ permissions on the resource protecting the database in which the login page is located.
Change password page	Name of page to use when changing password.
Custom Error Page	The name of a .csp or .cls page that is displayed if an error occurs when generating a page within this application.

To perform general editing on a privileged routine application or a client application, the procedure is:

1. In the Management Portal menu, select **System Administration** > **Security** > **Applications**, which displays the different application types.
2. Choose **Web Applications**, **Privileged Routine Applications**, or **Client Applications**. This displays the page for the selected application type.
3. On the applications page, select the application to edit by clicking on its name. This displays the **Edit** page for the application.
4. By default, the **General** tab appears. For privileged routine applications and client applications, the page's fields are:
   - **Privileged routine application name** or **Application path and name** — An identifier for the application
   - **Description** — A description of the application
   - **Enabled** — Whether or not the application is available. When enabled, an application is available, subject to user authentication and authorization; when disabled, it is not available.
   - **Resource required to run the application** — A resource for which users must have the Use permission (enabled as part of a privilege in a role) in order to perform certain actions. For web and client applications, this resource is required in order to simply operate the application; for privileged routine applications, this resource is required to invoke the **AddRoles** method, which gives the application its ability to escalate roles.

8.3.2 Editing an Application: The Application Roles Tab

You can configure an application so all its users receive certain roles, which are known as *application roles*.

To specify application roles for an application, the procedure is:

1. In the Management Portal menu, select **System Administration** > **Security** > **Applications**, which displays the different application types.
2. Choose **Web Applications**, **Privileged Routine Applications**, or **Client Applications**. This displays the page for the selected application type.
3. On the applications page, select the application to edit by clicking on its name. This displays the **Edit** page for the application.
4. On the **Edit** page, go to the **Application Roles** tab.

5. To specify one or more application roles, click on the roles listed in the **Available** list. Move them into the **Selected** list with the arrows.
6. Click **Assign** to establish the application roles.

8.3.3 Editing an Application: The Matching Roles Tab

You can configure an application to support what are called *matching roles* and *target roles*. If a user is assigned to a matching role, then running the application causes Caché to assign the user to any associated target roles. An application can have multiple matching roles; for each matching role, it can have multiple target roles; and multiple matching roles can have the same target role.

To establish a matching role and its target roles for an application, the procedure is:

1. In the Management Portal menu, select **System Administration** > **Security** > **Applications**, which displays the different application types.
2. Choose **Web Applications**, **Privileged Routine Applications**, or **Client Applications**. This displays the page for the selected application type.
3. On the applications page, select the application to edit by clicking on its name. This displays the **Edit** page for the application.
4. On the **Edit** page, go to the **Matching Roles** tab.
5. On the **Matching Roles** tab, choose the role to be a matching role from the **Select a matching role** drop-down.
6. To select the accompanying target role(s), click on the roles listed in the **Available** list. Move them into the **Selected** list with the arrows.
7. Click **Assign** to establish the matching role and its target role(s).

8.3.4 Editing an Application: The Routines/Classes Tab

This tab is for privileged routine applications only. On this tab, you can specify the classes or routines that are part of a privileged routine application.

To add a class or routine to privileged routine application, the procedure is:

1. In the Management Portal menu, go to the **Privileged Routine Applications** page (**System Administration** > **Security** > **Applications** > **Privileged Routine Applications**).
2. On the **Privileged Routine Applications** page, there is a list of applications that can be edited. Click the **Name** of the relevant application. This displays the **Edit Privileged Routine Application** page for the application.
3. On the **Edit Privileged Routine Application** page, go to the **Routines/Classes** tab.
4. In the **Routine/Class name** field, enter the name of the routine or class to be added to the application.
5. Specify whether you are adding a **Routine** or a **Class** by selecting the corresponding check box.
6. Click **Assign** to add the routine or class to the application.

8.4 System Applications

Each Caché instance comes with a number of system applications. There is always access to these applications, even if the **%Service_CSP** service is disabled. The system applications are:

Table 8–3: Caché System Web Applications

Name	Purpose or Managed Interactions	Associated Resource
/csp/broker	Common static file store. For InterSystems internal use only.	
/csp/docbook	InterSystems main documentation (including this book).	
/csp/documatic	InterSystems class reference documentation.	**%Development**
/csp/sys	General Portal access.	
/csp/sys/exp	Data Management options in the Portal.	**%Development**
/csp/sys/mgr	Configuration and Licensing options in the Portal.	**%Admin_Manage**
/csp/sys/op	Operations options in the Portal.	**%Admin_Operate**
/csp/sys/sec	Security Management and Encryption options in the Portal.	**%Admin_Secure**
/isc/studio/rules	Mapping to the CSP rules files.	
/isc/studio/templates	Mapping to system-defined Studio template files.	**%Development**

For more information on the /isc/studio/rules, application, see the chapter "Developing Custom Tags" in *Using Caché Server Pages (CSP)*. For more information on the /isc/studio/templates application, see the chapter "Using Studio Templates" in *Using Studio*.

9

Auditing

Logging certain key events in a secure audit database is a major aspect of Caché security. Caché allows you to monitor events and add entries to the audit database when those events occur. These events can be within Caché itself or part of an application. The knowledge that all activities are being monitored and that all logs can be reviewed is often an effective deterrent.

This chapter provides information on various topics:

- Basic Auditing Concepts
- About Audit Events
- Managing Auditing and the Audit Database
- Other Auditing Issues

9.1 Basic Auditing Concepts

Caché allows you to enable or disable auditing for the entire Caché instance. When auditing is enabled, Caché logs all requested events. Auditable events fall into two categories:

- System events — Caché system events that are only logged if they are explicitly enabled.
- User events — Application events, which are only logged if they are explicitly enabled.

Caché system events are built-in events that monitor actions within Caché, such as start-up, shutdown, logins, and so on; system events also monitor security-related events, such as changes to security or audit settings.

Caché does not automatically audit database activity, such as inserts, updates, or deletes for a table, because this kind of activity typically generates so many audit entries as to be useless — or even counterproductive — due to the performance impact on the system. For example, if a medical records application were to log all access to patient medical information, then one such access event might result in hundreds or thousands of database accesses. It is much more efficient to have the application create a single audit entry, rather than have the database manager generate thousands.

9.1.1 Enabling or Disabling Auditing

In the **Auditing** menu (**System Administration** > **Security** > **Auditing**), there are selections to enable and disable auditing. If the **Enable Auditing** choice is available, this means that auditing is disabled; if the **Disable Auditing** choice is available, this means that auditing is enabled. Caché auditing is disabled by default for minimal-security installations; it is enabled by default for normal and locked-down installations.

Enabling Auditing

To turn on auditing, on the **Auditing** menu (**System Administration** > **Security** > **Auditing**), select the **Enable Auditing** choice.

Disabling Auditing

To turn off auditing, on the **Auditing** menu (**System Administration** > **Security** > **Auditing**), select the **Disable Auditing** choice.

If you enable (turn on) auditing, then Caché audits:

- All system events that are enabled
- All user events that are enabled

9.2 About Audit Events

The following sections describe different aspects of audit events:

Note: This section describes how to manage audit events with the Management Portal. To manage audit events programmatically, use the Security.Events class.

9.2.1 Elements of an Audit Event

Audit information is available in the CACHEAUDIT database. New entries are added to the end of the log. When you view the audit log, you see the following elements for each entry:

Time (also called UTCTimestamp)

UTC date/time when the event was logged.

Event Source*

The component of the Caché instance that is the source of the event. For Caché events, this is always "%System". For user-defined events, the name can be any string that includes alphanumeric characters or punctuation, except for colons and commas; it can begin with any of these characters except for the percent sign. This can be up to 64 bytes.

Event Type*

Categorizing information for the event. This string can include any alphanumeric characters or punctuation, except for colons and commas; it can begin with any of these characters except for the percent sign. This can be up to 64 bytes.

Event* (also called Event Name)

Identifier of the event being logged. This string can include any alphanumeric characters or punctuation, except for colons and commas; it can begin with any of these characters except for the percent sign. This can be up to 64 bytes.

PID (also known as a Process ID)

Operating system ID of the Caché process that logged the event. Caché uses the OS PID in its native form.

CSP Session (search results only)

The session ID, if there is one, of the CSP session that caused the event.

User (also called Username)

Value of *$USERNAME* for the process that logged the event.

Description

A short field which can be used by applications to summarize the audit event. This field is intended for display/explanation purposes only. The combination of EventSource, EventType, and Event uniquely define a particular kind of audit event. The Description is a user readable explanation. This can be up to 128 characters.

*Each different kind of event is uniquely identified by the combination of its EventSource, its EventType, and the Event itself.

When you click **Details**, you see some of the same elements and the following additional elements:

Timestamp

Date/time when the event was logged, in local time.

JobId

ID of the job.

IP Address

IP address of client associated with the process that logged the event.

Executable

The client application associated with the process that logged the event, if there is one.

System ID

The machine and Caché instance that logged the event. For example, for the machine *MyMachine* and the instance *MyInstance*, the system ID is `MyMachine:MyInstance`.

Index

The index entry in the data structure containing the audit log.

Roles

For all events except LoginFailure, the value of *$ROLES* for the process that logged the event. For LoginFailure, a value of " ", as the user is not logged in.

Namespace

The namespace that was current when the event was logged.

Routine

The routine or subroutine that was running when the event was logged.

User Info

User-defined information about the process, added programmatically via the %SYS.ProcessQuery interface.

O/S Username

Username given to the process by the operating system. When displayed, this is truncated to 16 characters.

This is the actual operating system username only for UNIX® systems.

For Windows:

- For a console process, this is the operating system username.
- For Telnet, this is the *$USERNAME* of the process.
- For client connections, this is the operating system username of the client.

Status

The value of any %Status object that was audited.

Event Data

A memo field where applications can store up to 3632952 bytes of data associated with the audit event. For example, it can contain a set of application values at the time of the event or can summarize the old and new states of a record or field.

9.2.2 About System Audit Events

System audit events are predefined events that are available for auditing by default. General information about them appears in the table on the **System Audit Events** page (**System Administration** > **Security** > **Auditing** > **Configure System Events**), where the columns are:

- Event Name — The Event Source (which is always %SYSTEM), Event Type, and Event proper, all together and concatenated with slashes (“/”).
- Enabled — Whether or not the event is enabled (turned on) for auditing.
- Total — The number of events of this type that have occurred since the last startup of Caché.
- Written — The number of events of this type that have been written to the audit log since the last startup of Caché. This number may differ from the total occurrences.
- Reset — Allows you to clear the audit log for this event reset its counter to zero. For more information on counters, see “About Counters.”
- Change Status — Allows you to enable or disable the event.

They monitor events within the Caché system and are distinguishable by their EventSource value of %SYSTEM:

Table 9–1: System Audit Events

Event Type and Event	Occurs When	EventData Contents	Default Status
%DirectMode/ DirectMode	Any command is executed in direct mode.	Text of command.	Off

Event Type and Event	Occurs When	EventData Contents	Default Status
%Login/ Login	A user successfully logs in.		Off
%Login/ LoginFailure	A login attempt fails.	Username.	Varies*
%Login/ Logout	A user logs out.		Off
%Login/ Terminate	A process terminates abnormally.	Varies, as does the Description field's content; see below.	Off
%SMPExplorer/ Change	Data is altered using the Portal, such as by creating, editing, deleting, compiling, dropping, replacing, or purging classes or tables.	Varies, as does the Description field, depending on the action taken. Includes relevant content such as the compile flags or the schema and table being dropped.	Off
%SMPExplorer/ ExecuteQuery	A query is executed using on the Portal's SQL page.	The syntax of the executed query.	Off
%SMPExplorer/ Export	Data is exported through the Portal.	The options selected for data export.	Off
%SMPExplorer/ Import	Data is imported through the Portal.	The options selected for data import.	Off
%SMPExplorer/ ViewContents	Data is viewed through the Portal.	The filters that determined what data was viewed. The Description field specifies what was viewed, such as a list of classes, an individual global, or process information.	Off
%SQL/ DynamicStatement	A dynamic SQL call is executed.	The statement text and the values of any host-variable arguments passed to it. If the total length of the statement and its parameters exceeds 3,632,952 characters, the event data is truncated.	Off
%SQL/ EmbeddedStatement	An embedded SQL call is executed.	The statement text and the values of any host-variable arguments passed to it. If the total length of the statement and its parameters exceeds 3,632,952 characters, the event data is truncated.	Off

Event Type and Event	Occurs When	EventData Contents	Default Status
%SQL/ XDBCStatement	A remote SQL call is executed using ODBC or JDBC.	The statement text and the values of any host-variable arguments passed to it. If the total length of the statement and its parameters exceeds 3,632,952 characters, the event data is truncated.	Off
%Security/ ApplicationChange	An application definition is created, changed, or deleted.	Action (create new, modify, or delete), old and new application data.	On
%Security/ AuditChange	Auditing is stopped or started, entries are erased or deleted, or the list of events being audited is changed.	Action (stop, start, erase, delete, or specify), old and new audit settings.	On
%Security/ AuditReport	Any standard audit report is run.	Identification of audit report.	On
%Security/ DBEncChange	There is a change related to database or data-element encryption.	Varies, as does the Description field's content. See below.	On
%Security/ DomainChange	A domain definition is created, changed, or deleted.	Action (new, modify, delete), old and new domain data.	On
%Security/ LoginRuleChange	This event is not currently in use. Even when it is enabled (On), it does not appear in the audit log.		On
%Security/ Protect	A process generates a security protection error.	Error.	Off
%Security/ ResourceChange	A resource definition is created, changed, or deleted.	Action (new, modify, or delete), old and new resource data.	On
%Security/ RoleChange	A role definition is created, changed, or deleted.	Action (create new, modify, or delete), old and new role data.	On
%Security/ SSLConfigChange	An SSL/TLS configuration's settings are changed.	The changed fields with old and new values.	On
%Security/ ServiceChange	A service's security settings are changed.	Old and new service security settings.	On
%Security/ SystemChange	System security settings are changed.	Old and new security settings.	On

Event Type and Event	Occurs When	EventData Contents	Default Status
%Security/ UserChange	A user definition is created, changed, or deleted.	Action (create new, modify, or delete), old and new user data.	On
%System/ AuditRecordLost	An audit entry has not been added to the audit database due to resource limitations that constrain the audit system (such as disk or database full).	None.	On
%System/ ConfigurationChange	Caché successfully starts with a configuration different than the previous start, a new configuration is activated while Caché is running, or a lock is deleted through the Portal or through the ^LOCKTAB utility.	Username for the user who made the change; previous and new values of the changed element. For deleted locks, information about which lock was deleted.	On
%System/ DatabaseChange	There are changes to database properties. See below.	Details about the particular change. See below.	On
%System/ JournalChange	Journaling is started or stopped for a database or process.	When journaling is started, the name of the database and its maximum size; when journaling is stopped, none.	On
%System/ OSCommand	An operating-system command is issued from within the system, such as through a call to the $ZF(-100) function.	The operating system command that was invoked; the directory in which it was invoked; and any flags associated with the command.	On
%System/ RoutineChange	A method or routine is compiled or deleted on the local instance. For more details, see below.	No content, though the Description field depends on the change itself; see below.	Off
%System/ Start	The system starts.	Indication of whether recovery was performed.	On
%System/ Stop	Caché is shut down.		On
%System/ SuspendResume	A process is suspended or resumed.	The process ID of the process.	Off
%System/ UserEventOverflow	An application attempts to log an undefined event.	The name of the event that the application attempted to log.	On

*The LoginFailure event is off by default for minimal-security installations; it is on by default for normal and locked-down installations.

Important: If auditing is enabled, then all enabled events are audited.

9.2.2.1 About the %System/%Login/Logout and %System/%Login/Terminate Events

A process generates a %System/%Login/Logout event if the process ends because of:

- A HALT command
- Exiting application mode because of a QUIT command
- Executing the **Terminate** method of the SYS.Process class to terminate itself (which is the same as executing **HALT**).

A process generates a %System/%Login/Terminate event if the process exits for any other reason, including:

- The user closes the Terminal window, resulting in a Terminal disconnect. If the process is in application mode, the Description field of the audit record includes the statement "^routinename client disconnect" (where *routinename* is the first routine that the process ran); if the process is in programmer mode, the Description field includes the statement "Programmer mode disconnect."
- A Terminal session is ended by an action in another process, including ^RESJOB, ^JOBEXAM, or the Management Portal. If the process is in application mode, the Description field of the audit record includes the statement "^routine-name client disconnect" (where *routinename* is the first routine that the process ran) ; if the process is in programmer mode, the Description field includes the statement "Programmer mode disconnect." Note that the event data will contain the pid of the process which terminated them.
- A core dump or process exception. When a process gets a core dump or exception, it is too late for it to write to the audit file. Therefore, when the clean daemon runs to clean up the state of the process, it writes an audit record to the log with a description "Pid <process nunber> Cleaned".
- A TCP Client disconnect. When a process detects that a client has disconnected, this results in an audit record with a Description field which contains the name of the executable that disconnected, such as "<client application> client disconnect".

9.2.2.2 About the %System/%Security/DBEncChange Event

A process generates a %System/%Security/DBEncChange event because of:

- Encryption key activation
- Encryption key deactivation
- Encryption key and key file creation
- Encryption key file modification
- Encryption settings modification

The EventData includes the encryption key's ID and key file when these are relevant to the event.

9.2.2.3 About the %System/%System/DatabaseChange Event

A process generates a %System/%System/DatabaseChange because of any of the following changes to a database:

- Creation
- Modification
- Mounting
- Dismounting

- Compaction
- Truncation
- Global compaction
- Defragmentation

For creation and modification, changes to the following properties cause auditing events (which are included in the event data):

- BlockSize (Create only)
- ClusterMountMode (Cluster systems only)
- ExpansionSize
- GlobalJournalState
- MaxSize
- NewGlobalCollation
- NewGlobalGrowthBlock
- NewGlobalIsKeep
- NewGlobalPointerBlock
- ReadOnly
- ResourceName
- Size

For mounting and dismounting, the event data records the database that was mounted or dismounted. For compaction, truncation, global compaction, and defragmentation, the event data includes include the parameters that the user selected.

9.2.2.4 About the %System/%System/RoutineChange Event

A process generates a %System/%System/RoutineChange event because a routine has been compiled or deleted. When enabled, this event causes a record to be written to the audit log whenever a routine or class is compiled. The Description field of the audit record includes the database directory where the modification took place, what routine or class was modified, and the word "Deleted" if the routine was deleted.

Caché audits events on the local server but not for associated instances. For example, if one instance of Caché is an application server that is associated with another instance that is a database server, creating and compiling a new routine on the application server is not audited on the database server, even if the RoutineChange audit event is enabled on the database server. To create a comprehensive list of all changes on all associated instances, enable the relevant events on all the instances and combine their audit logs.

9.2.3 Enabling and Disabling System Events

To enable or disable system events:

1. From the Management Portal home page, go to the **System Audit Events** page (**System Administration** > **Security** > **Auditing** > **Configure System Events**).
2. On the **Configure System Audit Events** page, locate the event that you wish to enable or disable and select **Change Status** from the right-most column of the table. This changes the Enabled status from No to Yes, or vice versa.

9.2.4 About User Events

In addition to system events, Caché allows you to create custom events that can be added to the audit database through your application. All currently defined events are listed on the **User-Defined Audit Events** page (**System Administration** > **Security** > **Auditing** > **Configure User Events**).

9.2.4.1 Creating a User-defined Event

For Caché to audit a user-defined event, it must be added to the list of events and then enabled. The procedure is:

1. In the Management Portal, go to the **User-Defined Audit Events** page (**System Administration** > **Security** > **Auditing** > **Configure User Events**).
2. Click **Create New Event**. This displays the **Edit Audit Event** page.
3. On this page, enter values in the **Event Source**, **Event Type**, **Event Name**, and **Description** fields where these components have the purposes described in " Elements of an Audit Log Entry."
4. By default, the **Enabled** check box on this page is selected. Click it to disable the event.
5. Click the page's **Save** button to create the event.
6. Make sure that auditing is enabled.
7. Once the event is defined and auditing is enabled, you can add the event to the audit log by executing the following command:

   ```
   Do $SYSTEM.Security.Audit(EventSource,EventType,Event,EventData,Description)
   ```

   using the *EventSource*, *EventType*, *Event*, and *EventData* values that you defined in the Portal. For more details, see the section " Adding an Entry to the Audit Log."

9.2.4.2 Adding an Entry to the Audit Log

Applications can add their own entries to the audit log with the **$SYSTEM.Security.Audit** function:

```
Do $SYSTEM.Security.Audit(EventSource,EventType,Event,EventData,Description)
```

where *EventSource*, *EventType*, *Event*, *EventData*, and *Description* are as described in the section "Elements of an Audit Log Entry." Both the *EventData* and *Description* arguments can hold variables or literal values (where strings must appear in quotation marks). Caché provides all other elements of the log item automatically.

The content of *EventData* can span multiple lines. Its content is processed in a manner similar to the argument of the ObjectScript Write command, so it uses the following form:

```
"Line 1"_$Char(13,10)_"Line 2"
```

In this case, the content listed in the Audit Detail is displayed as "Line 1", then `$Char(13,10)` is a carriage return and line feed, then there is "Line 2".

For example, a medical records application from XYZ Software Company might use values such as:

```
$SYSTEM.Security.Audit(
    "XYZ Software",
    "Medical Record",
    "Patient Record Access",
    765432,
    "Access to medical record for patient 765432"
    )
```

Note that the application uses the EventData element to record the ID of the patient whose record was accessed.

Further, if there is an "XYZ Software/Record Update/Modify Assignment" event defined and enabled, then the following code changes the value of a user-selected element of a list and notes the change in the audit database:

```
For i=1:1:10 {
    Kill fVal(i)
    Set fVal(i) = i * i
}

Read "Which field to change? ",fNum,!
Read "What is the new value? ",newVal,!
Set oldVal = fVal(fNum)
Set fVal(fNum) = newVal
Set Data = "Changed field " _ fNum _ " from " _ oldVal _ " to "_ newVal _ "."
Set Description = "Record changed by user with an application manager role"
Do $SYSTEM.Security.Audit(
    "XYZ Software",
    "Record Update",
    "Modify Assignment",
    Data,
    Description
)
Write "Field changed; change noted in audit database."
```

Audit returns 1 or 0 to indicate that the addition succeeded or failed.

No privilege is required to add an entry to the audit log.

9.2.4.3 Deleting User Events

If you delete a user event, it is no longer available as part of the Caché instance for auditing. To delete a user event:

1. From the Management Portal home page, go to the **User-Defined Audit Events** page (**System Administration** > **Security** > **Auditing** > **Configure User Events**).
2. On this page, locate the event that you wish to enable or disable and select **Delete** from the column near the right-hand part of the table.
3. When prompted, confirm that you wish to delete the event.

9.3 Managing Auditing and the Audit Database

When events are logged, they are visible in the audit database, CACHEAUDIT. The audit database also contains general information, including the name of the server, the name of the Caché configuration, when the log was started, and when the log was closed.

The following actions are available for managing the audit log:

- Viewing the Audit Database
- Copying, Exporting, and Purging the Audit Database
- Encrypting the Audit Database
- General Management Functions

9.3.1 Viewing the Audit Database

To view the audit database, select **View Audit Database** from the **Auditing** menu. This displays the **View Audit Database** page (**System Administration** > **Security** > **Auditing** > **View Audit Database**). This page allows you to view the audit database and refine a search based on the following fields:

- Event Source — The component of the Caché instance that is the source of the event, as described in "Elements of an Audit Event." Clicking the button to the right of the field displays a list of values in use. The asterisk ("*") chooses all values; no other wildcards are supported.
- Event Type — Any categorizing information for the event, as described in "Elements of an Audit Event." Clicking the button to the right of the field displays a list of values in use. The asterisk ("*") chooses all values; no other wildcards are supported.
- Event Name (also called Event) — The identifier of the event being logged, as described in "Elements of an Audit Event." Clicking the button to the right of the field displays a list of values in use. The asterisk ("*") chooses all values; no other wildcards are supported.
- System IDs — An identifier for the instance of Caché that appears in each audit log entry. This identifier is of the form *machine_name*:*instance_name*, so that if you have an instance of Caché called "MyCache" running on a machine called "MyMachine", then its System ID is "MyMachine:MyCache". To search for multiple system IDs, provide a comma-separated list. The asterisk ("*") chooses all values; no other wildcards are supported.
- PIDs — The operating-system ID of the Caché process that logged the event, as described in "Elements of an Audit Event."
- Begin Date & Time — The date and time for the first event to be displayed (midnight at the beginning of the current day, by default).
- End Date & Time — The date and time for the last (most recent) event to be displayed (the current time, by default)
- Sort by — Orders the results by:
  - Reverse Date — From most recent to least recent
  - Events — By event name, in alphabetical order
  - Users — By username, in alphabetical order
  - PID — By operating-system process ID, from lowest to highest
- Maximum Rows — The maximum number of rows to display in a listing of the audit log (up to 10,000).
- Color by — The field (if any) that determines how the search results are colored. Fields that can determine search result coloring are Description, Event, Event Source, Event Type, PID, Time Stamp, and Username.
- Users — The user who has caused the event. The asterisk ("*") chooses events causes by all users; no other wildcards are supported.

9.3.2 Copying, Exporting, and Purging the Audit Database

The audit log is stored in the %SYS.Audit table in the %SYS namespace; all audit data is mapped to the CACHEAUDIT database and protected by the **`%DB_CACHEAUDIT`** resource. By default, the %Manager role holds the **`Read`** permission on this resource and no role holds the **`Write`** permission.

The audit log database is managed with the same tools other Caché databases. For example, you can use the Management Portal to specify its initial size, growth increment, and location; while the audit log database does not have a specified maximum size, it is constrained by disk space and other such factors.

The Management Portal allows you to perform special management operations on the audit database:

- Copying — You can copy entries for one or more days to a specified namespace.
- Exporting — You can export entries for one or more days from the log to a file.
- Purging — You can remove entries for one or more days from the log.

Note: All these operations act on all entries for one or more days. There are no operations for particular entries.

9.3.2.1 Copying the Audit Database

Caché allows you to copy all or part of an audit database to a namespace other than CACHEAUDIT. To do this:

1. From the Management Portal home page, go to the **Copy Audit Log** page (**System Administration** > **Security** > **Auditing** > **Copy Audit Log**).
2. On the **Copy Audit Log** page, first select either:
   - **Copy all items from the audit log**
   - **Copy items that are older than this many days from audit log** In the field here, enter a number of days; any item older than this is copied to the new namespace.
3. Next, use the drop-down menu to choose the namespace where you wish to copy the audit entries.
4. If you wish to delete the audit items after they are copied, select the check box with that choice.
5. Click **OK** to copy the entries.

Caché places the selected audit log entries in the ^*CacheAuditD* global in the selected namespace. To view this data:

1. From the Management Portal home page, go to the **Globals** page (**System Explorer** > **Globals**).
2. From the **Globals** page, select the following items in the following order:
   a. The **Databases** radio button from the upper left area of the page.
   b. The name of the database holding the copied audit log entries.
   c. The **System** check box that appears above the list of globals.

   This displays a list of globals in the database, including ^*CacheAuditD*. Globals are listed without the preceding "^" character that is needed to manipulate them programmatically or in the Terminal.

   Note: Clicking **View Globals** on this page refreshes the page but unchecks in the **System** check box, thereby making ^*CacheAuditD* unavailable.
3. Click **Data** from the CacheAuditD line to display detailed information on the audit log entries.

Once you have copied audit data to another namespace, you can use the queries of the %SYS.Audit class to look at that data.

9.3.2.2 Exporting the Audit Database

Caché allows you to export all or part of an audit database. To do this:

1. From the Management Portal home page, go to the **Export Audit Log** page (**System Administration** > **Security** > **Auditing** > **Export Audit Log**).
2. On the **Export Audit Log** page, first select either:
   - **Export all items from the audit log**
   - **Export items that are older than this many days from audit log** In the field here, enter a number of days; any item older than this is exported to the new namespace.

3. Next, in the **Export to file** field, enter the path of the file where you wish to export the audit entries. If you do not enter a full path, the root for the path provided is cachesys/Mgr/ where *cachesys*is the default name of the installation directory.
4. If you wish to delete the audit items after they are exported, select the check box with that choice.
5. Click **OK** to export the entries.

9.3.2.3 Purging the Audit Database

Caché allows you to purge all or part of a database.

Important: Purging the database is not a reversible action — purged items are permanently removed. You cannot restore items to the audit database once you have purged them.

To do this:

1. From the Management Portal home page, go to the **Purge Audit Log** page (**System Administration** > **Security** > **Auditing** > **Purge Audit Log**).
2. On the **Purge Audit Log** page, first select either:
   - **Purge all items from the audit log**
   - **Purge items that are older than this many days from audit log** In the field here, enter a number of days; any item older than this is purged.
3. Click **OK** to purge the entries.

9.3.3 Encrypting the Audit Database

Caché allows you to encrypt the database that holds the audit log. This is described in the section " Configuring Caché Database Encryption Startup Settings" in the chapter "Managed Key Encryption" in the *Caché Security Administration Guide*.

9.3.4 General Management Functions

Because the audit log is stored in a table, you can manage it with standard Caché system management tools and techniques:

- Journaling is always turned on for it.
- You can use standard Caché commands to read it. In addition, its contents are accessible via standard SQL and you can use any standard SQL tool to work with it.
- You can back it up using standard Caché database backup facilities.
- If it becomes full, a <FILEFULL> error occurs and is handled in the same way as for any other Caché database. To avoid this situation, see the "Maintaining the Size of the Audit Database" section.

Note: All access is subject to standard security restrictions at the database and/or namespace levels, or through SQL for table-based activity.

The %SYS.Audit table in the %SYS namespace holds the audit log. All audit data is mapped to the CACHEAUDIT database. (You can also copy audit data to any other database using the functionality described in the section "Copying the Audit Database"; you can then use the %SYS.Audit class, which is available in every namespace, to query the audit log.)

9.3.4.1 Maintaining the Size of the Audit Database

As Caché runs, it writes to the audit log. Without intervention, this will eventually fill the audit database. If the audit database becomes full, Caché halts. To properly store audit information and also prevent any downtime, you should regularly export and save the contents of the audit database and then purge its contents.

To do this:

1. Export the contents of the audit database as described in the "Exporting the Audit Database" section.

   Note: InterSystems recommends that you export all entries from the database.

2. Check that the exported contents of the audit database are valid.

   Important: InterSystems recommends that you confirm that this data is valid, as purging the data is a non-reversible action.

3. Purge old entries from the existing database as described in "Purging the Audit Database".

   Important: InterSystems recommends that you purge all entries except those of the last day, which ensures that there is an overlap in the different groups of saved entries.

CAUTION: If the audit database becomes full, Caché does not record audit entries for actions that cause audit events. Further, in a forensic context, the existence of only a single AuditRecordLost audit entry indicates that *at least* one record was lost.

9.4 Other Auditing Issues

This section covers the following topics:

- Freezing Caché If There Can Be No Audit Log Writes
- About Counters

9.4.1 Freezing Caché If There Can Be No Audit Log Writes

During operations of Caché, it may become impossible to write to the audit database. This can happen due to a filled disk, a failed network connection, or some other reason. If this occurs, Caché can then work in either of two different modes:

- The default behavior is to generate an error when there is a problem writing to the Audit log.
- Alternatively, a switch can be set so that the system freezes if there is a problem writing to the Audit log.

To set this switch, use the ^SECURITY routine:

1. In the Terminal, go to the %SYS namespace:

   ```
   > zn "%SYS"
   ```

2. Start ^SECURITY:

   ```
   > Do ^SECURITY
   ```

3. Within ^SECURITY, choose option 6 (Auditing Setup); within Auditing Setup, choose option 1 (Enable auditing).

4. When it displays the prompt: `Freeze system on audit database error?` Enter "Yes" or "Y". Confirm any changes when prompted.

This establishes the behavior of freezing the system.

For a disk full error, the way to recover from this is to force down the system, free up space on the audit disk, then restart. For an error caused by database corruption, the audit database must be moved or deleted, and a new one created or copied into its place. Note that a restart of the system will not be enough to clear the error, since the restart may write audit records, and cause the system to freeze again.

For example, with the default behavior, if the disk containing the audit database fills up, then the attempt to write to the audit log will generate a <FILEFULL> error (disk full). The audit record is not written to the audit log, and is therefore lost. When the problem is resolved, an entry is written into the audit log that lists how many audit events were lost.

When this switch is set, and a write error occurs when writing to the audit log, the process which was trying to write to the audit log receives an error (such as <FILEFULL>). The process then writes the error message to the cconsole.log, and then freezes the system.

9.4.2 About Counters

To facilitate security monitoring, Caché keeps a counter for each audit event type and makes these counters available via the Caché monitoring interface. These counters are maintained even if auditing is not enabled. As an example, a site might monitor the LoginFailure event counter, to help detect break-in attempts.

Note: Audit counters are reset when the instances is restarted.

9.4.2.1 Resetting the Counters for a System Event

To reset the counters for a system event:

1. From the Management Portal home page, go to the **System Audit Events** page (**System Administration** > **Security** > **Auditing** > **Configure System Events**).
2. On this page, locate the event that you wish to enable or disable and select **Reset** from the column near the right-hand part of the table.
3. When prompted, click **OK**. This resets both the **Total** and **Written** counters for the event.

9.4.2.2 Resetting the Counters for a User Event

To reset the counters for a user event:

1. From the Management Portal home page, go to the **User-Defined Audit Events** page (**System Administration** > **Security** > **Auditing** > **Configure User Events**).
2. On this page, locate the event that you wish to enable or disable and select **Reset** from the column near the right-hand part of the table. This resets both the **Total** and **Written** counters for the event.
3. When prompted, click **OK**. This resets both the **Total** and **Written** counters for the event.

10

Managed Key Encryption

Caché includes support for managed key encryption, a suite of technologies that protects data at rest. These technologies are:

- Block-level database encryption, also known simply as database encryption — A set of tools to allow creation and management of databases in which all the data is encrypted. Such databases are managed through the Management Portal.
- Data element encryption for applications, also known simply as data element encryption — A programmatic interface so that applications can include code to encrypt and decrypt individual data elements (such as particular class properties) as they are stored to and retrieved from disk.
- Encryption key management — A set of tools in the Management Portal for creating and managing data-encryption keys and for managing key files. Both database encryption and data element encryption use key files to support their functionality.

Each Caché instance can simultaneously have up to four data-encryption keys activated for database encryption and up to four data-encryption keys activated for data element encryption (*activating* a key makes it available for encryption and decryption operations); the database-encryption keys can be the same keys or other keys as the data-element encryption keys.

When you create a data-encryption key, an encrypted copy of it is stored in a key file and must be decrypted to be used. It is imperative that you store the key file in a secure location where it cannot be damaged in any way. Each key file can hold up to four keys.

Caché uses AES (the Advanced Encryption Standard) to perform its encryption and decryption when Caché writes to or reads from disk. For databases, Caché writes and reads in fixed-length blocks, and the entire database is encrypted, except for the single label block. This includes the data itself, indices, bitmaps, pointers, allocation maps, and incremental backup maps. For data elements, only the specified data is encrypted, and a unique identifier for the encryption key is included with the encrypted data on disk.

Encryption and decryption have been optimized, and their effects are both deterministic and small for any Caché platform. (This chapter also includes a section that addresses how the Caché database encryption facilities affect functionality related to but separate from databases themselves.)

WARNING! The loss of or damage to all copies of a key file will prevent encrypted data — whether data elements or databases — from being decrypted.

Topics in this chapter include:

- Managing Keys and Key Files
- Recommended Policies for Managing Keys and Key Files

- Using Encrypted Databases
- Using Data Element Encryption
- Emergency Situations
- Other Information

10.1 Managing Keys and Key Files

A *key*, short for *data-encryption key*, is a 128–, 196–, or 256–bit bit string that is used with a cryptographic algorithm to reversibly encrypt or decrypt data. Each key has a unique identifier (known as a GUID), which Caché displays as part of key management activities. Key management is the set of activities associated with creating encryption keys, activating keys, and deactivating keys.

A *key file* is a file that holds encrypted copies of one or more data-encryption keys. Key file management is the set of activities associated with key files themselves, such as adding administrators to or removing administrators from key files.

Within a key file, each key is available to every administrator in the key file. (In this chapter, the term *administrator* refers to a key administrator, not a Caché administrator.) All keys are stored in an encrypted form along with administrator information; each data-encryption key is individually encrypted using a *master key*. For each administrator in the key file, there is a unique, encrypted copy of the master key, which is encrypted using a *key-encryption key* — where each key-encryption key is derived from an individual key administrator's password. Encryption tasks require an activated key, and Caché requires an administrator username and password to decrypt the key so that the key can be activated.

Topics in this section involve tasks related to key files, keys, and administrators. All tasks are performed in the Management Portal. Some tasks are for managing keys and key files, even if no keys in the key file have been activated on an instance of Caché; other tasks are related to an instance of Caché, activating keys for it, and so on. There are also other miscellaneous tasks.

For managing keys and key files (even if no keys in the key file have been activated on an instance of Caché), tasks are:

- Creating a Key File
- Adding a Key to a Key File
- Deleting a Key from a Key File
- Adding an Administrator to a Key File
- Deleting an Administrator from a Key File

Related to an instance of Caché and encryption keys for it, tasks are:

- Activating a Database Encryption Key
- Deactivating a Database Encryption Key
- Specifying the Default Encryption Key or Journaling Encryption Key for an Instance
- Activating a Data Element Encryption Key
- Deactivating a Data Element Encryption Key

Other miscellaneous tasks are:

- Testing for a Valid Administrator Username-Password Pair
- Managing Keys and Key Files with Multiple-Instance Technologies

Note: If you wish to configure encryption for journal files or the CACHETEMP and CACHE databases, this is part of Caché startup configuration. See the section "Configuring Caché Database Encryption Startup Settings" for details.

A Note about Encryption Key File Formats

Beginning in version 2015.1, Caché introduced support for encryption files that hold multiple keys (up to four), which are *version 2.0 key files*. (The original key file format, version 1.0, holds only a single key.) Version 2.0 key files are the preferred format, and are the default when creating a key file. To convert a version 1.0 key file to version 2.0, use the ^EncryptionKey character-based utility.

Note: If an instance uses multiple keys at startup time (such as with journal files, the audit database, and other databases), then those keys must all be in a single version 2.0 key file. This allows them all to be available when the instance starts.

10.1.1 Creating a Key File

When you create an encryption key file, it contains one key. To create an encryption key file and its initial key, the procedure is:

1. From the Management Portal home page, go to the **Create Encryption Key File** page (**System Administration** > **Encryption** > **Create New Encryption Key File**).
2. On the **Create Encryption Key File** page, specify the following values:
   - **Key File** — The name of the file where the encryption key is stored; this can be an absolute or relative path name.

     If you enter an absolute file name, the key file is placed in the specified directory on the specified drive; if you enter a relative file name, the key file is placed in the manager's directory for the Caché instance (which is below the Caché installation directory — that is, in <cache-install-dir>/mgr/). Also, no file suffix is appended to the file name, so that the file MyKey is saved simply with that file name. You can also use the **Browse** button to the right of this field to choose the directory where Caché will create the key file. (If you provide the name of an existing file, Caché will not overwrite it and the save will fail.)

     WARNING! Any key stored in <cache-install-dir>/Mgr/Temp is deleted when Caché next reboots — *never* store a key in <cache-install-dir>/Mgr/Temp.

   - **Administrator Name** — The name of an administrator who can activate the key. There must be at least one administrator.

     Because the database encryption functionality exists independent of Caché security, this name need not match any user names that are part of Caché security. By default, the initial administrator name value is the current username. The administrator name cannot include Unicode characters.

   - **Password** — A password for this user.

     Because the database encryption functionality exists independent of Caché security, this password need not match the password that a user has for Caché security. Note that this password is not stored anywhere on disk; it is the responsibility of the administrator to ensure that this information is not lost.

     InterSystems suggests that this password follow the administrator password strength guidelines. If someone can successfully guess a valid password, the password policy is too weak. Also, this password cannot include Unicode characters.

   - **Confirm Password** — The password for this user entered again to confirm its value.
   - **Cipher Security Level** — The length of the key, where choices are 128–bit, 192–bit, and 256–bit.
   - **Key File Format** — Either

  - **1.0 - Single key only** — the original data-encryption key format
  - **2.0 - Single or multiple keys** — (the default) the current data-encryption key format

  Note: InterSystems recommends the use of the 2.0 key format.

- **Key Description** — For keys in 2.0 key format only, text to describe the key that is initially created and stored in the key file. This text appears in the **Description** column of the **Encryption Keys Defined in Key File** table.

3. Click **Save** at the top of the page to save the key file to disk.
4. Having just created a key, follow the instructions in the section "Protection from Accidental Loss of Access to Encrypted Data" to create and store a backup copy of the newly updated key file.
5. Refer to the section "Protection from Unauthorized Access to Encrypted Data" for details about measures to prevent currently or formerly privileged users from gaining unsanctioned access to encrypted data.

This creates a key file with a single database-encryption key in it and with a single administrator. The page displays ID for the key, which is a string such as `9158980E-AE52-4E12-82FD-AA5A2909D029`. The key ID is a unique identifier for the key which Caché displays here and on other pages. It provides a means for you to keep track of the key, regardless of its location. This is important because, once you save the key file, you can move it anywhere you choose; this means that Caché cannot track it by its location.

The key is encrypted using the master encryption key, and there is a single copy of master encryption key, which is encrypted using the administrator's key-encryption key (KEK). You can add additional keys to the key file according to the instructions in the section "Creating a Key." You can add administrators to the key file according to the instructions in the section "Adding an Administrator to a Key File."

WARNING! InterSystems strongly recommends that you create and store a backup copy of the key file. Each time you create a database-encryption key, it is a unique key that cannot be re-created. Using the same administrator and password for a new key still results in the creation of a different and unique key. If an unactivated key is lost and cannot be recovered, the encrypted database that it protected will be unreadable and its data will be *permanently lost*.

10.1.2 Adding a Key to a Key File

To create a key, you can either:

- Create a key file. This causes Caché to create a key and place it in the file. To create a key file, see the section "Creating a Key File."
- Add a key to an existing key file, as described in this section.

To add a key to an existing key file, the procedure is:

1. From the Management Portal home page, go to the **Manage Encryption Key File** page (**System Administration** > **Encryption** > **Manage Encryption Key File**).
2. On the **Manage Encryption Key File** page, in the **Key File** field, enter the name of the key file to which you want to add and store the key; click **OK**. This displays information about that key file; at the bottom of the shaded area, the **Encryption Keys Defined in Key File** table displays a list of the one to four keys in the key file. If there are three or fewer keys in the file, you can create a new key and add it to the file.
3. Click the **Add** button below the **Encryption Keys Defined in Key File** table to add a key to the key file. This displays the **Add a New Encryption Key** screen.
4. In the **Add a New Encryption Key** screen, enter values in the following fields:

- **Existing Administrator Name** — The name of an administrator associated with the key file. (Administrators associated with the file appear in the **Administrators Defined in Key File** table on the **Manage Encryption Key File** page.)
- **Existing Administrator Password** — This administrator's password.
- **Description** — Text to describe the key. This text appears in the **Description** column of the **Encryption Keys Defined in Key File** table.

5. Click **OK** to save the key to the key file. This displays information about it in the **Encryption Keys Defined in Key File** table, including its ID, which is a string such as `9158980E-AE52-4E12-82FD-AA5A2909D029`. (The key ID is a unique identifier for the key which Caché displays here and on other pages. It provides a means for you to keep track of the key, regardless of its location. This is important because, once you save the key file, you can move it anywhere you choose; this means that Caché cannot track it by its location.)
6. Having just added a new key to the key file, follow the instructions in the section "Protection from Accidental Loss of Access to Encrypted Data" to create and store a backup copy of the newly updated key file.
7. Refer to the section "Protection from Unauthorized Access to Encrypted Data" for details about measures to prevent currently or formerly privileged users from gaining unsanctioned access to encrypted data.

WARNING! InterSystems strongly recommends that you create and store a backup copy of the key file. Each time you create a database-encryption key, it is a unique key that cannot be re-created. Using the same administrator and password for a new key still results in the creation of a different and unique key. If an unactivated key is lost and cannot be recovered, the encrypted database that it protected will be unreadable and its data will be *permanently lost*.

10.1.3 Deleting a Key from a Key File

To delete a key from a key file, the procedure is:

1. From the Management Portal home page, go to the **Manage Encryption Key File** page (**System Administration** > **Encryption** > **Manage Encryption Key File**).
2. On the **Manage Encryption Key File** page, in the **Key File** field, enter the name of the key file from which you want to delete the key; click **OK**. This displays information about that key file; at the bottom of the shaded area, the **Encryption Keys Defined in Key File** table displays a list of the keys in the key file.
3. In the table of keys, click **Delete** in the row for a key to delete that key. Clicking **Delete** displays a confirmation page for the action. (If there is only one key in the file, there is no **Delete** button, as it is not permitted to delete this key.)

   If the key's **Delete** button is not available, this is because the key is the default encryption key or the journal encryption key for the file. To delete the key, first specify that another key is the default encryption key or the journal encryption key for the file by clicking **Set Default** or **Set Journal** for the other key.
4. Click **OK** on the confirmation dialog to delete the key from the file.

10.1.4 Adding an Administrator to a Key File

To add an administrator to an existing key file, the procedure is:

1. From the Management Portal home page, go to the **Manage Encryption Key File** page (**System Administration** > **Encryption** > **Manage Encryption Key File**).
2. In the **Key File** field, enter the path and filename of the key file to open and click **OK**; you can also use the **Browse** button to look for the key. Once the Portal opens the key file, it displays a table with the administrators listed in the file; administrator names appear in all capital letters, regardless of how they were defined.

3. In the table of administrators, click **Add** to add a new administrator. This displays a page with the following fields:
   - **Existing Administrator Name** — The name of an administrator already in the file.
   - **Existing Administrator Password** — The password associated with the already existing administrator in the file.
   - **New Administrator Name** — The name of the new administrator to be added to the file. Because the encryption functionality is independent of Caché security, the administrator name need not match any user names that are part of Caché security. This user name cannot include Unicode characters
   - **New Administrator Password** — The password for the new administrator. InterSystems suggests that this password follow the administrator password strength guidelines; also, this password cannot include Unicode characters. Because the encryption functionality is independent of Caché security, the password need not match the password that a user has for Caché security.
   - **Confirm New Administrator Password** — Confirmation of the password for the new administrator.

   Complete these fields and click **OK**. You have now added a new administrator to the key file.

Once you have added the new administrator to the key file, you may wish to copy the key file, making sure that each copy is in a secure location. Further, InterSystems strongly recommends that you create multiple administrators for each key, one of which has the name and password written down and stored in a secure location, such as in a fireproof safe. However, if copies of the key file are made and later on, as an administrative function, a new administrator is added, only the copy of the key file with the new administrator will be up to date.

Note: When you add a new administrator to a key file, that administrator's password is permanently associated with the entry for the administrator name created in the file. Once assigned, passwords cannot be changed. If you wish to assign a new password, delete the entry in the key file for that administrator name and then create a new entry with the same name and a new password.

10.1.5 Deleting an Administrator from a Key File

To delete an administrator from a key file, the procedure is:

1. From the Management Portal home page, go to the **Manage Encryption Key File** page (**System Administration** > **Encryption** > **Manage Encryption Key File**).
2. In the **Key File** field, enter the path and filename of the key and click **OK**. This displays a table with the administrators listed in the file (as well as a table of encryption keys in the file).
3. In the table of administrators, click **Delete** next to an administrator to remove that administrator for the key. Clicking **Delete** displays a confirmation page for the action. (If there is only one administrator in the file, there is no **Delete** button, as it is not permitted to delete this administrator.)
4. Click **OK** to delete the administrator from the file.

10.1.6 Activating a Database Encryption Key

Caché supports up to four simultaneously activated keys for database encryption. To activate a key for database encryption, the procedure is:

1. From the Management Portal home page, go to the **Database Encryption** page (**System Administration** > **Encryption** > **Database Encryption**). If there are already any activated keys, the page displays a table listing these.
2. On this page, click **Activate Key**, which displays the fields for activating a key.
3. Enter values for the following fields:

- **Key File** — The name of the file where the encryption key is stored. If you enter an absolute file name, Caché looks for the key file in the specified directory on the specified drive; if you enter a relative file name, Caché looks for the key file starting in the manager's directory for the Caché instance (which is below the Caché installation directory — that is, in <cache-install-dir>/mgr/). You can also use the **Browse** button to display a dialog for opening the key file.
- **Administrator Name** — The name of an administrator for this key, specified either when the key was created or edited.
- **Password** — The password specified for the named administrator.

4. Click the **Activate** button.

Caché then attempts to activate all the keys in the specified file. If there are not enough slots to activate all the keys in the file, then Caché opens as many keys as it can.

After key activation, the page displays the table of activated keys. For each key that Caché activates, the page adds the key to table of activated keys and displays the key's identifier. For each activated key, you can also perform various actions:

- **Set Default** — Click to specify that Caché uses this key when creating new encrypted databases. For more details, see the section "Specifying the Default Encryption Key or Journaling Encryption Key for an Instance."
- **Set Journal** — Click to specify that Caché uses this key to encrypt journal files. For more details, see the section "Specifying the Default Encryption Key or Journaling Encryption Key for an Instance."
- **Deactivate** — Click to deactivate this key. For more details, see the section "Deactivating a Database Encryption Key"

Note: The table of keys does not display any file or path information. This is because, once a key file is created, any sufficiently privileged operating system user can move it; hence, Caché may not have accurate information about the operating system location and can only rely on the accuracy of the GUID for the activated key in memory. To activate a second or subsequent key, note the identifier(s) for the currently activated key(s) first, so that you can identify the new one.

10.1.7 Deactivating a Database Encryption Key

To deactivate a database encryption key, the procedure is:

1. From the Management Portal home page, go to the **Database Encryption** page (**System Administration** > **Encryption** > **Database Encryption**). If a key is currently activated, its identifier appears in the table of keys.
2. You cannot deactivate a key if it is the default key either for new encrypted databases or for encrypting journal files. If you wish to deactivate a key that is Caché is using for either of these activities, then you must select a different key to be used for them. Do this by clicking **Set Default** or **Set Journal** for another key. Once the key is not in use for either of these activities, its **Deactivate** button will be available.
3. To deactivate the key, click **Deactivate** in its row.

Note: If it is not possible to deactivate the key for some other reason, the Portal displays an error message. Caché does not allow you deactivate a key under the following circumstances:

- The CACHETEMP and CACHE databases are encrypted.
- There is a currently-mounted encrypted database (other than CACHETEMP and CACHE) that is encrypted with this key.
- The key is currently in use to encrypt journal files. (Note that if you change the journal file encryption key, until you switch the journal file, Caché continues to use the old key for encryption.)

See below for information about how to address the underlying condition.

4. Click **OK** on the confirmation dialog to deactivate the key.

To deactivate the key, each underlying condition requires a different action:

- For any encrypted database except CACHETEMP and CACHE, dismount the database on the **Databases** page (**System Operation** > **Databases**). You can then deactivate the key.
- For CACHETEMP and CACHE, specify that these databases are not to be encrypted and then restart Caché. To do this, select **Configure Startup Settings** on the **Database Encryption** page; either you can choose not to activate a database encryption key at startup (in which case Caché turns off encryption for CACHETEMP and CACHE) or you can choose interactive or unattended database encryption key activation at startup (in which cases the choice whether or not to encrypt CACHETEMP and CACHE becomes available — choose "No").
- For encrypted journal files, ensure that no encrypted journal file is required for recovery. This is described in the section "Encrypted Journal Files and Configuring Startup without Key Activation."

10.1.8 Specifying the Default Database Encryption Key or Journal Encryption Key for an Instance

Each instance has a default database encryption key and a default journal encryption key. The instance sets the initial value for each of these when an administrator first activates a database encryption key; the key that is initially the default depends on the key(s) that are in the activated key file. These values are preserved across Caché shutdowns and restarts.

To specify a new key for either of these purposes, the procedure is:

1. From the Management Portal home page, go to the **Database Encryption** page (**System Administration** > **Encryption** > **Database Encryption**. This displays a table of currently activated encryption keys for the instance.
2. In the table of encryption keys:
   - To specify a new default encryption key, click **Set Default** for that key. The **Set Default** button for the current default key is unavailable.
   - To specify a new journal encryption key, click **Set Journal** for that key. The **Set Journal** button for the current journal encryption key is unavailable.
3. When prompted to confirm your action, click **OK.**

Caché then sets the selected key as the default or journal encryption key. If a key is either the default or journal encryption key, then it cannot be deleted (since it is required for operations on the Caché instance). Hence, specifying either of these for a key makes the key's **Delete** button unavailable.

10.1.9 Activating a Data Element Encryption Key

Caché supports up to four activated keys at one time for data element encryption. To activate a key for data element encryption, the procedure is:

1. From the Management Portal home page, go to the **Data Element Encryption** page (**System Administration** > **Encryption** > **Data Element Encryption**). If there are already any activated keys, the page displays a table listing these.
2. On the **Data Element Encryption** page, select **Activate Key**, which displays the fields for activating a key. If key activation is not available, this is because there are already four activated data element keys.
3. Enter values for the following fields:
   - **Key File** — The name of the file where the encryption key is stored. If you enter an absolute file name, Caché looks for the key file in the specified directory on the specified drive; if you enter a relative file name, Caché looks for the key file starting in the manager's directory for the Caché instance (which is below the Caché installation directory — that is, in <cache-install-dir>/mgr/).
   - **Administrator Name** — The name of an administrator for this key, specified either when the key was created or edited.
   - **Password** — The password specified for the named administrator.
4. Click the **Activate** button.

Caché then attempts to activate all the keys in the specified file. If there are not enough slots to activate all the keys in the file, then Caché opens as many keys as it can.

After key activation, the page displays the table of activated keys. For each key that Caché activates, the page adds the key to table of activated keys and displays the key's identifier.

Note: The table of keys does not display any file or path information. This is because, once the key file is activated, any sufficiently privileged operating system user can move the key; hence, Caché may not have accurate information about the operating system location and can only rely on the accuracy of the GUID for the activated key in memory. To activate a second or subsequent key, note the identifier(s) for the currently activated key(s) first, so that you can identify the new one.

10.1.10 Deactivating a Data Element Encryption Key

To deactivate a data element encryption key, the procedure is:

1. From the Management Portal home page, go to the **Data Element Encryption** page (**System Administration** > **Encryption** > **Data Element Encryption**) page. If there are any activated keys, the page displays a table listing them.
2. In the table of activated keys, for the key you wish to deactivate, click **Deactivate**. This displays a confirmation dialog.
3. In the confirmation dialog, click **OK**.

When the **Data Element Encryption** page appears again, the row in the table for the deactivated key should no longer be present.

10.1.11 Testing for a Valid Administrator Username-Password Pair

You can test interactively whether or not an administrator username-password pair is valid by attempting to activate a key in Caché. You can:

- Use the Management Portal. For more information, see either "Activating a Database Encryption Key" or "Activating a Data Element Encryption Key."
- Use the ^DATABASE utility. For more information, see "^DATABASE" section of the "Using the Character-based Security Management Routines" appendix.
- Run the database encryption conversion utility cvencrypt on an unmounted database (either encrypted with the database encryption key or unencrypted), up to the confirmation prompt:

  ```
  Continue? [Y/N]:
  ```

  At this prompt, answer No. For more information on cvencrypt, see the chapter "Using the cvencrypt Utility."

Do not perform this process programmatically, since it requires storing the password on disk, which is not recommended. (Unattended key activation at startup is a highly restricted special exception to storing a password on disk, as described in the section "Configuring Startup with Unattended Key Activation.")

10.1.12 Managing Keys and Key Files with Multiple-Instance Technologies

If you are using encrypted databases or journal files within a Caché cluster, the Caché instances on all nodes in the cluster must share a single database-encryption key.

Before enabling journal file encryption for any instance that belongs to a Caché mirror, see Activating Journal Encryption in a Mirror in the "Mirroring" chapter of the *Caché High Availability Guide* for important information. (There are no mirroring-related requirements in regard to database encryption.)

There are two ways to enable sharing of a single key:

- All of the instances share a single key file, which is located on one cluster node or mirror member.

  In this case, if you change the single copy of the key file, then these changes are visible to all nodes or members. However, if the host holding the key file becomes unavailable to the other nodes or members, any attempt to read the key from the key file fails; this can prevent Caché instances from restarting properly.
- Each cluster node or mirror member has its own copy of the key file.

  Here, if you change the key file, then you propagate copies of the key file (containing the same key) to all the other nodes or members. This increases the burden of administering the key file (which is typically small), but ensures that each instance of Caché always has a key available at startup.

Important: Whether there are single or multiple key files, the database-encryption key itself is the same for all instances.

10.1.12.1 Using a Single Key File

1. Create a database-encryption key on one node or member. For more information on this procedure, see the section "Creating a Key File."
2. Secure this key according to the instructions in the section "Protection from Accidental Loss of Access to Encrypted Data."

   CAUTION: Failure to take these precautions can result in a situation in which the encrypted databases or journal files are unreadable and *permanently lost*.
3. Configure each instance of Caché for unattended startup and provide Caché with the path to the key file. For more information on this procedure, see the section "Configuring Startup with Unattended Key Activation."

Since all the Caché instances use the same key, they are able to read data encrypted by each other. Any changes to the key file are visible to all instances.

Important: Refer to the section "Protection from Unauthorized Access to Encrypted Data" for details about measures to prevent currently or formerly privileged users from gaining unsanctioned access to encrypted data.

10.1.12.2 Using Multiple Key Files

If you choose to use multiple copies of the key file:

1. Create a database-encryption key on one node or member. For more information on this procedure, see the section "Creating a Key File."
2. Secure this key according to the instructions in the section "Protection from Accidental Loss of Access to Encrypted Data."

   CAUTION: Failure to take these precautions can result in a situation in which the encrypted databases or journal files are unreadable and *permanently lost*.

3. Make a copy of the key file for each of the remaining nodes or members.
4. On each node or member:
   a. Get a copy of the key file and put it in a secure and stable location on that machine.
   b. Configure each instance of Caché for unattended startup. For more information on this procedure, see the section "Configuring Startup with Unattended Key Activation."

Since each copy of the key file contains the same key, all the Caché instances are able to read data encrypted by each other. Since each Caché instance has a key file on its machine, the key file should always be available for a Caché restart. If there are any changes to the key file (such as adding or removing administrators), you must propagate new copies of the key file to each machine and reconfigure each instance of Caché for unattended startup using the new copy of the key file (even if that file is in the same location as the old file).

Important: Refer to the section "Protection from Unauthorized Access to Encrypted Data" for details about measures to prevent currently or formerly privileged users from gaining unsanctioned access to encrypted data.

10.2 Recommended Policies for Managing Keys and Key Files

The following sections address important topics for managing encrypted databases:

- Protection from Accidental Loss of Access to Encrypted Data
- Protection from Unauthorized Access to Encrypted Data

10.2.1 Protection from Accidental Loss of Access to Encrypted Data

Once you have created and activated a key, you can encrypt data. However, to ensure that such data is always available, it is strongly suggested that you take the following preventative actions:

- Create a second copy of a key file (a backup) and place it in a secure location, such as on some removable media that is stored in a fireproof safe.

- Create an additional administrator, the name and password of which are written down and stored in a secure location, such as in a fireproof safe at a site that is sufficiently far from where the key is in use.

CAUTION: Failure to take these precautions can result in a situation where the encrypted data will be *permanently inaccessible* — there will be no way to read it.

10.2.2 Protection from Unauthorized Access to Encrypted Data

To read or alter encrypted data, the appropriate encryption key for the Caché instance must be activated. Key activation requires three separate and separable elements: (1) the encrypted database or database containing the encrypted data elements, (2) the key itself, and (3) someone with a knowledge of a username and password for using the key. The design of the data encryption mechanism allows for the separation of these three factors. This separation allows an organization to keep the key in a secure location, separate from the encrypted data and under the control of authorized individuals who cannot use the key. This prevents those individuals with knowledge of how to use the key from doing so in an unauthorized manner — even if they have access to the data.

Put another way, Caché makes it possible to secure encryption keys so that those who have the key cannot use it and those who can use the key do not have it. By this arrangement, the key can only be activated when those who have it make it available to those who can use it.

CAUTION: If administrators have free access to a key file, then it is possible for them to make unauthorized copies of the key file. Such copies might be used by formerly authorized members of an organization, could be lost, and so on.

The degree of secure storage required is a function of the sensitivity of the data. Under the strictest circumstances, activation might occur as follows:

1. A system administrator (who does not have the key but does have the information to use the key) needs to re-start Caché. After or as part of a Caché re-start, the database encryption key needs to be activated, so that encrypted databases or databases containing encrypted data can be mounted.
2. The system administrator contacts the key holder (who has the key but lacks the information to use it). The key holder may be part of a site's staff that provides physical security; the key itself may even be stored in a safe or vault.
3. The key holder retrieves the key, which may be on a CD, USB drive, or other portable storage device. The key holder then brings the key to where it is used for key activation.
4. The system administrator then performs key activation — under the oversight of the key holder, who ensures that the system administrator performs no other actions, such as copying the key file.
5. Once the key has been activated, the key holder returns the device holding the key to its physically secure location until it is needed again.

Implementing this arrangement is for the purpose of preventing the system administrator from obtaining the key. Again, this is because possessing the key would enable the administrator to gain potentially unauthorized access to the encrypted data.

10.3 Using Encrypted Databases

To protect entire databases that contain sensitive information, Caché supports block-level database encryption (or, for short, database encryption). Database encryption is technology that allows you to create and manage databases that, as entire entities, are encrypted; it employs the Caché key management tools to support its activities.

When you create a database, you can choose to have it be encrypted; this option is available if there is a currently activated key. Once you have created an encrypted database, you can use it in the same way as you would use an unencrypted database. The encryption technology is transparent and has a small and deterministic performance effect.

The database encryption functionality also supports the ability to encrypt the audit log and journal files; for any instance that serves as a destination shadow server, enabling journal file encryption also results in the encryption of the journal files that the instance receives from its source production server. Both these features require access to the database encryption key at startup time, as described in the section "Configuring Caché Encryption Startup Settings."

Management tasks for encrypted databases include:

- Creating an Encrypted Database
- Establishing Access to an Encrypted Database
- Closing an Encrypted Database
- Moving an Encrypted Database between Instances
- Configuring Caché Database Encryption Startup Settings
- About Encrypting the Databases that Ship with Caché

10.3.1 Creating an Encrypted Database

When creating a new database, you can specify that it is encrypted. However, before you can create an encrypted database, Caché must have an activated database-encryption key. Hence, the procedure is:

1. Activate a database-encryption key.
2. From the Management Portal home page, go to the **Local Databases** page (**System Administration** > **Configuration** > **System Configuration** > **Local Databases**).
3. On the **Local Databases** page, select **Create New Database**. This displays the **Create Database** wizard.
4. On the second page of the wizard, in the **Encrypt Database?** box, select **Yes**. This causes Caché to create an encrypted database. On all the other pages of the wizard, choose database characteristics as you would when creating any database. (For more information on creating databases, see the "Create Local Databases" section of the "Configuring Caché" chapter of the *Caché System Administration Guide*.)

Note: Caché also provides the cvencrypt utility to encrypt unencrypted databases or decrypt encrypted databases, if this is necessary.

10.3.2 Establishing Access to an Encrypted Database

To perform various operations, such as adding a database to a mirror, the database must be mounted. However, for an encrypted database to be mounted, its key must be activated. Hence, access to the database requires activating the key and mounting the database, and the procedure for this is:

1. Activate the key.
2. From the Management Portal home page, go to the **Databases** page (**System Operation** > **Databases**).
3. On this page, for the database that you wish to mount, select the **Mount** button in the far right column of its row in the table of databases. After selecting **OK** on the confirmation screen, the database is mounted. If the key is not activated, Caché cannot mount the database and displays an error message.

You can now access the data within the database.

10.3.3 Closing the Connection to an Encrypted Database

To close the connection to an encrypted database, the procedure is:

1. From the Management Portal home page, go to the **Databases** page (**System Operation** > **Databases**).
2. On this page, select the **Dismount** button on the right in the table of databases. After selecting **OK** on the confirmation screen, the database is dismounted.
3. Deactivate the key.

Because the activated key is used for each read and write to the database, you cannot deactivate the key without first dismounting the database. If you attempt to deactivate the key prior to dismounting the database, Caché displays an error message.

10.3.4 Moving an Encrypted Database Between Instances

If your organization has multiple Caché instances, you may need to use an encrypted database on one instance that was created on another instance using a different key. To move the data from one instance to another, back up and then re-key the database using the **cvencrypt** utility. For more information on this process, see the section "Converting an Encrypted Database to Use a New Key" in the "Using the cvencrypt Utility" appendix of this book.

10.3.5 Configuring Caché Database Encryption Startup Settings

By default, an instance of Caché does not have a database encryption key available at startup time. This means that, by default, there are several features that are not available (see below). Caché can either gather the encryption key file information interactively or without human intervention (called "unattended startup"). This section describes how to set up three database-encryption startup options:

- Configuring Startup without Key Activation
- Configuring Startup with Interactive Key Activation
- Startup with Unattended Key Activation

The features that depend on having a key available at startup time are:

- Encrypting the Caché audit log.
- Encrypting the CACHETEMP and CACHE databases. (Either both are encrypted or neither.)
- Encrypting Caché journal files and, for a destination shadow server, encrypting the journal files received from the source production server. (Either both are encrypted or neither.)
- Having an encrypted database mounted at startup time.

10.3.5.1 Configuring Startup without Key Activation

This is the default behavior for an instance of Caché prior to having any keys activated. If you have set up key activation at startup and choose to turn it off, the procedure is:

1. From the Management Portal home page, go to the **Database Encryption** page (**System Administration** > **Encryption** > **Database Encryption**).
2. Select **Configure Startup Settings**. This displays the area with options for configuring Caché startup and other options for encrypted databases.
3. In this area, from the **Startup Options** list, select **None**.

4. Click **Save**. Caché may prevent you from performing this action if:
   - Any encrypted databases are required at startup. See "Required Encrypted Databases and Configuring Startup without Key Activation" for more details.
   - There are any encrypted journal files with open transactions. See "Encrypted Journal Files and Configuring Startup without Key Activation" for more details.
   - The audit log is encrypted. (The error message for this refers to an encrypted database because Caché stores the audit log in a database called CACHEAUDIT.) See "The Caché Audit Log and Configuring Startup without Key Activation" for more details.

   Address the issue that is preventing the change and then perform this procedure again. Once any issues are corrected, you will be able to successfully change to having startup without key activation.

Required Encrypted Databases and Configuring Startup without Key Activation

If the instance has encrypted databases that are required at startup and you attempt to configure startup not to involve key activation, the Management Portal displays an error message stating this and indicating that the key activation option cannot be changed. (If the error message refers to the CACHEAUDIT database, this is because the audit log is encrypted.)

To configure Caché to start without activating an encryption key, any encrypted databases can only be mounted after startup. To configure a database to be mounted after startup, the procedure is:

1. Confirm that the database is mounted or mount it. To do this:
   a. From the Management Portal home page, go to the **Databases** page (**System Operation** > **Databases**).
   b. Find the database's row in the table of databases. If it is mounted, there is a **Dismount** choice in its row; if it is not mounted, there is no **Dismount** choice and there *is* a **Mount** choice.
   c. If it is not mounted, select **Mount**
   d. On the confirmation screen, select **OK**. (The database needs to be writeable, so do not select the **Read Only** check box.)

2. Edit the database's properties so that it is not mounted at startup. To do this:
   a. Go to the **Local Databases** page (**System Administration** > **Configuration** > **System Configuration** > **Local Databases**).
   b. Find the database's row in the table of databases.
   c. Select the database by clicking on its name. This displays the page for editing the database.
   d. On this **Edit** page, clear the **Mount Required at Startup** check box.
   e. Click **Save**.

The database will no longer be mounted at startup. This means that it will no longer require key activation at startup (though it may be required for other reasons.)

Encrypted Journal Files and Configuring Startup without Key Activation

If the instance uses journaling and you attempt to configure startup not to involve key activation, you may be unable to turn off key activation at startup. These conditions are:

- The instance is configured to encrypt its journal files (either as a source production server or as a destination shadow server).
- There are open transactions in the journal file (which is fairly likely on a busy system).

If this occurs, you need to suspend the use of encrypted journal files before you change the startup key activation settings. To do this, the procedure is:

1. On the **Database Encryption** page (**System Administration** > **Encryption** > **Database Encryption**), change the **Encrypt Journal Files** setting to "No." Leave the **Key Activation at Startup** setting as it is.
2. Switch journal files. To do this, click **Switch Journal** on the **Journals** page (**System Operation** > **Journals**).

Once all open transactions within the encrypted journal files have either been committed or rolled back, you can then change the Caché startup configuration.

CAUTION: Even after there are no open transactions, you may need the encrypted journal files to restore a database. For this reason, it is very important that you maintain copies of the key file containing the key used to encrypt these files.

For more information on journal files generally, see the "Journaling" chapter of the *Caché Data Integrity Guide*.

The Caché Audit Log and Configuring Startup without Key Activation

If the instance has an encrypted audit log and you attempt to configure startup not to involve key activation, Caché displays an error message that an encrypted database is required at startup, such as:

```
ERROR #1217: Can not disable database encryption key activation at startup.
Encrypted databases are required at startup:
C:\InterSystems\Cache\Mgr\CacheAudit\
```

The error message refers to encrypted databases because the audit log is stored in the Caché database CACHEAUDIT.

The audit log cannot be encrypted if Caché starts without activating an encryption key. To configure startup not to involve key activation, you must change the Caché setting to specify that the instance uses an unencrypted audit log. The procedure is:

1. Back up the instance's audit data.
2. Go to the **Database Encryption** page (**System Administration** > **Encryption** > **Database Encryption**).
3. Select **Configure Startup Settings**, which displays the area with options for configuring Caché startup and other options for encrypted databases.
4. Under **Optionally Encrypted Data**, in the **Encrypt Audit Log** list, click "No".

Changing this setting causes Caché to erase any existing audit data, to start using unencrypted auditing immediately, and to write an AuditChange event to the audit log.

CAUTION: If you have not backed up audit data, changing the encryption setting for the audit log results in the loss of that existing audit data.

10.3.5.2 Configuring Startup with Interactive Key Activation

This is the default behavior for an instance of Caché if a key has been activated. With interactive key activation, the Caché instance prompts for the location of a key and its associated information during its startup.

Important: On Windows, interactive key activation is incompatible with configuring Caché as a service that starts automatically as part of system startup.

To configure Caché for interactive key activation:

1. From the Management Portal home page, go to the **Database Encryption** page (**System Administration** > **Encryption** > **Database Encryption**).
2. Select **Configure Startup Settings**. This displays the **Startup Options** area, which includes the **Key Activation at Startup** list.

3. In the **Key Activation at Startup** list, select **Interactive**. If the previous value for the field was "None", then this displays the page's **Optionally Encrypted Data** area.
4. The fields in this area are:
   - **Encrypt CACHETEMP and CACHE Databases** — Allows you to specify whether or not the CACHETEMP and CACHE databases are encrypted. To encrypt them, select "Yes"; to have them be unencrypted, select "No."
   - **Encrypt Journal Files** — Allows you to specify whether or not the instance encrypts its own journal files and, if it is a destination shadow server, if it encrypts the journal files it receives from its source production server. To encrypt journal files, select "Yes"; to have them be unencrypted, select "No." This choice depends on startup options because Caché startup creates a new journal file; if you choose encryption, startup requires a key.

     Note: This change takes effect the next time that Caché switches journal files. To begin journal file encryption without a restart, switch journal files after completing this page.

   - **Encrypt Audit Log** — Allows you to specify whether or not Caché encrypts the audit log. To encrypt the audit log, select "Yes"; to have it be unencrypted, select "No." This choice depends on startup options because the Caché startup procedure records various events in the audit log; if you choose encryption, startup requires a key.

     CAUTION: This change takes effect immediately and deletes any existing audit data. Back up the audit database prior to changing this setting; otherwise, audit data will be lost.

5. Click **Save** to save the selected settings.

Important: If Caché is configured to

- Encrypt CACHETEMP and CACHE, journal files, or the audit log
- Require an encrypted database at startup

then failure to activate the required encryption key causes a Caché startup failure. If this occurs, use Caché emergency startup mode to configure Caché not to require any encrypted facilities at startup.

10.3.5.3 Configuring Startup with Unattended Key Activation

Unattended startup requires that all the elements required to decrypt an encrypted database be available when Caché starts running. This includes the Caché instance itself, the encrypted database, the database-encryption key file, and the username and password used for unattended database encryption key activation. By making all these items available, the security of the data in Caché becomes entirely dependent on the physical security of the machine(s) holding these elements. If your site cannot ensure this physical security, your data will then be subject to the same level of risk as if it were not encrypted; to avoid this situation, either use interactive startup (which prevents the simultaneous exposure of these elements) or ensure the physical security of the relevant machine(s).

CAUTION: InterSystems recommends that you do *not* use unattended key activation.

If you want Caché to activate a key at startup without requiring any human intervention, then:

1. You need to have a key currently activated. To activate a key, see the section "Activating a Key."
2. From the Management Portal home page, go to the **Database Encryption** page (**System Administration** > **Encryption** > **Database Encryption**).
3. Select **Configure Startup Settings**. This displays the **Startup Options** list.
4. In **Startup Options**, select **Unattended (NOT RECOMMENDED)**. This changes the fields that the page displays.
5. The **Startup Options** area expands to display three fields. Complete these:

- **Key File** — The path of the database-encryption key file. This can be an absolute or relative path; if you specify a relative path, it is relative to the Caché installation directory. Click **Browse** to search for the database-encryption key file on the file system.
- **Administrator Name** — An administrator for this key file.
- **Password** — The administrator's password.

6. Complete the fields in the **Optionally Encrypted Data** area:
   - **Encrypt CACHETEMP and CACHE Databases** — Allows you to specify whether or not the CACHETEMP and CACHE databases are encrypted. To encrypt them, select "Yes"; to have them be unencrypted, select "No."
   - **Encrypt Journal Files** — Allows you to specify whether or not the instance encrypts its own journal files and, if it is a destination shadow server, if it encrypts the journal files it receives from its source production server. To encrypt journal files, select "Yes"; to have them be unencrypted, select "No." This choice depends on startup options because Caché startup creates a new journal file; if you choose encryption, startup requires a key.

     Note: This change takes effect the next time that Caché switches journal files. By default, this occurs the next time that Caché restarts. To begin journal file encryption without a restart, switch journal files after completing this page.

   - **Encrypt Audit Log** — Allows you to specify whether or not Caché encrypts the audit log. To encrypt the audit log, select "Yes"; to have it be unencrypted, select "No." This choice depends on startup options because the Caché startup procedure records various events in the audit log; if you choose encryption, startup requires a key.

     CAUTION: This change takes effect immediately and deletes any existing audit data. Back up the audit database prior to changing this setting; otherwise, audit data will be lost.

7. Click **Save** to save the selected settings.

When you configure Caché for unattended startup, it adds another administrator to the database-encryption key file; that administrator has a system-generated name and password. Ideally, the key file should be on a medium that can be physically locked in place, such as a lockable CD-ROM or DVD drive in a rack; the data center facility where it is stored should be locked and monitored. Do *not* store the database-encryption key on the same drive as any databases that it is used to encrypt.

Important: If Caché is configured to

- Encrypt CACHETEMP and CACHE, journal files, or the audit log
- Require an encrypted database at startup

then failure to activate the encryption key causes a Caché startup failure. If this occurs, use Caché emergency startup mode to configure Caché not to require any encrypted facilities at startup.

10.3.6 About Encrypting the Databases that Ship with Caché

Each instance of Caché ships with a number of databases. The ability to encrypt and the value of encryption depends on the database:

- CACHE: Can be encrypted in conjunction with the CACHETEMP database. Encrypting CACHE requires that a key be available at startup, since the database is required at startup time.
- CACHEAUDIT: Can be encrypted. Encrypting CACHEAUDIT requires that a key be available at startup, since the database is required at startup time.
- CACHELIB: Must not be encrypted. (Note that all content in CACHELIB is publicly available.)

- CACHESYS: Must not be encrypted. If an instance has an encrypted form of this database, Caché cannot start.
- CACHETEMP: Can be encrypted in conjunction with the CACHE database. Encrypting CACHETEMP requires that a key be available at startup, since the database is required at startup time.
- DOCBOOK: Can be encrypted. (Note that all content in DOCBOOK is publicly available.)
- SAMPLES: Can be encrypted. (Note that all content in SAMPLES is publicly available.)
- USER: Can be encrypted.

10.4 Using Data Element Encryption

Data element encryption provides a means of encrypting application data at a finer level of granularity than the database as a whole; it is for sensitive data elements whose exposure must be prevented. For example, customer records can exclusively encrypt the credit card field; patient records can exclusively encrypt fields that display test results (such as for HIV testing); or a record that includes a social security number can exclusively encrypt that field.

Data element encryption is available programmatically (via an API), rather than through the Management Portal. Because it is accessible through an API, you can use it in your application code. You have the option of using data element encryption with database encryption (though there is no requirement to use both).

For an application to use data element encryption, the required keys must be available when the application is running. To make a key available, activate it; for details, either see the following section "Programmatically Managing Keys" or, if using the Portal, see "Activating a Key for Data Element Encryption". When a key is activated, Caché displays its unique identifier in the table of activated keys; the application then uses this identifier to refer to the key so that it can be loaded from memory for encryption operations. Since there can be up to four keys simultaneously activated, data element encryption provides the infrastructure for tasks that require multiple keys.

When encrypting data for data element encryption, Caché stores the encryption key's unique identifier with the resulting ciphertext. The unique identifier enables the system to identify the key at decryption time using only the ciphertext itself.

This section describes:

- Programmatically Managing Keys
- Data Element Encryption Calls
- Support for Re-Keying Data in Real Time

10.4.1 Programmatically Managing Keys

Since data element encryption is available through an API, there are also a set of calls for managing keys:

- **$SYSTEM.Encryption.CreateEncryptionKey**
- **$SYSTEM.Encryption.ActivateEncryptionKey**
- **$SYSTEM.Encryption.DeactivateEncryptionKey**
- **$SYSTEM.Encryption.ListEncryptionKeys**

These are all methods of the %SYSTEM.Encryption class.

10.4.2 Data Element Encryption Calls

The system methods available for data element encryption are all methods of the %SYSTEM.Encryption class and are:

- $SYSTEM.Encryption.AESCBCManagedKeyEncrypt
- $SYSTEM.Encryption.AESCBCManagedKeyDecrypt
- $SYSTEM.Encryption.AESCBCManagedKeyEncryptStream
- $SYSTEM.Encryption.AESCBCManagedKeyDecryptStream

These method names all begin with "AESCBCManagedKey" because the methods use AES (the Advanced Encryption Standard) in cipher block chaining (CBC) mode and are part of the suite of tools for managed key encryption.

Important: The AESCBC methods that do not include "ManagedKey" in their names are older methods and cannot be used for these purposes.

10.4.2.1 $SYSTEM.Encryption.AESCBCManagedKeyEncrypt

The signature of this method as it is usually called is:

```
$SYSTEM.Encryption.AESCBCManagedKeyEncrypt
        (
        plaintext As %String,
        keyID As %String,
        )
    As %String
```

where:

- *plaintext* — The unencrypted text to be encrypted.
- *keyID* — The GUID of the data-encryption key to be used to encrypt plaintext.
- The method returns the encrypted ciphertext.

If the method fails, it throws either the <FUNCTION> or <ILLEGAL VALUE> error. Place calls to this method in a **Try-Catch** loop; for more information on Try-Catch, see the section "The TRY-CATCH Mechanism" in the "Error Processing" chapter of *Using Caché ObjectScript*.

For more details, see the **$SYSTEM.Encryption.AESCBCManagedKeyEncrypt** class reference content.

10.4.2.2 $SYSTEM.Encryption.AESCBCManagedKeyDecrypt

The signature of this method as it is usually called is:

```
$SYSTEM.Encryption.AESCBCManagedKeyDecrypt
        (
        ciphertext As %String
        )
    As %String
```

where:

- *ciphertext* — The encrypted text to be decrypted.
- The method returns the decrypted plaintext.

If the method fails, it throws either the <FUNCTION> or <ILLEGAL VALUE> error. Place calls to this method in a **Try-Catch** loop; for more information on Try-Catch, see the section "The TRY-CATCH Mechanism" in the "Error Processing" chapter of *Using Caché ObjectScript*.

You do not need to include the key ID with this call, as the key ID is associated with the ciphertext to be decrypted.

For more details, see the **$SYSTEM.Encryption.AESCBCManagedKeyDecrypt** class reference content.

10.4.2.3 $SYSTEM.Encryption.AESCBCManagedKeyEncryptStream

The signature of this method as it is usually called is:

```
$SYSTEM.Encryption.AESCBCManagedKeyEncryptStream
        (
        plaintext As %Stream.Object,
        ciphertext As %Stream.Object,
        keyID As %String,
        )
    As %Status
```

where:

- *plaintext* — The unencrypted stream to be encrypted.
- *ciphertext* — The variable to receive the encrypted stream.
- *keyID* — The GUID of the data-encryption key to be used to encrypt *plaintext*.
- The method returns a %Status code.

For more details, see the **$SYSTEM.Encryption.AESCBCManagedKeyEncryptStream** class reference content.

10.4.2.4 $SYSTEM.Encryption.AESCBCManagedKeyDecryptStream

The signature of this method as it is usually called is:

```
$SYSTEM.Encryption.AESCBCManagedKeyDecryptStream
        (
        ciphertext As %Stream.Object,
        plaintext As %Stream.Object
        )
    As %Status
```

where:

- *ciphertext* — The encrypted stream to be decrypted.
- *plaintext* — The variable to receive the unencrypted stream.
- The method returns a %Status code.

You do not need to include the key ID with this call, as the key ID is associated with the ciphertext to be decrypted.

For more details, see the **$SYSTEM.Encryption.AESCBCManagedKeyDecryptStream** class reference content.

10.4.3 Support for Re-Keying Data in Real Time

Data element encryption allows Caché applications to support *re-keying data*, which is re-encrypting an encrypted data element with a new key.

Because an encrypted data element has its encrypting key identifier stored with it, this simplifies the process of re-keying data. Given merely the handle to ciphertext and an activated key, an application can perform re-keying. For example, data element encryption supports the ability to re-key sensitive data without any downtime; this is particularly useful for users required to perform this action for legal reasons, such as those fulfilling PCI DSS (Payment Card Industry Data Security Standard) requirements.

If you need to re-key data, create a new key and specify to your application that this is the new encryption key. You can then perform an action such as running a background application to decrypt the elements and encrypt them with the new

key. This uses the specified key for encryption and always uses the correct key for decryption, since it is stored with the encrypted data.

10.5 Emergency Situations

This section describes what to do under certain circumstances when you are in danger of losing data. These include:

- If the File Containing an Activated Key is Damaged or Missing
  - If There Is a Backup Copy of the Key File with a Known Administrator Username and Password
  - If There Is No Backup Copy of the Key File or the Key has No Known Administrator Username and Password

    CAUTION: This is a *dire* situation. Act *immediately*.

- If the Database-Encryption Key File Is Required at Startup and Is Not Present
  - If You Can Make the Key File Available
  - If a Backup Key File Is Available
  - If No Key File Is Available

10.5.1 If the File Containing an Activated Key is Damaged or Missing

In this situation, the following circumstances have occurred:

- A database-encryption key has been activated for the Caché instance.
- Caché is using encrypted data.
- The key file containing the database-encryption key becomes damaged.

10.5.1.1 If There Is a Backup Copy of the Key File with a Known Administrator Username and Password

CAUTION: This procedure is for an emergency situation, where encrypted data in Caché databases is in danger of being lost.

If the file containing an activated key becomes inaccessible or damaged, *immediately* perform the following procedure:

1. Get the backup copy of the key file. This is the copy that you stored as described in the section "Protection from Accidental Loss of Access to Encrypted Data."
2. Make a new backup copy of the key file and store it in a safe place. Follow all the precautions specified in the section "Special Cautions to Preserve the Data in Encrypted Databases."
3. Set up Caché to use the new copy of the key:
   - If you are using interactive startup, incorporate the new copy of the key into your startup procedures.
   - If you are using unattended startup, then re-configure your startup options using the new copy of the key file — even if you are setting it up for the same options as before.

10.5.1.2 If There Is No Backup Copy of the Key File or the Key has No Known Administrator Username and Password

Important: THIS PROCEDURE IS FOR AN EMERGENCY SITUATION, WHERE ENCRYPTED DATA IN CACHÉ DATABASES IS IN DANGER OF BEING LOST.

If the file containing the activated key becomes inaccessible or damaged while Caché is running, *immediately* perform the following procedure for each database encrypted with that key:

1. *Do not shut down Caché.*

   Do not deactivate the currently active key.

   WARNING! Shutting down Caché or deactivating the active key will cause the permanent loss of your data.

2. Contact the InterSystems Worldwide Response Center. Engineers there can help guide you through the following procedure and answer any questions that may arise.
3. Prevent all users from making any changes to the database with encrypted content while copying its data to an unencrypted database. To do this, the procedure is:
   a. From the Management Portal home page, go to the **Databases** page (**System Operation** > **Databases**).
   b. On the **Databases** page, if the encrypted database is mounted, select the **Dismount** option in the next-to-last column in that database's row. Then select **OK** in the confirmation dialog.
   c. When the **Databases** page appears again, select the **Mount** option in the last column in the database's row.
   d. On the **Mount database** confirmation screen, check the **Read Only** box and select **OK**.

   It is critical that no one makes any changes to the database during this procedure. Mounting the database read-only prevents any user from changing any data.
4. Copy all data in unencrypted form to another database. The procedure for copying the data is:
   a. In the Terminal, go to the %SYS namespace:

   ```
   REGULARNAMESPACE> zn "%SYS"
   ```

   b. From that namespace, run the **^GBLOCKCOPY** command:

   ```
   %SYS>d ^GBLOCKCOPY

   This routine will do a fast global copy from a database to another database or
   to a namespace. If a namespace is the destination, the global will follow any
   mappings set up for the namespace.

   1) Interactive copy
   2) Batch copy
   3) Exit

   Option?1
   ```

   c. At the **^GBLOCKCOPY** prompt, specify 1, for an interactive copy:

   ```
   Option? 1

   1) Copy from Database to Database
   2) Copy from Database to Namespace
   3) Exit

   Option?
   ```

   d. When **^GBLOCKCOPY** prompts for a copy type, select 1, for copying from database to database

   ```
   Option? 1
   Source Directory for Copy (? for List)?
   ```

Here, either specify the name of the encrypted database or enter ? to see a numbered list of databases, which includes the encrypted database. If you enter ?, **^GBLOCKCOPY** displays a list such as this one:

```
Source Directory for Copy (? for List)? ?

1) C:\InterSystems\MyCache\mgr\
2) C:\InterSystems\MyCache\mgr\cache\
3) C:\InterSystems\MyCache\mgr\cacheaudit\
4) C:\InterSystems\MyCache\mgr\cachelib\
5) C:\InterSystems\MyCache\mgr\cachetemp\
6) C:\InterSystems\MyCache\mgr\docbook\
7) C:\InterSystems\MyCache\mgr\encrypted1\
8) C:\InterSystems\MyCache\mgr\encrypted2\
9) C:\InterSystems\MyCache\mgr\samples\
10) C:\InterSystems\MyCache\mgr\unencrypted\
11) C:\InterSystems\MyCache\mgr\user\

Source Directory for Copy (? for List)?
```

Enter the number of the encrypted database, such as 7 here.

e. When **^GBLOCKCOPY** prompts for a destination directory for copying the data, enter the name of an unencrypted database or ? for a list similar to the one for the source directory.

f. When **^GBLOCKCOPY** asks if you wish to copy all globals, enter `Yes` (can be `Yes`, `Y`, `y`, and so on):

```
All Globals? No => y
```

g. If there is an empty global in the database, **^GBLOCKCOPY** will now ask if you wish to copy it. This will appear something like the following:

```
All Globals? No => y

^oddBIND     contains no data
Include it anyway? No =>
```

Enter `No` (can be `No`, `N`, `n`, and so on), which is the default.

h. **^GBLOCKCOPY** then asks if you wish to skip all the other empty globals. Enter `Yes` (can be `Yes`, `Y`, `y`, and so on), which is the default:

```
Exclude any other similar globals without asking again? Yes =>
```

There then appears a list of all the empty globals that are not being copied:

```
Exclude any other similar globals without asking again? Yes => Yes
^oddCOM      contains no data --  not included
^oddDEP      contains no data --  not included
^oddEXT      contains no data --  not included
^oddEXTR     contains no data --  not included
^oddMAP      contains no data --  not included
^oddPKG      contains no data --  not included
^oddPROC     contains no data --  not included
^oddPROJECT  contains no data --  not included
^oddSQL      contains no data --  not included
^oddStudioDocument contains no data --  not included
^oddStudioMenu contains no data --  not included
^oddTSQL     contains no data --  not included
^oddXML      contains no data --  not included
^rBACKUP     contains no data --  not included
^rINC        contains no data --  not included
^rINCSAVE    contains no data --  not included
^rINDEXEXT   contains no data --  not included
^rINDEXSQL   contains no data --  not included
^rMACSAVE    contains no data --  not included
9 items selected from
29 available globals
```

i. **^GBLOCKCOPY** then asks if you wish to disable journaling for this operation:

```
Turn journaling off for this copy? Yes =>
```

Enter `Yes` (can be `Yes`, `Y`, `y`, and so on), which is the default.

j. **^GBLOCKCOPY** then asks if to confirm that you wish to copy the data:

```
Confirm copy? Yes =>
```

Enter `Yes` (can be `Yes`, `Y`, `y`, and so on), which is the default. Depending on the size of the database and the speed of the processor, you may see the status of the copy as it progresses. When it completes, **^GBLOCKCOPY** displays a message such as:

```
Copy of data has completed
```

k. **^GBLOCKCOPY** then asks if you wish to save statistics associated with the copy. Enter `No` (can be `No`, `N`, `n`, and so on), which is the default:

```
Do you want to save statistics for later review? No =>
```

Control then returns to the main prompt.

5. Test that the copied data is valid. You can do this by examining the classes, tables, or globals in the Management Portal's System Explorer for the database into which **^GBLOCKCOPY** has copied the data.
6. If the data is valid, perform steps 3 and 4 of this procedure for each database encrypted with the inaccessible or damaged key.
7. Once you have made copies of every encrypted database into an unencrypted database, make a second copy of each database, preferably to a different machine than that which holds the first copy of each.
8. Now — *and only now* — you can dismount all encrypted databases and deactivate the active key (that is, the key for which the key file is missing or damaged). Caché requires that you dismount all encrypted databases prior to deactivating their key.

You now have your data in one or more unencrypted databases and there is no activated key.

To re-encrypt the formerly encrypted databases, the procedure is:

1. Create a new database-encryption key according to the procedure described in the section "Creating a Key."
2. Create a new backup copy of the key file as described in "Protection from Accidental Loss of Access to Encrypted Data."

   CAUTION: Make sure you take the precautions described in "Protection from Accidental Loss of Access to Encrypted Data"; failure to follow these procedures can result in the permanent loss of data.

3. Create one or more new encrypted databases, using the new key.
4. Import the data exported in the previous procedure into the new encrypted database(s).

Your data is now stored in encrypted databases for which you have a valid key and a backup copy of the key file containing that key.

10.5.2 If the Database-Encryption Key File Is Required at Startup and Is Not Present

Under certain conditions related to the required use of a database-encryption key file at startup, the system starts in single-user mode. These conditions are:

- Caché is configured for either interactive or unattended startup.
- Startup specifies that journal files and/or the CACHETEMP and CACHE databases are encrypted, or an encrypted database is specified as required at startup.

- The database-encryption key file is not present.

10.5.2.1 If You Can Make the Key File Available

This situation may have been caused simply by the appropriate key file not being present at Caché startup time — such as if the media holding it is not currently available.

To correct the condition, after Caché starts running in single-user mode, then the procedure is:

1. Shut down Caché. For example, if the instance of Caché is called "MyCache", the command to do this would be:

   ```
   ccontrol force MyCache
   ```

2. If you know the location where Caché is expecting to find the database-encryption key file, then place the key file there. (Otherwise, you need to run **^STURECOV** as specified in the next section.)
3. Start Caché again.

Caché should start in its typical mode (multi-user mode) and operate as expected.

10.5.2.2 If a Backup Key File Is Available

If the appropriate key file is not present at Caché startup time and is not available, you may have a backup key file available. If so, then to correct the condition, after Caché starts running in single-user mode, then the procedure is:

1. Contact InterSystems Worldwide Response Center. Engineers there can help guide you through the following procedure and answer any questions that may arise.
2. Start a Terminal session according to the instructions in the most recent entry in the cconsole.log file. Typically, this specifies starting a Terminal session with the `-B` flag.

   For example, at a Windows command line, for an instance of Caché called "MyCache" that is installed in the default location, the command would be:

   ```
   C:\MyCache\bin\cache -sC:\MyCache\mgr -B
   ```

   This connects you with Caché in the operating system terminal window; the prompt in that window changes from the operating system prompt to the Caché %SYS prompt.
3. If you have or can obtain a copy of the database-encryption key file (such as a backup), then place a copy of the key file in a location accessible to Caché.
4. Run the **^STURECOV** (startup recovery) routine at the Terminal prompt. In that routine, activate the encryption key using an administrator username and password in that file. (You do not need to exit **^STURECOV** when you have completed this process.)
5. When you are satisfied that Caché is ready for use, use **^STURECOV** to complete the startup procedure. Caché then starts in multi-user mode.

Caché should now operate as expected.

10.5.2.3 If No Key File Is Available

If you do not have any copy of the database-encryption key file, then the procedure is:

1. Contact InterSystems Worldwide Response Center (WRC). Engineers there can help guide you through the following procedure and answer any questions that may arise.
2. Start a Terminal session with the `-B` flag. For example, at a Windows command line, for an instance of Caché called "MyCache" that is installed in the default location, the command would be:

```
C:\MyCache\bin\cache -sC:\MyCache\mgr -B
```

This connects you with Caché in the operating system terminal window; the prompt in that window changes from the operating system prompt to the Caché %SYS prompt.

3. If any encrypted databases require mounting at startup, disable this feature for them:
   a. From the Management Portal Home page, go to the **Local Databases** page (**System Administration** > **Configuration** > **System Configuration** > **Local Databases**).
   b. Click the name of the database in the table of databases. This displays the **Edit:** page for the database.
   c. On the **Edit:** page, clear the **Mount Required at Startup** check box.
   d. Click **Save**.
4. Run the **^STURECOV** routine at the Terminal prompt. In that routine, configure Caché database startup options not to require a database encryption key. This means that the CACHETEMP and CACHE databases as well as journal files should now operate as expected; it also means that any encrypted databases cannot be mounted.
5. When you are satisfied that Caché is ready for use, use **^STURECOV** to complete the startup procedure. Caché then starts in multi-user mode.

As you perform this procedure, you may need to perform other actions, according to the instructions of the representative from the WRC. Follow these instructions.

CAUTION: If you have not performed the actions described in the section "Protection from Accidental Loss of Access to Encrypted Data," then your data may no longer be available in any form. This is a very serious problem, but if you do not have a key, there is no way to retrieve the lost data.

10.6 Other Information

This section addresses:

- Key File Encryption Information
- Encryption and Database-Related Caché Facilities

10.6.1 Key File Encryption Information

Database encryption administrator names are stored in the clear in the key file. Database encryption administrator passwords are not stored; when entered, they are used, along with other data, to derive key-encryption keys. If someone can successfully guess a valid password, the password policy is too weak. Key-encryption keys are derived using the PBKDF2 algorithm with 512 bits of salt and 65,536 iterations, making dictionary and brute force attacks impractical. Nonetheless, database encryption key files should be securely stored separately from the encrypted databases if you want three-factor security (database, key file, password); for details on protecting data in this way, see the section "Protection from Unauthorized Access to Encrypted Data."

10.6.2 Encryption and Database-Related Caché Facilities

Caché database encryption protects database files themselves. Regarding related facilities in Caché:

- Caché online backups are not encrypted. To ensure that the Caché database is encrypted in a backup, it is recommended that you quiesce Caché and then perform a file system backup (as described in the section "External Backup" in the "Backup and Restore" chapter of the *Caché Data Integrity Guide*).
- In the write image journal (WIJ) file, the blocks for encrypted databases are encrypted. (For clustered systems, the same is true for the PIJ file.)
- The CACHETEMP and CACHE databases can optionally be encrypted. To provide encryption for CACHETEMP and CACHE, see the section "Configuring Caché Encryption Settings."
- You can optionally encrypt journal files; on a destination shadow server, this switch also encrypts all the journal files that it receives from its source production server. To encrypt journal files, see the section "Configuring Caché Encryption Settings."

11

SQL Security

Caché has both system-level security, and an additional set of SQL-related security features. The Caché SQL security provides an additional level of security capabilities beyond its database-level protections. Some of the key differences between SQL and system-level security are:

- SQL protections are more granular than system-level protections. You can define privileges for tables, views, and stored procedures.
- SQL privileges can be granted to users as well as to roles. System-level privileges are only assigned to roles.
- Holding an SQL privilege implicitly grants any related system privileges that are required to perform the SQL action. (Conversely, system-level privileges do not imply table-level privileges.) The different types of privileges are described in the "SQL Privileges and System Privileges" section.

11.1 SQL Privileges and System Privileges

To manipulate tables or other SQL entities through SQL-specific mechanisms, a user must have the appropriate SQL privileges. System-level privileges are not sufficient.

Note: Roles are shared by SQL and system level security: a single role can include both system and SQL privileges.

Consider the following example for an instance of Caché on a Windows machine:

- There is a class in the USER namespace called User.MyPerson. This class is projected to SQL as the SQLUser.MyPerson table.
- There is a user called Test, who belongs to no roles (and therefore has no system privileges) and who has all privileges on the SQLUser.MyPerson table (and no other SQL privileges).
- There is a second user, called Test2. This user is assigned to the following roles: **%DB_USER** (and so can read or write data on the USER database); **%SQL** (and so has SQL access through the %Service_Bindings service); and, through a custom role, has privileges for using the Console and **%Development**.

If the Test user attempts to read or write data in the SQLUser.MyPerson table through any SQL-specific mechanism (such as one that uses ODBC), the attempt succeeds. This is because Caché makes the Test user a member of the **%DB_USER** and **%SQL** role to establish the connection; this is visible in audit events that the connection generates, such as the %System/%Login/Login event. (If the Test user attempts to use the Terminal or Management Portal, these attempts fail, because the user lacks sufficient privilege for these.)

If the Test2 user attempts to read or write data in the SQLUser.MyPerson table through any SQL-specific mechanism (such as one that uses ODBC), the attempt fails because the user does not have sufficient privileges for the table. (If the Test2 user attempts to view the same data in the Terminal using object mechanisms, the attempt succeeds — because the user is sufficiently privileged for this type of connection.)

11.2 The SQL Service

The **%Service_SQL:Use** privilege controls a user's ability to connect using a Caché object or SQL client and then use SQL. When a user attempts to connect to Caché, the server determines whether the user holds any SQL-level privileges for the namespace. If the user holds at least one such privilege, then the server automatically adds two roles: the **%SQL** role, which has the **%Service_SQL:Use** privilege, and the implicit database role for the namespace's default database. As a result of this automatic role addition, it is not necessary for SQL users to hold any database privileges, because the server adds them automatically.

With the exception of this role addition at connect-time, no automatic role escalation occurs during the processing of an SQL statement. The user must hold the necessary system-level privileges when the SQL statement is executed.

Note that the **%Service_SQL:Use** privilege is only required to use SQL in a client/server configuration. For example, a user running an application that employs server-side embedded SQL requests does not require this permission.

The **%CREATE_TABLE** command is namespace-specific: granting a user this privilege for a specific namespace enables the user to create new tables in that namespace only.

11.2.1 CREATE USER

The SQL **CREATE USER** statement can be used to create Caché users. The newly created user has no roles.

Under some circumstances, a username can be implicitly used as an SQL schema name. This may pose problems if the username contains characters that are forbidden in an SQL identifier. For example, in a multiple domain configuration the username contains the "@" character.

Caché handles this situation differently depending on the setting of the *Delimited Identifiers* configuration parameter:

- If the use of delimited identifiers is enabled, no special processing occurs.
- If the use of delimited identifiers is disabled, then any forbidden characters are removed from the username to form a schema name. For example, the username "documentation@intersystems.com" would become the schema name "documentationintersystemscom".

This does not affect the value returned by the SQL **CURRENT_USER** function. It is always the same as *$USERNAME*.

11.2.2 Effect of Changes

Setting an SQL security value takes effect when the user next connects, not during that user's current session.

11.2.3 Required Privileges for Working with Tables

Creating a user in SQL, with the statement

```
CREATE USER <username> IDENTIFY BY <password>
```

is equivalent to performing the same action using the Management Portal. For the user to be able to work with a particular table, privileges for that table must be explicitly granted, such as with the Management Portal.

The minimum privilege required to work with a particular table is: any SQL privilege, like **SELECT**, on the relevant table. If the user has the right to perform a **SELECT** command, then this grants the ability to read and use but not write; analogously, **INSERT**, **UPDATE**, and **DELETE** provide those privileges.

12

System Management and Security

This chapter covers the following topics:

12.1 System Security Settings Page

The System Security Settings page (**System Administration** > **Security** > **System Security**) provides links to pages that configure the entire Caché instance for security. These pages are:

- System-Wide Security Parameters
- Authentication/CSP Session Options
- LDAP Options

12.2 System-Wide Security Parameters

This section describes security issues that affect an entire Caché instance. This includes the system-wide security parameters and handling sensitive data in memory images.

Caché includes a number of system-wide security parameters. You can configure these on the **System Security Settings** page (**System Administration** > **Security** > **System Security** > **System-wide Security Parameters**). These are:

- **Enable audit** — Turns auditing on or off. This check box performs the same action as the **Enable Auditing** and **Disable Auditing** links on the **Auditing** page (**System Administration** > **Security** > **Auditing**). For more information on auditing, see the chapter "Auditing." [Default is off]
- **Enable configuration security** — Specifies whether configuration security is on or off, as described in the section "Protecting Caché Configuration Information." [Default is off]
- **Default security domain** — Allows you to choose the instance's default security domain. For more information on security domains, see the section "Managing Caché Security Domains". [Default is the domain established during installation]
- **Inactive limit (0–365)** — Specifies the maximum number of days that a user account can be inactive, which is defined as the amount of time between successful logins. When this limit is reached, the account is disabled. A value of 0 (zero) means that there is no limit to the number of days between logins. [Default is 0 for minimal-security installations and 90 for normal and locked-down installations]
- **Invalid login limit (0-64)** — Specifies the maximum number of successive unsuccessful login attempts. After this limit is reached, either the account is disabled or an escalating time delay is imposed on each attempt; the action depends on the value of the **Disable account if login limit reached** field. A value of 0 (zero) means that there is no limit to the number of invalid logins. [Default is 5]
- **Disable account if login limit reached** — If checked, specifies that reaching the number of invalid logins (specified in the previous field) causes the user account to be disabled.
- **Password Expiration Days (0–99999)** — Specifies how frequently passwords expire and, therefore, how frequently users must change their passwords (in days). When initially set, specifies the number of days until passwords expire. A value of 0 (zero) means that the password never expires; however, setting this field to 0 does not affect users for whom the **Change Password on Next Login** field has been set. [Default is 0]

  CAUTION: This setting affects all accounts for the Caché instance, including those used by Caché itself. Until passwords are updated for these accounts, it may impossible for various operations to proceed and this may lead to unexpected results.

- **Password pattern** — Specifies the acceptable format of newly created passwords. See "Password Strength and Password Policies" for more information. [Default for an instance with Minimal and Normal security is 3.32ANP; default for a locked-down instance is 8.32ANP.]
- **Password validation routine** — Specifies a user-provided routine (or entry point) for validating a password. See the **PasswordValidationRoutine** method in the Security.System class for more information.
- **Role required to connect to this system** — If set to an existing role, specifies that a user must be a member of this role (as a login role) in order to log into the system.

  If you are using LDAP authentication or OS-based LDAP authorization, InterSystems strongly recommends that you create a role that is required to connect and that you specify its name in this field. For more information, see the "Setting Up a Role Required for Login" section of the "Using LDAP" chapter.
- **Enable writing to percent globals** — Specifies whether write access to percent globals is implicitly granted to all users; if not checked, write access is controlled by normal security mechanisms. For more information on the percent globals and CACHESYS (the database that holds them), see the section "CACHESYS, the Manager's Database" in the chapter "Assets and Resources." [Default is controlled by normal security mechanisms.]
- **Allow multiple security domains** — Specifies whether there is support for multiple Caché security domains. For more information on security domains, see the section "Managing Caché Security Domains." [Default is a single domain]

- **Superserver SSL/TLS Support** — Specifies if the superserver supports or requires the use of SSL/TLS for client connections.

  Important: Before you can configure the superserver to use SSL/TLS, there must be an existing configuration called `%SuperServer`. For more information about using SSL/TLS with the Caché superserver, see "Configuring the Caché Superserver to Use SSL/TLS."

  Options are:

  – **Disabled** — The superserver refuses client connections that use SSL/TLS. (That is, it only accepts client connections that do *not* use SSL/TLS.)

  – **Enabled** — The superserver accepts but does not require SSL/TLS.

  – **Required** — The superserver requires client connections to use SSL/TLS.

12.2.1 Protecting Sensitive Data in Memory Images

Certain error conditions can cause the contents of a process's memory to be written to a disk file, known as a "core dump." This file contains copies of all data that was in use by the process at the time of the dump, including potentially sensitive application and system data. This can be prevented by disallowing core dumps on a system-wide basis. The method for disallowing core dumps varies according to the operating system in use; for details, consult the documentation of your operating system.

12.3 Authentication Options

The **Authentication/CSP Sessions Options** page (**System Administration** > **Security** > **System Security** > **Authentication/CSP Options**) allows you to enable or disable authentication mechanisms for the entire Caché instance:

- If an authentication mechanism is disabled for the entire Caché instance, then it is not available for any service.
- If an authentication mechanism is enabled for the entire Caché instance, then it is available for all the services that support it. To enable the authentication mechanism for a particular service, use the **Edit Service** page for that property; this page is available by selecting the service from the **Services** page (**System Administration** > **Security** > **Services**).

  Note: Not all services support all mechanisms.

The authentication options are:

- Allow Unauthenticated access — Users may connect without authenticating. (If login dialog appears, the user can leave the **Username** and **Password** fields blank and click **OK** to log in.)
- Allow Operating System authentication — Caché uses the operating system's user identity to identify the user.
- Allow Password authentication — Caché uses its own native tools to authenticate a username and password that are registered with it. This mechanism is also known as "Caché login."
- Allow Delegated authentication — Caché delegates the process of authentication to a user-defined function.
- Allow Kerberos authentication — Caché performs authentication using Kerberos.
- Allow LDAP authentication — Caché uses an available LDAP database to authenticate users.
- Allow LDAP cache credentials authentication — Caché keeps a cached copy of LDAP credentials so that it can authenticate LDAP users if the LDAP database becomes unavailable.

- Allow creation of Login Cookies — Caché uses cookies that are shared among enabled web applications to authenticate users, so that they do not need to enter a username and password when first using a new application.
- Login Cookie expire time (secs) — The duration of a login cookie, in seconds. This field is only relevant if Login Cookies are enabled for the instance.
- Allow Two-factor Time-based One-time Password authentication — Caché provides a verification code to the user via an authentication device or an app that runs on the user's phone; the user then enters the code to complete the authentication process. If selected, the **Authentication/CSP Session Options** page displays the fields for configuring two-factor authentication.
- Allow Two-factor SMS text authentication — Caché provides a security code to the user via a mobile phone text message; the user then enters the code to complete the authentication process. If selected, the **Authentication/CSP Session Options** page displays the fields for configuring two-factor authentication.

If there are multiple authentication options, Caché uses cascading authentication. For more information on authentication, see the chapter "Authentication."

12.4 The Secure Debug Shell

Caché includes the ability to suspend a routine and enter a shell that supports full debugging capabilities (as described in the "Command-line Routine Debugging" chapter of *Using Caché ObjectScript*). Caché also includes a secure debug shell, which has the advantage of ensuring that users are prevented from exceeding or circumventing their assigned privileges.

The secure debug shell helps better control access to sensitive data. It is an environment that allows users to perform basic debugging, such as stepping and displaying variables, but does not allow them to do anything that changes the execution path or results of a routine. This protects against access that can lead to issues such as manipulation, malicious role escalation, and the injection of code to run with higher privileges.

The secure debug shell starts when a **Break** command is executed, a breakpoint or watchpoint is encountered, or an uncaught error is issued.

Within the secure debug shell, the user cannot invoke:

- Any command that can modify a variable.
- Any function that can modify a variable.
- Any command that can call other routines.
- Any command that affects the flow of the routine or the environment.

Within the secure debug shell, when a user attempts to invoke a restricted command or function, Caché throws a <COMMAND> or <FUNCTION> error, respectively.

12.4.1 Enabling Use of the Secure Shell

By default, users at the debug prompt maintain their current level of privileges. To enable the secure shell for the debug prompt and thereby restrict the commands that the user may issue, the user must hold the **%Secure_Break:Use** privilege (the **Use** permission for the **%Secure_Break** resource). To give a user this privilege, make the user a member of a role which includes the **%Secure_Break:Use** privilege, such as the predefined **%SecureBreak** role.

12.4.2 Restricted Commands and Functions

This section lists the restricted activities within the secure debug shell:

12.4.2.1 Restricted ObjectScript Commands

The following are the restricted ObjectScript commands for the secure debug shell:

- **CLOSE**
- **DO**
- **FOR**
- **GOTO** with an argument
- **KILL**
- **LOCK**
- **MERGE**
- **OPEN**
- **QUIT**
- **READ**
- **RETURN**
- **SET**
- **TCOMMIT**
- **TRESTART**
- **TROLLBACK**
- **TSTART**
- **VIEW**
- **XECUTE**
- **ZALLOCATE**
- **ZDEALLOCATE**
- **ZINSERT**
- **ZKILL**
- **ZREMOVE**
- **ZSAVE**
- user commands except **ZW** and **ZZDUMP**

12.4.2.2 Restricted ObjectScript Functions

The following are the restricted ObjectScript functions for the secure debug shell:

- **$CLASSMETHOD**

- **$COMPILE**
- **$DATA(,var)** — two-argument version only
- **$INCREMENT**
- **$METHOD**
- **$ORDER(,,var)** — three-argument version only
- **$PROPERTY**
- **$QUERY(,,var)** — three-argument version only
- **$XECUTE**
- **$ZF**
- **$ZSEEK**
- **$ZUTIL**
- any extrinsic function

12.4.2.3 Restricted Object Constructions

No method or property references are allowed. Property references are restricted because they could invoke a **propertyGet** method. Some examples of the object method and property syntax constructions that are restricted are:

- **#class(classname).ClassMethod()**
- **oref.Method()**
- oref.Property
- **$SYSTEM.Class.Method()**
- **..Method()**
- ..Property

Note: Even without passing a variable by reference, a method can modify public variables. Since a property reference could invoke a **propGet** method, no property access is allowed.

12.4.2.4 Restricted MultiValue Commands

The following are the restricted MultiValue commands for the secure debug shell:

- **MV**
- **MVCALL**
- **MVDIM**

12.5 Password Strength and Password Policies

Caché allows you to specify requirements for user passwords by supplying a string of the form:

```
X.Y[ANP]
```

where

- *X* is the minimum number of characters in the password.
- *Y* is the maximum number of characters in the password.
- *A*, *N*, and *P* specify whether *A*lphabetic characters, *N*umeric characters, and *P*unctuation characters are permitted in the password.

These rules are based on the ObjectScript pattern matching functionality. This functionality is described in the "Pattern Matching" section of the "Operators and Expressions" chapter of *Using Caché ObjectScript*.

Note: The value for this parameter does not affect existing passwords.

12.5.1 Suggested Administrator Password Strength

Ideally, administrator passwords should be a random mixture of uppercase and lowercase alphabetic characters, numerals, and punctuation. InterSystems strongly recommends a minimum password length of 12 such random characters.

12.6 Protecting Caché Configuration Information

Caché configuration information is stored in a text file outside of Caché. This file is known as a Caché parameter file and often referred to as a cache.cpf file. Because this file can be modified while Caché is not running, Caché controls the ability to start a system with a modified cache.cpf.

To protect your instance against intentional or accidental misconfiguration, check the **Configuration Security** box to "on". If Caché startup detects that the Caché parameter file has been modified outside the control of the Management Portal since the last time Caché was started, Caché startup requests a username and password to validate the changes. The username supplied must have **`%Admin_Manage:Use`** privileges. If an appropriate username and password cannot be provided, Caché allows the operator to choose as follows:

1. Re-enter the username and password.
2. Start using the last known good configuration.
3. Abort startup.

If the operator chooses option 2, Caché renames the parameter file that was invoked at startup (*file*.cpf) with the suffix _rejected (*file*.cpf_rejected). Caché then overwrites the *file*.cpf with the last known good configuration (_LastGood_.cpf) and starts up using this configuration.

Note: The protections for the cache.cpf file are not a substitute for operating-system–level security. It is recommended that you protect the configuration file by strictly limiting the ability of users to modify it, at the operating-system level.

For more information on the configuration file generally, see the *Caché Parameter File Reference*.

12.7 Managing Caché Security Domains

Caché security domains provide a grouping of users that corresponds to Kerberos realms and Windows domains. If your instance is using Kerberos, its Caché domain corresponds to a Kerberos realm. If you are using a Windows domain, this also corresponds to a Kerberos realm.

While a security domain name often takes the form of an Internet domain name, there is no requirement that it do so. A security domain name can contain any character except "@".

12.7.1 Single and Multiple Domains

You can configure Caché for either a single-domain or multiple-domain configuration.

For an instance with a single domain:

- The *$USERNAME* variable does not include the domain name.
- System utilities do not show the domain name when displaying usernames.
- It is prohibited to specify a username from any domain other than the default domain (described in the following section).

For an instance with multiple domains:

- The *$USERNAME* variable includes the domain name.
- System utilities show the domain name when displaying usernames.

In a multiple-domain configuration, a fully-qualified user identifier consists of a username, an at sign ("@"), and a domain name, such as, "info@intersystems.com".

To specify support for a single domain or multiple domains, use the **Allow multiple security domains** field of the **System-wide Security Parameters** page of the Management Portal (**System Administration** > **Security** > **System Security** > **System-wide Security Parameters**), described in the System-wide Security Parameters section of this chapter.

12.7.2 The Default Security Domain

Each instance has a default security domain. This is the domain assumed for any username where no domain is specified. For example, if the default domain is "intersystems.com", the user identifiers "info" and "info@intersystems.com" are equivalent. When Caché is installed, it uses the local domain name to provide an initial value for the parameter.

For instances with multiple security domains, you can select a new default security domain using the **Default Security Domain** field of the **System-wide Security Parameters** page (**System Administration** > **Security** > **System Security** > **System-wide Security Parameters**), described in the System-wide Security Parameters section of this chapter.

12.7.3 Listing, Editing, and Creating Domains

The **Security Domains** page (**System Administration** > **Security** > **Security Domains**) provides a table that lists the existing security domains for a Caché instance.

For each domain, the table has:

- The domain's name.
- The domain's description.
- An **Edit** button, which allows you to edit the domain's description (but not its name). Since you cannot change a domain's name, create a new domain with the preferred name and then delete the existing domain.

- A **Delete** button, which, after prompting, allows you to remove a domain from the instance.

The page also has a **Create New Domain** button. Selecting this displays the **Edit Domain** page which accepts a domain name and an optional domain description. After entering this information, select **Save** to create the domain.

12.8 Security Advisor

To assist system managers in securing a Caché system, Caché includes a Security Advisor. This is a Web page that shows current information related to security in the system configuration. It recommends changes or areas for review, and provides links into other system management Web pages to make the recommended changes.

Important: The Security Advisor provides general recommendations, but does not have any knowledge of an instance's needs or requirements. It is important to remember that each Caché instance has its own requirements and constraints, so that issues listed in the Security Advisor may not be relevant for your instance; at the same time, the Security Advisor may not list issues that are of high importance for you. For example, the Security Advisor exclusively recommends that services use Kerberos authentication, but, depending on your circumstances, authentication through the operating system, Caché login, or even unauthenticated access may be appropriate.

There are some general features in the Security Advisor:

- **Details** button — Each section has a **Details** button. Selecting this button displays the page for managing that aspect of Caché regulated by the section.
- **Name** button — Each named item in each section is a link. Selecting one of these items displays the page for managing that item.
- **Ignore** check box — Each named item in each section has an associated **Ignore** check box. Selecting this grays out the line for the specified item. The line does not appear if Caché is set up according to the Security Advisor's recommendations, whether or not the **Ignore** check box is selected.

12.8.1 Auditing

This section displays recommendations on auditing itself and on particular audit events:

- Auditing should be enabled — Auditing creates a record that can provide forensic information after any notable or unusual system events.
- Auditing for this event type should be enabled — Auditing particular events can provide more specific information about various topics. Specifically, since the events noted when not enabled are:
  - The DirectMode event — Auditing this event can provide information about connections to Caché that give users significant privileges.
  - The Login event — Auditing this event can provide information questionable logins.
  - The LoginFailure event — Auditing this event can provide information about attempts to gain inappropriate access to the system.

12.8.2 Services

This section displays recommendations on Caché services. For each service, depending on its settings, the Security Advisor may address any of the following issues:

- Ability to set % globals should be turned off — Since % globals often hold system information, allowing users to manipulate these globals can result in serious, pervasive, and unpredictable effects.
- Unauthenticated should be off — Unauthenticated connections give all users, including the unidentified UnknownUser account, unregulated access to Caché through the service.
- Service should be disabled unless required — Access through any service monitored by the Security Advisor can provide an undue level of system access.
- Service should use Kerberos authentication — Access through any other authentication mechanism does not provide the maximum level of security protection.
- Service should have client IP addresses assigned — By limiting the number of IP addresses from which connections are accepted, Caché may be able to more tightly oversee the connections to it.
- Service is Public — Public services give all users, including the unidentified **`UnknownUser`** account, unregulated access to Caché through the service.

12.8.3 Roles

This section displays recommendations for all roles that hold possibly undue privileges; other roles are not listed. For each role, the Security Advisor may address any of the following issues:

- This role holds privileges on the Audit database — Read access to the Audit database may expose audited data inappropriately; Write access to the Audit database may allow the inappropriate insertion of data into that database.
- This role holds the **`%Admin_Secure`** privilege — This privilege can allow for the establishing, modifying, and denying access of users to assets; it also allows the modification of other security-related features.
- This role holds Write privilege on the %CACHESYS database — Write access to the %CACHESYS database may allow the compromise of system code and data.

12.8.4 Users

This section displays recommendations related to users generally and for individual user accounts. In this area, the Security Advisor may address any of the following issues:

- At least 2 and at most 5 users should have the **`%All`** role — Too few users holding **`%All`** can lead to access problems in an emergency; too many users holding it can open the system to compromise
- This user holds the %All role — Explicitly announcing which users hold **`%All`** can help eliminate any who hold it unnecessarily.
- UnknownUser account should not have the **`%All`** role — A system cannot be properly secured if anonymous users have all privileges. While this is part of any instance with a Minimal security level, such an instance is not properly secured by design.
- Account has never been used — Unused accounts provide an attractive point of entry for those attempting to gain unauthorized access.
- Account appears dormant and should be disabled — Dormant accounts (those that have not been used for over 30 days) provide an attractive point of entry for those attempting to gain unauthorized access.
- Password should be changed from default password — This is a commonly attempted point of entry for those attempting to gain unauthorized access.

12.8.5 CSP, Privileged Routine, and Client Applications

Each application type has its own section, which makes it simpler to review details for each application type. These sections display recommendations related to access to and privileges granted by applications. In this area, the Security Advisor notes the following issues:

- Application is Public — Public applications give all users, including the unidentified **UnknownUser** account, unregulated access to the data associated with the application and actions that the application supports. This is even more notable if the application also grants the **%All** role, either conditionally or absolutely.
- Application conditionally grants the **%All** role — A system cannot be properly secured if users have the possibility of holding all privileges. This is even more notable if the application is also public.
- Application grants the **%All** role — A system cannot be properly secured if users have all privileges. This is even more notable if the application is also public.

12.9 Effect of Changes

When you make changes to various security settings, the amount of time for these to take effect are as follows:

- Changes to user properties, such as the roles assigned to the user, are effective with the next login for that user. They have no effect on processes that are already running.
- Changes to services, such as whether a service is enabled or authentication is required, are effective for future connection attempts. Existing connections are not affected.
- Changes to role definitions are effective immediately for any subsequent privilege checks. These affect database resources immediately, because they are checked for each database access. For services and applications, they are effective with subsequent connection attempts or application initiations.

Note: The times listed here are the latest times that changes take effect; in some cases, changes may be effective earlier than indicated.

12.10 Emergency Access

Caché provides a special emergency access mode that can be used under certain dire circumstances, such as if there is severe damage to security configuration information or if no users with the **%Admin_Manage:Use** or **%Admin_Security:Use** privileges are available (that is, if all users are locked out). Although Caché attempts to prevent this situation by ensuring that there is always at least one user with the **%All** role, that user may not be available or may have forgotten the password.

When Caché is running in emergency access mode, only a single user (called the *emergency user*) is permitted. This username does not have to be previously defined within Caché. In fact, even if the username is defined in Caché, the emergency user is conceptually a different user. The emergency username and password are only valid for the single invocation of emergency mode.

Other important points about emergency access mode:

- **%Service_Console**, **%Service_Terminal**, and **%Service_CSP** are the only services enabled.
- There is only access using Caché login — no other authentication mechanism is supported.

- For the web applications that control the Portal (/csp/sys and /csp/sys/*), the standard login page (%CSP.Login.cls) is used during emergency access even if there is a custom login page available; this ensures that the emergency user has access to the Portal, since a custom login page may prevent authentication from occurring. For other web applications, if there is a custom login page, then that page is used during emergency login.
- Two-factor authentication is disabled. This avoids any situation where two-factor authentication might prevent the emergency user from being able to authenticate.

12.10.1 Invoking Emergency Access Mode

To start Caché in emergency access mode, you must have the appropriate operating-system privileges:

- On Windows systems, the user must be a member of the Administrators group.
- On UNIX® and MacOS systems, the user must be root or the owner of the instance.

Caché performs authentication by checking operating-system-level characteristics.

12.10.1.1 Invoking Emergency Access Mode on Windows

To start Caché in emergency access mode:

1. Start a command prompt, running it as an administrator. This can either be:
   - The Windows Command Prompt program. Right-click the **Command Prompt** choice in the menu and then choose **Run as Administrator**.
   - The Windows PowerShell. While you can run this as either an administrator or a user without extra privileges, this procedure assumes that you are running as an administrator; to run as a user without extra privileges, use the `-verb runas` argument when you invoke the command, which is described in PowerShell documentation.
2. Go to the bin directory for your Caché installation.
3. In that directory, invoke Caché at the command line using the appropriate switch and passing in the username and password for the emergency user. This depends on the command prompt that you are using:
   - For the Windows Command prompt, the command is:

     ```
     ccontrol start <instance> /EmergencyId=<username>,<password>
     ```

     This starts an emergency-mode Caché session with only one allowed user where:
     - <*instance*> specifies the instance being started in emergency mode
     - <*username*> is the sole user of the system
     - <*password*> is that user's password
   - For the Windows PowerShell, the command is:

     ```
     start-process .\ccontrol.exe -ArgumentList "start <instance> /EmergencyId=<username>,<password>"
     ```

     This starts an emergency-mode Caché session with only one allowed user where:
     - <*instance*> specifies the instance being started in emergency mode
     - <*username*> is the sole user of the system
     - <*password*> is that user's password

Note: On Windows, unlike other operating systems, the `EmergencyId` switch is preceded by a slash ("/").

For example, at the instance MyCache, to start Caché in emergency mode with user Eugenia with the password 52601, the command would be:

```
ccontrol start MyCache /EmergencyId=Eugenia,52601
```

The only user who can then log in is the emergency user, using the appropriate password, such as:

```
Username: Eugenia
Password: *****
Warning, bypassing system security, running with elevated privileges
```

Once Caché has started, you can start the Terminal from the Caché cube or run any CSP application. This provides access to the Management Portal and all character-based utilities. Using this access, you can change any settings as necessary and then restart Caché in its normal mode.

12.10.1.2 Invoking Emergency Access Mode on UNIX® and Mac OS

To start Caché in emergency access mode, invoke Caché at the command line using the appropriate switch and passing in the username and password for the emergency user:

```
./ccontrol start <cache-instance-name> EmergencyId=<username>,<password>
```

This starts an emergency-mode Caché session with only one allowed user where:

- <cache-instance-name> specifies the instance being started in emergency mode
- <username> is the sole user of the system
- <password> is <username>'s password

Note: If going from one of these operating systems to Windows, remember that on Windows only, the `EmergencyId` switch is preceded by a slash ("/").

For example, at the instance MyCache, to start Caché in emergency mode with user Eugenia with the password 5262001, the command would be:

```
./ccontrol start MyCache EmergencyId=Eugenia,52601
```

The only user who can then log in is the emergency user, using the appropriate password, such as:

```
Username: Eugenia
Password: *****
Warning, bypassing system security, running with elevated privileges
```

Once Caché has started, you can run Terminal or any CSP application. This provides access to the Management Portal and all character-based utilities. Using this access, you can change any settings as necessary and then restart Caché in its normal mode.

12.10.2 Emergency Access Mode Behavior

In emergency access mode, Caché has the following constraints and behaviors:

- The emergency user is the only permitted user. Any attempt by another user to log in will fail.
- The emergency user has the **%ALL** role.

- Console, Terminal and CSP are the only services that are enabled. All other services are disabled. This does not affect the enabled or disabled status of services when Caché starts in non-emergency mode; only the current (emergency), in-memory information about services is affected.
- For the enabled services, only authenticated access is permitted. Caché uses its own password authentication for the services, where the emergency access username and password must be used.
- After emergency access login, Caché attempts to audit all events for the active process; Caché start-up proceeds even if this is not possible. Login failures in emergency access mode are not audited.
- The emergency user can make changes to the Caché configuration, but these changes are not activated until the next time that Caché is started in normal (not emergency) mode. This is in contrast to the normal operation of Caché, in which configuration changes are primarily activated without restarting Caché.

13

Using SSL/TLS with Caché

This chapter describes the use of Caché with SSL (Secure Sockets Layer) and TLS (Transport Layer Security), its successor. Caché supports the use of SSL/TLS to secure connections of several types:

- From various client applications that interact with the Caché superserver (including ODBC, JDBC, and Studio). This also includes connections from Caché shadow destinations to Caché shadow sources.
- From Telnet clients that interact with the Telnet server.
- For use with TCP connections where a Caché instance is the client or server (or a Caché instance is at each end).

As a server, Caché accepts connections and establishes the use of SSL; as a client, Caché is able to connect to servers that require the use of SSL. In all cases, Caché uses what is called an SSL/TLS *configuration*, which specifies the various characteristics of a Caché instance as part of an SSL/TLS connection.

This chapter has the following sections:

13.1 About SSL/TLS

SSL/TLS provides strong protection for communication between pairs of entities. It allows you to perform authentication, data integrity protection, and data encryption.

SSL was created at Netscape in the mid nineteen-nineties. Version 3.0, which is still in use, was released in 1996. TLS was created as a standardization of SSL 3.0 and version 1.0 was released in 1999. The current version of TLS is 1.2. Among the supported versions of SSL/TLS for Caché, InterSystems recommends the use of the latest version available.

The process by which an SSL/TLS connection is established between two entities is known as the SSL/TLS handshake, and it uses a client/server model. Completion of the handshake means that, according to the requirements of the client and the server:

- The client has authenticated the server.
- The server has authenticated the client. (If the client and the server have both authenticated each other, this known as mutual authentication.)
- The client and server have agreed upon session keys. (*Session keys* are the keys for use with a symmetric-key algorithm that allow the entities to protect data during subsequent communications.)
- Subsequent communication can be encrypted.
- The integrity of subsequent communication can be verified.

The *ciphersuites* of the client and server specify how these activities occur as part of the handshake or are supported for a protected connection. Specifically, a peer's ciphersuites specify what features and algorithms it supports. The client proposes a set of possible ciphers for use; from among those proposed, the server selects one. (If there are no common ciphers between the client and server, the handshake fails.)

To perform the handshake, SSL/TLS typically uses public-key cryptography (though it can use other means, such as the Diffie-Hellman protocol). With public-key cryptography, each peer (either the client or the server) has a public key and a private key. The private key is a sensitive secret value and the public key is a widely published value; typically, the public key is encapsulated in a certificate, which also contains identifying information about the holder, such as a name, organization, location, issuer validity, and so on. For Caché, an SSL/TLS *configuration* (described in the section "About Configurations") specifies a named set of SSL/TLS-related values, including a certificate file, a private key file, and an optional set of ciphersuites.

If successful, the handshake creates session keys that are used to protect subsequent communications.

While Caché and applications require various interactions with SSL/TLS, the end-user typically has no such direct interactions. For example, a browser uses SSL/TLS to establish a secure connection with a specified web site by requiring that the site (the server, in this case) authenticate itself to the browser (which occurs unbeknownst to the browser's user) and the lock icon that appears in the browser is designed to indicate that SSL/TLS is protecting the connection.

13.2 About Configurations

Caché can support multiple *configurations*, each of which specifies a named set of SSL/TLS-related values. All its existing configurations are activated at startup. When you create a new configuration from the Management Portal, Caché activates it when you save it. The page for managing SSL/TLS configurations is the **SSL/TLS Configurations** page (**System Administration** > **Security** > **SSL/TLS Configurations**).

This section covers the following topics:

- Creating or Editing an SSL/TLS Configuration
- Deleting a Configuration
- Reserved Configuration Names

13.2.1 Creating or Editing an SSL/TLS Configuration

The page for creating or editing an SSL/TLS configuration is the **SSL/TLS Configurations** page (**System Administration** > **Security** > **SSL/TLS Configurations**). To create a new configuration, click **Create New Configuration**, which displays the **New SSL/TLS Configuration** page; to edit an existing configuration, click **Edit** to the right of the name of the configuration. (You can also create a new set of configurations for a mirror member by clicking **Create Configurations for Mirror**; for more information on mirroring and SSL/TLS, see "Configuring Caché to Use SSL/TLS with Mirroring."

When creating or editing an SSL/TLS configuration, the following fields are available:

- **Configuration Name** — The string by which the configuration is identified. Configuration names can contain any alphanumeric character and any punctuation except the "|" character. If you are creating a configuration for the Caché superserver, the configuration name must be `%SuperServer`; for more information about this topic, see "Configuring the Caché Superserver to use SSL/TLS."
- **Description** — Any text.
- **Enabled** — Whether or not the configuration is available for activation.
- **Type** — Intended purpose for this configuration, where the choice is **Client** or **Server**; the default is **Client**. Clients initiate the use of the protocol and servers respond to the initial request. (The Caché superserver uses a server configuration; SSL/TLS clients, such as a shadow destination, use a client configuration.) The value chosen for this field determines:
  - Whether the next field is the **Server certificate verification** or **Client certificate verification** field. If the configuration is for a client, the next field is the **Server certificate verification**, which specifies the verification that may be required for the certificate of any server to which the client is connecting; if the configuration is for a server, the next field is the **Client certificate verification**, which specifies the verification that may be required for the certificate of any client that attempts to connect to the server.
  - The behavior of the **File containing trusted Certificate Authority certificate(s)** field.
- **Server certificate verification** or **Client certificate verification** — Specifies whether or not the configuration requires the verification of the peer to which it is connecting.

  A configuration for a client must have a specified **Server certificate verification** and supports possible values of:
  - **None** — Continues under all circumstances.
  - **Require** — Continues only if certificate verification succeeds.

  A configuration for a server must have a specified **Client certificate verification** and supports possible values of:
  - **None** — Specifies that the server neither requests nor requires a client certificate.
  - **Request** — Allows the client to provide or not provide a certificate. If the client provides no certificate, then authentication proceeds; if the client provides a certificate and verification fails, then authentication fails.
  - **Require** — Specifies that the client must provide a certificate; authentication depends on verification of the certificate.
- **File containing trusted Certificate Authority certificate(s)** — The path and name of a file that contains the X.509 certificate(s) in PEM format of the Certificate Authority (CA) or Certificate Authorities that this configuration trusts. The configuration uses the certificates of the trusted CA(s) to verify peer certificates. Typically, a production system uses certificates from commercial CAs with publicly available certificates.

  Regarding this field, note the following:
  - You can specify the path of the file as either an absolute path or as a path relative to the *<install-dir>*`/mgr/` directory.

- For a server configuration with a **Client certificate verification** value of **None**, this field is not available, since there is no peer verification.
- Certificates from the Windows Certificate Export Wizard must be in base-64-encoded X.509 format, not the default of DER-encoded binary X.509.
- With mirroring, the configuration must also have enough information to verify its own certificate.

For information on how these certificates are used, see the section "Establishing the Required Certificate Chain." For information on file names for these certificates and how to verify a certificate chain, see the OpenSSL documentation on the **verify** command.

- **This client's credentials** or **This server's credentials** — The files (if needed) containing the X.509 certificate and private key for the local configuration:
  - **File containing this client's certificate** or **File containing this server's certificate** — The full location of the configuration's own X.509 certificate(s), in PEM format. This can be specified as either an absolute or a relative path. This can include a certificate chain. For information on how this is used for authentication, see the section "Establishing the Required Certificate Chain." (Note that certificates from the Windows Certificate Export Wizard must be in base-64 encoded X.509 format, not the default of DER encoded binary X.509.)
  - **File containing associated private key** — The full location of the configuration's private key file, specified as either an absolute or relative path.
  - **Private key type** — The algorithm used to generate the private key, where valid options are **DSA** (Digital Signature Algorithm) and **RSA** (Rivest, Shamir, and Adleman, for the algorithm's inventors).
  - **Private key password** — An optional password for encrypting and decrypting the configuration's private key.

    Note: If the private key is password-protected and you do not enter a value here, Caché cannot confirm that the private key and the certificate's public key match each other; this can result in mismatched keys being saved as a key pair.

  - **Private key password (confirm)** — A retyping of the optional password to ensure that it is the intended string.
- **Cryptographic settings**:
  - **Protocols** — Those communications protocols that the configuration considers valid, where the choices are one or more of SSLv3, TLSv1.0, TLSv1.1, and TLSv1.2. The protocols that are enabled by default are TLSv1.1 and TLSv1.2.

    Note: InterSystems recommends the use of TLSv1.1 or TLSv1.2 only. InterSystems recommends that you do not use SSLv3 or earlier versions.

  - **Enabled ciphersuites** — The set of ciphersuites used to protect communications between the client and the server. See the "Enabled Ciphersuites Syntax" section for more information on this topic.

Note: The required fields vary, depending on whether the configuration is to be a client or server and on the desired features. Not all fields are required for all SSL/TLS configurations.

To complete the process of creating or editing a configuration, use the following buttons, which appear at the top of this page:

- **Save** — Dismisses the dialog, saving and then activating the configuration. This saves changes to an existing configuration or the configuration being created.
- **Cancel** — Dismisses the dialog without saving changes to an existing configuration or without saving a configuration being created.

- **Test** — Checks for valid configuration information. If the configuration's role is as a client, selecting this button also prompts for a server (its host name, not its URL) and a port number; Caché then tries to establish a test connection to that server. (This button is not available when creating a server configuration.)

  Note: The **Test** button may not be able to successfully connect with all TLS servers, even if the configuration has no errors. This is because the connection test performs a TLS handshake followed by an HTTP request. If the server expects a StartTLS message before the handshake (such as for use with LDAP, SMTP, FTPS, or another protocol), then the test fails, even though the actual SSL/TLS connection to the server succeeds.

13.2.1.1 Required Information for Certificates

When a client authenticates a server, the client needs to have the full certificate chain from the server's own certificate to the server's trusted CA certificate — including all intermediaries between the two.

There is an issue when setting up a server SSL/TLS configuration and the server's trusted CA certificate is not a root certificate. In order for authentication to work properly, the client needs to have access to all the certificates that constitute the certificate chain from the server's personal certificate to a self-signed trusted CA certificate. This chain can be obtained from the combination of the server's certificate file (sent during the handshake) and the client's trusted CA certificate file. The self-signed trusted root CA certificate must be in the client's CA certificate file, and the server's personal certificate must be the first entry in the server's certificate file. Other certificates may be divided between the two locations. The same constraints apply in reverse when a client authenticates to a server.

Regarding certificate formats, note that certificates from the Windows Certificate Export Wizard must be in base-64 encoded X.509 format, not the default of DER encoded binary X.509.

13.2.1.2 Enabled Ciphersuites Syntax

A configuration only allows connections that use its enabled ciphersuites. To specify enabled ciphersuites, you can either:

- Provide a list of individual ciphersuites, using each one's name.
- Use OpenSSL syntax for specifying which ciphersuites to enable and disable.

Both the list of ciphersuite names and the syntax for specifying enabled ciphersuites is described on the ciphers(1) man page at openssl.org. This syntax allows you to specify guidelines for requiring or proscribing the use of various features and algorithms for a configuration.

The default set of cipher suites for a Caché configuration is `ALL:!aNULL:!eNULL:!EXP:!SSLv2` which breaks down into the following group of colon-separated statements:

- `ALL` — Includes all cipher suites except the eNULL ciphers
- `!aNULL` — Excludes ciphers that do not offer authentication
- `!eNULL` — Excludes ciphers that do not offer encryption
- `!EXP` — Excludes export-approved algorithms (both 40- and 56-bit)
- `!SSLv2` — Excludes SSL v2.0 cipher suites

13.2.1.3 A Note on Caché Client Applications Using SSL/TLS

For certain activities, you can use Caché instances to support client applications that interact with the Caché superserver. For example, when using SSL/TLS to protect a shadowing connection, a Caché instance serving as a shadow destination would be an SSL/TLS client.

When using client applications that interact with the Caché superserver using SSL/TLS, the following aspects of the configuration require particular attention:

- **Configuration Name** — While there are no constraints on the name of clients, this information is required to configure the connection.
- **Type** — Because the instance is serving with an SSL/TLS client, the type must be specified to be of type **Client**.
- **Ciphersuites** — The specified ciphersuites need to match those required or specified by the server.

It is also necessary to ensure that the client and the server are configured so that each may verify the other's certificate chain, as described in the section "Establishing the Required Certificate Chain."

13.2.2 Deleting a Configuration

The page for deleting an SSL/TLS configuration is the **SSL/TLS Configurations** page (**System Administration** > **Security** > **SSL/TLS Configurations**). To delete a configuration, click **Delete** to the right of the name of the configuration. The Portal prompts you to confirm the action.

13.2.3 Reserved Configuration Names

Caché reserves several SSL/TLS configuration names for use with particular features. When using such a feature, you must use the reserved configuration name(s). The reserved configuration names are:

- `%MirrorClient` — For a mirror member when acting as an SSL/TLS client. For more information on mirroring and SSL/TLS, see "Configuring Caché to Use SSL/TLS with Mirroring."
- `%MirrorServer` — For a mirror member when acting as an SSL/TLS server. For more information on mirroring and SSL/TLS, see "Configuring Caché to Use SSL/TLS with Mirroring."
- `%SuperServer` — For the Caché superserver when accepting connections from other Caché components. For more information about configuring the superserver to use SSL/TLS, see the next section.
- `%TELNET/SSL` — For the Windows Telnet server when accepting connections protected by SSL/TLS. For more information on mirroring and Telnet, see "Configuring the Caché Telnet Server for SSL/TLS."

Important: For SSL/TLS to function properly, you must use the exact case for each configuration name as it appears here.

13.3 Configuring the Caché Superserver to Use SSL/TLS

To use SSL/TLS for communications among components of Caché, configure the Caché superserver to use SSL/TLS. To do this, the procedure is:

1. From the Management Portal home page, go to the **SSL/TLS Configurations** page (**System Administration** > **Security** > **SSL/TLS Configurations**).
2. On the **SSL/TLS Configurations** page, select **Create New Configuration**, which displays the **New SSL/TLS Configuration** page.
3. On the **New SSL/TLS Configuration** page, create an SSL/TLS server configuration with a configuration name of `%SuperServer` (using the exact case as specified here). For details about creating an SSL/TLS configuration, see the section "Creating or Editing an SSL/TLS Configuration."
4. On the **System-wide Security Parameters** page (**System Administration** > **Security** > **System Security** > **System-wide Security Parameters**), for the **Superserver SSL/TLS Support** field, choose **Enabled**. This specifies that the superserver supports (but does not require) SSL/TLS-secured connections.

Note: If you wish to configure the superserver to require SSL/TLS-secured connections, first specify that SSL/TLS is simply enabled.

5. Set up clients to use SSL/TLS as appropriate (see the next section).

13.4 Configuring the Caché Telnet Service to Use SSL/TLS

The Caché Telnet service (**%Service_Telnet**) supports SSL/TLS-protected connections. To establish the use of SSL/TLS for the Telnet service, the steps are:

1. Configuring the Caché Telnet Server for SSL/TLS
2. Configuring Telnet Clients for SSL/TLS

13.4.1 Configuring the Caché Telnet Server for SSL/TLS

To configure the Caché Telnet server to use SSL/TLS, the procedure is:

1. If there is not already a %SuperServer SSL/TLS configuration associated with the Caché server, create one as described in the section "Creating or Editing an SSL/TLS Configuration."
2. From the Management Portal home page, go to the **SSL/TLS Configurations** page (**System Administration** > **Security** > **SSL/TLS Configurations**).
3. On the **SSL/TLS Configurations** page, select **Create New Configuration**, which displays the **New SSL/TLS Configuration** page.
4. On the **New SSL/TLS Configuration** page, create an SSL/TLS configuration with a configuration name of %TELNET/SSL.
5. On the **System-wide Security Parameters** page (**System Administration** > **Security** > **System Security** > **System-wide Security Parameters**), for the **Superserver SSL/TLS Support** field, choose **Enabled.** This specifies that the Telnet server supports (but does not require) SSL/TLS-secured connections.

   Note: Even if you plan to configure the superserver to require SSL/TLS-secured connections, first specify that SSL/TLS is simply enabled.

6. Make sure the Telnet service, **%Service_Telnet**, is enabled. To do this:
   a. On the **Services** page (**System Administration** > **Security** > **Services**), click **%Service_Telnet** to display the **Edit Service** page for the Telnet service.
   b. On the **Edit Service** page, check **Service Enabled** if it is not already checked.
   c. Click **Save** to save the settings and to return to the **Services** page.

13.4.2 Configuring Telnet Clients for SSL/TLS

Caché accepts SSL/TLS connections from third-party Telnet clients. The required or recommended actions for configuring Telnet clients and servers depend on the selected ciphersuites and vary widely.

Important: To use SSL/TLS with the InterSystems Telnet client, contact the InterSystems Worldwide Response Center (WRC).

The following guidelines apply:

- If the Telnet client requires server authentication, then the server must provide a certificate and the client must have access to the server's certificate chain.
- If the Caché Telnet server *requires* client authentication, then the client must provide a certificate and the server must have access to the client's certificate chain.
- If the Caché Telnet server *requests* client authentication, then the client has the option of providing a certificate and a certificate chain to its certificate authority (CA). If the client does not provide a certificate, then authentication succeeds; if it provides a non-valid certificate or certificate chain, then authentication fails.

For information on how certificate and certificate chains are used for authentication, see the section "Establishing the Required Certificate Chain."

13.5 Configuring Java Clients to Use SSL/TLS with Caché

This section describes how to configure a Java client application to use SSL/TLS when it communicates with Caché. This communication occurs through the superserver, so a related required step is setting up the superserver to use SSL/TLS; this is described in the section "Configuring the Caché Superserver to Use SSL/TLS." Java clients can be implemented using either JDBC or object bindings.

The process for configuring a Java client application to use SSL/TLS with Caché is:

1. Determine if the client requires a keystore or a truststore. This depends on several factors: whether or not the Caché server requests or requires client authentication; whether server authentication is required; and the ciphersuites in use. See "Determining the Need for a Keystore and a Truststore" for more information.
2. Create a configuration file with properties in it to provide those features. See "Creating a Client Configuration" for more information.
3. In the code for the client application, optionally specify the name of the client configuration; if you do not specify a name, Java uses the default configuration information. See "Specifying the Use of the Client Configuration" for more information.

13.5.1 Determining the Need for a Keystore and a Truststore

A keystore serves as a repository for the client's private key, public key certificate, and any Certificate Authority (CA) information. This information is needed (1) if the Caché server requires client authentication or (2) if the ciphersuite in use requires a client key pair:

- Whether or not the Caché server requires client authentication is determined by the choice for the **Peer certificate verification level** field on the **Edit SSL/TLS Configuration** page for that Caché instance's "%SuperServer" SSL/TLS configuration. If the field has a value of `Require`, the client must have a certificate; if the field has a value of `Request`, the server checks a certificate if one is available.
- The client and server agree upon a ciphersuite to use. This ciphersuite determines whether or not there is a client certificate, a key pair, or both. The enabled server ciphersuites are determined by the value of the **Enabled ciphersuites** field on the **Edit SSL/TLS Configuration** page for that Caché instance's "%SuperServer" SSL/TLS configuration. The ciphersuites available to the client depend on the version of Java it is using.

If the client has a private key and certificate, these are stored in the client's keystore; the keystore can also hold the client's root CA certificate and any intermediate CA certificates. To authenticate the server, the client may need to have the root CA certificate for the server and any intermediate CA certificates, these can be stored either in the client's truststore or

along with client certificate information in the keystore. For more information on keystores and truststores, see the section "Keystores and Truststores" in the *Java Secure Socket Extension (JSSE) Reference Guide*.

13.5.2 Creating a Client Configuration

The behavior of a Java client depends on the values of properties in its configuration. The configuration gets these values from what is known as a "configuration file," either from the configuration file's default values or from its configuration-specific values. The following sections describe how configuration files work:

- Configuration Files, Configurations, Properties, Values, and Defaults
- Java Client Configuration Properties
- A Sample Configuration File
- Naming the Configuration File

13.5.2.1 Configuration Files, Configurations, Properties, Values, and Defaults

Each configuration file specifies values for the properties that one or more configurations use. The file includes both default values and configuration-specific values, in the form of name-value pairs. Generally, unversioned property names specify default values for properties and versioned property names specify configuration-specific values.

If a configuration file contains only one configuration definition, that single configuration can use unversioned properties. However, it cannot have an associated name property. Without a named configuration, invoke the configuration without specifying a name (as described in "Specifying the Use of the Client Configuration" and "Specifying a Configuration without a Name").

If a configuration file contains multiple configurations, each configuration is defined by the existence of a numbered version of the *name* property of the form *name.n*, where *n* is the number of the configuration. The names of a configuration's other properties use the same version number as the *name* property, so that they have a form of *propertyname.n* where *propertyname* is the name of the property and *n* is the number of the configuration.

The definitions in a configuration file are case-sensitive. Their use of spaces is flexible. The order of property definitions is also flexible.

To specify the default value of a property for use by all configurations, specify an unversioned property name and its value in the following form:

```
propertyName = propertyValue
```

For example, to specify the default value for the keyStoreType property as pkcs12, the form is:

```
keyStoreType = pkcs12
```

To override the default value for a property, specify a versioned property name, such as:

```
keyStoreType.1 = jceks
```

If a configuration file contains multiple configuration definitions, then these versions must use sequential ordering; if client application code refers to a configuration that follows a sequential gap, then an error results. For example, suppose that a configuration file has three versioned name properties: name.1, name.2, and name.4; the configuration associated with the name.4 property will not ever be created and a reference to it will fail with an error.

13.5.2.2 Java Client Configuration Properties

The properties are:

- cipherSuites — A comma-delimited list of ciphersuites, in order of preference for their use. The available ciphersuites depend on the JRE (Java Runtime Environment) on the machine. [Optional]
- debug — Whether or not debugging information is logged to the Java system.err file. This property can have a value of `true` or `false` (`false` is the default). The setting of this property has no effect on exception handling. [Optional]
- keyRecoveryPassword — Password used to access the client's private key; this was created at the same time as the private key pair. [Required if the private key has password protections and application code is not passing in the private key as an input parameter.]
- keyStore — The file for storing the client private key and certificate information. The keystore can also hold the content typically associated with the truststore. [Optional]
- keyStorePassword — Password to gain access to the keystore. [Required if a password was specified when the keystore was created.]
- keyStoreType — The format of the keystore file, if one is specified. [Optional]

  Supported formats are:

  – `jks` — Java KeyStore, the Java proprietary format. [Default]
  – `jceks` — Java Cryptography Extension KeyStore format.
  – `pkcs12` — Public Key Certificate Standard #12 format.

- logFile — The file in which Java records errors. [Optional]
- name — A versioned identifier for the Java client configuration. (Each name property must be versioned. Any unversioned name property is not meaningful and is ignored.)

  If the configuration file specifies only a single configuration and only uses unversioned property names, the name property is not required (as described in "Specifying the Use of the Client Configuration"). For information about specifying multiple configurations with a single configuration file, see the section "Configuration Files, Configurations, Properties, Values, and Defaults"). [Optional]
- protocol — [Required] The SSL or TLS protocol to be used for the connection. The options and the protocols they specify are:

  – `SSL` — Some variant of SSL.
  – `SSLv3` — Version 3 of SSL.
  – `TLS` — Some variant of TLS. [Default]
  – `TLSv1` — Version 1 of TLS (equivalent to version 3.1 of SSL).
  – `TLSv1.1` — Version 1.1 of TLS (equivalent to version 3.2 of SSL).

- trustStore — The file for storing the server's root CA certificate; it can also hold the certificates for any intermediate CAs. (This information can also be placed in the keystore.) [Optional]
- trustStorePassword — Password to gain access to the truststore. [Required if a password was specified when the keystore was created.]
- trustStoreType — The format of the truststore file, if one is specified. [Optional]

  Supported formats are:

  – `jks` — Java KeyStore, the Java proprietary format. [Default]
  – `jceks` — Java Cryptography Extension KeyStore format.
  – `pkcs12` — Public Key Certificate Standard #12 format.

13.5.2.3 A Sample Configuration File

The following is a sample configuration file for use with a Java client:

```
debug = true
protocol = SSLv3
cipherSuites = SSL_RSA_WITH_RC4_128_MD5
keyStoreType = JKS
trustStore = ca.ts
trustStoreType = JKS

name.1 = CacheJavaClient1
keyStore.1 = cjc1.ks
keyStorePassword.1 = cjc1kspw123&XtraChar$
trustStore.1 = cjc1.ts
trustStorePassword.1 = cjc1tspw[+0therNo|sechars]

name.2 = CacheJavaClient2
keyStore.2 = cjc2.ks
keyStoreType.2 = pkcs12

name.3 = CacheJavaClient3
debug.3 = false
cipherSuites.3 = TLS_RSA_WITH_AES_128_CBC_SHA
```

13.5.2.4 Naming the Configuration File

Either save the configuration file with the name SSLConfig.properties or set the value of the Java environment variable *com.intersys.SSL.ConfigFile* to the name of the file. The code checks for the file in the current working directory.

13.5.3 Specifying the Use of the Client Configuration

Once a configuration has been defined, client application code invokes it when connecting to the server. You can do this either with calls for the DriverManager object or the CacheDataSource object.

13.5.3.1 Using the DriverManager Object

With DriverManager, this involves the following steps:

1. Creating a Java Properties object.
2. Setting the value for various properties of that object.
3. Passing that object to Java Connection object for the connection from the client to the Caché server.

To specify information for the connection, the first part of the process is to create a Properties object from a configuration file and set the values of particular properties in it. In its simplest form, the code to do this is:

```
java.util.Properties prop = new java.util.Properties();
prop.put("connection security level", "10");
prop.put("SSL configuration name",configName);
prop.put("key recovery password",keyPassword);
```

where

- The connection security level of 10 specifies that the client is attempting use SSL/TLS to protect the connection.
- *configName* is a variable whose value holds the name of Java client configuration. If the configuration file has only default values and these are being used for a single configuration, do not include this line; see the following section, "Specifying a Configuration without a Name", for details.
- *keyPassword* is the password required to extract the client's private key from the keystore.

Once the Properties object exists and has been populated, the final step is to pass it to the connection from the Caché Java client to the Caché server. This is done in the call to the **DriverManager.getConnection** method. The form of this call is:

```
Connection conn = DriverManager.getConnection(CacheServerAddress, prop);
```

where *CacheServerAddress* is a string that specifies the address of the Caché server and *prop* is the properties object being passed to that string.

If this call succeeds, the SSL/TLS-protected connection has been established. Typically, application code containing calls such as those described in this section includes various checks for success and protection against any errors. For more details about using the Caché Java binding, see *Using Java with Caché*.

13.5.3.2 Using the CacheDataSource Object

With the CacheDataSource object, the procedure is to create the object, call its methods to set the relevant values, and establish the connection. The methods are:

- **setConnectionSecurityLevel** — This method takes a single argument: a connection security level of 10, which specifies that the client is attempting use SSL/TLS to protect the connection.
- **setSSLConfigurationName** — This method takes a single argument: a variable whose value holds the name of Java client configuration. If the configuration file has only default values and these are being used for a single configuration, do not include this line; see the following section, "Specifying a Configuration without a Name", for details.
- **setKeyRecoveryPassword** — This method takes a single argument: the password required to extract the client's private key from the keystore.

In its simplest form, the code to do this is:

```
try{
   CacheDataSource ds = new CacheDataSource();

   ds.setURL("jdbc:Cache://127.0.0.1:1972/Samples");
   ds.setConnectionSecurityLevel(10);
   ds.setSSLConfigurationName(configName);
   ds.setKeyRecoveryPassword(keyPassword);

   Connection dbconnection = ds.getConnection();
}
```

For a complete list of the methods for getting and setting properties, see the "CacheDataSource" section of the "Caché JDBC Compliance" chapter of *Using Caché with JDBC*. The JavaDoc for com.intersys.jdbc.CacheDataSource is under <install-dir>/dev/java/doc/index.html

13.5.3.3 Specifying a Configuration without a Name

If a configuration file contains only one configuration definition, that single configuration can use unversioned properties. However, it cannot have an associated name property.

When working with a DriverManager object, the Properties object uses only the default values from the configuration file. The code for creating this object differs from the typical case in that there is no call to specify a value for the "SSL configuration name" key:

```
java.util.Properties prop = new java.util.Properties();
prop.put("connection security level", "10");
prop.put("key recovery password",keyPassword);
```

When working with a CacheDataSource object, if you want to specify an unnamed configuration, simply do not call **setSSLConfigurationName**.

13.6 Configuring .NET Clients to Use SSL/TLS with Caché

This section describes how to configure a .NET client application to use SSL/TLS when it communicates with Caché. This communication occurs through the superserver, so a related required step is setting up the superserver to use SSL/TLS; the

process of creating or editing a configuration generally is described in the section "Creating or Editing an SSL/TLS Configuration" and that of setting up a superserver to use SSL/TLS is described specifically in the section "Configuring the Caché Superserver to Use SSL/TLS."

The process for establishing a .NET connection that uses SSL/TLS is:

1. Ensure that you have installed any relevant CA certificates for verifying the server certificate. The location for these is the current user's certificate store (Certificates – Current User\Trusted Root Certification Authorities).
2. Establish a connection to a server, based on the format of the connection string as described in the "Creating a Connection" section of the "Connecting to the Caché Database" chapter of *Using the Caché Managed Provider for .NET*. In addition to the name-value pairs for the server, port, and namespace, include the SSL keyword and specify its value as `true`. For example, a connection that uses SSL/TLS protection might have a connection string of the form:

   ```
   CacheConnect.ConnectionString =
       "Server=localhost; Port=1972; Namespace=SAMPLES; SSL=true;"
       + "Password=SYS; User ID=_SYSTEM;";
   ```

   The `true` value of the SSL keyword specifies that SSL/TLS secures the client-server connection (by authenticating the Caché server to the .NET client and, optionally, authenticating the client to the server). Once the secure connection is established, the Caché server uses the User ID and Password keywords to authenticate the identity of the user connecting through the .NET client. (Note that the connection string does not specify anything related to mutual authentication; it merely specifies a server, which in turn may request or require client authentication.)

13.7 Connecting from a Windows Client Using a Settings File

Topics in this section include:

13.7.1 Overview of the Process

If you are on Windows and are using Studio, ODBC, or the Terminal as an SSL/TLS client, you can use a settings file to configure connections and configurations. This mechanism is available even if there is no instance of Caché on the host.

To use a settings file:

1. Get the certificate authority (CA) certificate for the server. Store it on disk and note the location — you will use it later.
2. Create a file containing connection definitions and configuration definitions, as described in the "About the Settings File" section.

3. Name the file SSLDefs.ini and place it in the InterSystems\Cache directory in the directory for 32-bit common program files. Typically, this is the C:\Program Files (x86)\Common Files\InterSystems\Cache\ directory; if you need to locate the directory, check the value of the Windows environment variable *CommonProgramFiles(x86)* on 64-bit Windows or *CommonProgramFiles* on 32-bit Windows.

By creating the file and placing it in this location, it will automatically be used when you connect to a host and a port that match one of the connections listed in the file.

Note: This feature is only for connections that use the cconnect.dll or cconnect64.dll executable (which are for 32-bit and 64-bit machines, respectively). Connections that use other mechanisms (such as for ADO) do not use the settings file.

13.7.2 About the Settings File

A settings file holds specifications for both connections to SSL/TLS servers and the SSL/TLS configurations that those connections use. For each Windows host that is an SSL/TLS client, a single file holds all its connections and configurations. The necessary information to create a file is:

- The Syntax of the Settings File
- Connection Properties
- Configuration Properties

13.7.2.1 The Syntax of the Settings File

The settings file contains one ore more *connection definitions* and one or more *configuration definitions*:

- Each definition begins with an identifier for the connection or configuration. This appears in brackets on its own line, such as:

  ```
  [MyConfiguration]
  ```

  The identifier can include spaces and punctuation, such as:

  ```
  [MyOtherConfiguration, which connects outside of my local network]
  ```

- Each definition ends either with the next bracketed identifier or the end of the file.
- Each definition includes multiple key-value pairs. All of these use the syntax:

  ```
  key=value
  ```

- The group of key-value pairs specify the properties of a connection definition or configuration definition.
- The value in each key-value pair appears unquoted.

13.7.2.2 Connection Definitions

Each settings file contains one or more *connection definitions*, each of which specifies the properties an SSL/TLS connection and matches that connection to an SSL/TLS configuration. The first line of a connection definition is its identifier, which appears in brackets. After the identifier, there are multiple lines specifying information about the SSL/TLS server and the connection to it:

Address

Required. The address of the SSL/TLS server. This can be an IP address, an unqualified host name in the local domain, or a fully-qualified hostname. (Note: The client only uses the specified configuration if the values of both Address and either Port or TelnetPort match the server to which the client application is connecting.)

Port

Required. The port number on which the SSL/TLS server accepts connections. (Note: The client only uses the specified configuration if the values of both Address and either Port or TelnetPort match the server to which the client application is connecting.)

TelnetPort

The port number on the SSL/TLS server that accepts SSL/TLS-protected connections for InterSystems Telnet. If you do not specify this value, connections using InterSystems Telnet do not support SSL/TLS. (Note: The client only uses the specified configuration if the values of both Address and either Port or TelnetPort match the server to which the client application is connecting.)

SSLConfig

Required. The SSL/TLS configuration that the client uses when connecting to the server specified in this definition. Each configuration is defined in its own section.

13.7.2.3 Configuration Definitions

Each settings file contains one or more *configuration definitions*, each of which specifies the properties of an SSL/TLS configuration; for more information on SSL/TLS configurations, see "About Configurations." The first line of a configuration definition is its identifier, which appears in brackets; if the configuration identifier appears as the value of a connection definition's SSLConfig property, the connection uses the configuration to specify its behavior. After the identifier, there are multiple lines specifying the value of each of the configuration's properties:

`Protocols`

Required. The SSL/TLS protocol(s) that the configuration supports. The available protocols are:

- SSLv3 — 2
- TLSv1 — 4
- TLSv1.1 — 8
- TLSv1.2 — 16

where each protocol has a numeric value. To specify support for multiple protocols, use the sum of their values. Hence, to specify the use of TLSv1.1 and TLSv1.2, use:

```
Protocols=24
```

This property is equivalent to the **Protocols** field in the SSL/TLS configuration page in the Management Portal.

`VerifyPeer`

Required. Whether or not the client requires the verification of the server to which it is connecting:

- 0 — Does not require (and does not perform) peer verification; the connection is established under all circumstances.
- 2 — Requires peer verification; the connection is established only if verification succeeds. This is the recommended value; if you choose this value, you must specify a value for the CAFile property.

This property is equivalent to the **Server certificate verification** field in the SSL/TLS configuration page in the Management Portal.

VerifyHost

Whether or not the client checks if the Common Name or subjectAlternativeName fields of the server's certificate match the host name or IP address as specified in the connection definition:

- 0 — Does not check.
- 1 — Checks.

This property does not have an equivalent in the Management Portal. However, it is the same type of check as the SSLCheckServerIdentity property of the %Net.HttpRequest class.

CipherList

Required. The set of ciphersuites that the client supports for encryption and hashing. This property uses the syntax described on the ciphers(1) man page at openssl.org.

InterSystems strongly suggests using a value of `ALL:!aNULL:!eNULL:!EXP:!SSLv2`. For more information about ciphersuites syntax in Caché and the default value, see the "Enabled Ciphersuites Syntax" section.

This property is equivalent to the **Enabled ciphersuites** field in the SSL/TLS configuration page in the Management Portal.

CertFile

The absolute path and name of the file that contains the client's trusted certificate authority (CA) file; if the client does not have a CA, do not specify a value for this property. If specified, this is an X.509 certificate(s) in PEM format and can include a certificate chain. For information on how this value is used, see the section "Establishing the Required Certificate Chain." (Note that certificates from the Windows Certificate Export Wizard must be in base-64 encoded X.509 format, not the default of DER encoded binary X.509.)

This property is equivalent to the **File containing this client's certificate** field in the SSL/TLS configuration page in the Management Portal.

KeyFile

The absolute path and name of the configuration's private key file; if the client does not have a private key, do not specify a value for this property.

This property is equivalent to the **File containing associated private key** field in the SSL/TLS configuration page in the Management Portal.

Password

The password for decrypting the configuration's private key. If you are using a certificate with a password, this property is required; if you are not using a certificate for the client or if the private key does not have a password, do not specify a value for this property. (If the private key is password-protected and you do not provide a value here, Caché cannot decrypt and use the private key.)

This property is equivalent to the **Private key password** field in the SSL/TLS configuration page in the Management Portal.

KeyType

If the configuration has a private key and certificate, the format in which the configuration's private key is stored:

- DSA — 1

- RSA — 2

This property is equivalent to the **Private key type** field in the SSL/TLS configuration page in the Management Portal.

CAfile

Required. The absolute path and name of the file that contains the server's trusted certificate authority (CA) file. This is an X.509 certificate(s) in PEM format. Note that:

- If you have specified a VerifyPeer value of 2, you must provide this value.
- This is the certificate for CA of the server to which you are connecting, *not* the certificate for your CA.

This property is equivalent to the **File containing trusted Certificate Authority certificate(s)** field in the SSL/TLS configuration page in the Management Portal.

13.7.3 A Sample Settings File

The following sample file defines three connections and two configurations:

```
[MyServer1 SSL/TLS to an instance without SSL/TLS-protected InterSystems Telnet]
Address=127.0.0.1
Port=57777
SSLConfig=SSLConfig

[MyServer2 SSL/TLS to an instance with SSL/TLS-protected InterSystems Telnet]
Address=myserver2
Port=57778
TelnetPort=23
SSLConfig=SSLConfig

[MyServer3 SSL/TLS to an instance with SSL/TLS-protected InterSystems Telnet]
Address=myserver3.myexample.com
Port=57779
SSLConfig=SSLNoVerify

[SSLConfig]
Protocols=24
KeyType=2
VerifyPeer=2
CipherList=ALL:!aNULL:!eNULL:!EXP:!SSLv2
Password=
CertFile=c:\InterSystems\certificates\nopwclicert.pem
KeyFile=c:\InterSystems\certificates\nopwclikey.pem
CAfile=c:\InterSystems\certificates\cacert.pem

[SSLNoVerify]
Protocols=16
KeyType=2
VerifyPeer=0
CipherList=ALL:!aNULL:!eNULL:!EXP:!SSLv2
Password=
CertFile=c:\InterSystems\certificates\nopwclicert.pem
KeyFile=c:\InterSystems\certificates\nopwclikey.pem
CAfile=c:\InterSystems\certificates\cacert.pem
```

13.7.4 How It Works

Important: This section describes how InterSystems products use a settings file to establish an SSL/TLS connection. By describing the mechanisms in use, it includes alternate means of creating an SSL/TLS connection. InterSystems recommends that you use the standard approach described above, rather than the alternatives mentioned here.

Caché uses the settings file as follows:

1. When you attempt to establish an SSL/TLS connection, the InterSystems TCP/IP client connection library locates the settings file containing connection definitions and configurations. This file is cconnect.dll on 32-bit machines and cconnect64.dll on 64-bit machines. To do this:

   a. It checks the Windows registry for any SSL/TLS connection definitions.

   b. If there are no connection definitions in the registry, the library attempts to locate any SSL/TLS configurations that are stored in a settings file.

   c. If the *ISC_SSLconfigurations* environment variable exists, the library uses the value of that variable as the full path and file name of the settings file.

      Note: If you need to define the value of the *ISC_SSLconfigurations* environment variable, you may need administrator permissions.

   d. If the *ISC_SSLconfigurations* environment variable does not exist, the library uses the SSLdefs.ini file in the InterSystems\Cache directory under the 32-bit common program files directory identified by the Windows environment variables *CommonProgramFiles(x86)* on 64-bit Windows or *CommonProgramFiles* on 32-bit Windows.

2. Once it has located the settings file, the library locates the relevant connection definition for the connection you are attempting to establish.

   To do this, it searches the sections of the file for one that contains Address and Port properties that match those of the connection you are attempting to establish. When it locates such a section, it uses the value of the *SSLConfig* property there to locate the matching SSL/TLS configuration section.

3. In the specified SSL/TLS configuration section, the library uses the values of the configuration properties to specify the type of connection to initiate with the server.

13.8 Configuring Caché to Use SSL/TLS with Mirroring

This section covers the following topics:

- About Mirroring and SSL/TLS
- Creating and Editing an SSL/TLS Configuration for a Mirror

For general information about Caché support for mirroring, see the "Mirroring" chapter of the *Caché High Availability Guide*.

13.8.1 About Mirroring and SSL/TLS

To provide security within a mirror, you can configure its nodes to use SSL/TLS. This provides for both authentication of one node to another, and for encrypted communication between nodes. As sensitive data passes between the failover members (and to an async member), it is recommended to encrypt the communication to prevent data theft or alteration over the network. Additionally, since a failover member has the ability to request an ISCAgent to take action on another Caché system (such as to request journal file information or force Caché down), it is important to protect such communication between the failover members of a mirror (and their corresponding ISCAgent processes).

Note: If the failover members use database (or journal) encryption, then SSL/TLS is required for communications between them and with any async members. (Specifically, Caché checks if either member has an encryption key activated; if so, the instance requires that the user enable SSL/TLS with mirroring.) For more details on database encryption and journal file encryption, see the chapter "Managed Key Encryption."

To both participate in mirroring (either as a failover member or as an async member) and use SSL/TLS, an instance must have two Caché SSL/TLS configurations – one of type server and the other of type client; each of these must have an X.509 SSL/TLS certificate issued by a trusted Certificate Authority. The certificates should contain a unique identifier in the Common Name (CN) component of the certificate, such as the fully qualified domain name (FQDN) of the instance plus the member's Caché node name; because the CN is a field in a certificate's distinguished name (DN), establishing a unique CN ensures that the certificate's DN uniquely identifies the member. To create an instance's mirroring configurations, follow the procedure in the next section.

When SSL/TLS is enabled, the following actions occur:

1. Server authentication: When the client connects to the server, it requires the server to authenticate itself. This authentication verifies that the DN for the server's certificate matches the DN for a system configured in the client's mirror configuration. If there is no match, the client drops the connection.
2. Client authentication: When the server accepts a connection from a client, it requires the client to authenticate itself. This authentication also verifies that the DN for the client's matches the DN for a system configured in the server's mirror configuration. Again, if there is no match, the server drops the connection.
3. Encryption: The SSL/TLS protocol automatically uses the server's certificate to establish an encrypted channel between the client and the server, so that any data passing through this channel is encrypted and thereby secured.

InterSystems strongly recommends using SSL/TLS with a mirror.

Note on Configuring an Async Member with SSL/TLS

If a mirror uses SSL/TLS, then in addition to enabling SSL/TLS for the mirror and creating the configurations for each member (described in the following section), there are special steps that must be taken when configuring the second failover member or an async member; for more information, see the "Authorize the Second Failover Member or Async (SSL/TLS Only)" section of the "Mirroring" chapter of the *Caché High Availability Guide*. Specifically, for each failover member, on the **Mirror Monitor** page, you need to enter the DN (distinguished name) in the **ID listed as DN in member's X.509 credentials** field; you can copy the value of the DN from **X.509 Distinguished Name** field of the **Join as Async** page (**System Administration** > **Configuration** > **Mirror Settings** > **Join as Async**) for the async member. (Caché populates the **X.509 Distinguished Name** field based on the information in the async member's certificate.)

Note on Disabling SSL/TLS for a Mirror

To disable SSL/TLS for an existing mirror, disable it on the primary member.

Important: Use of SSL/TLS with mirroring is highly recommended. Disabling SSL/TLS for a mirror is strongly discouraged.

13.8.2 Creating and Editing an SSL/TLS Configuration for a Mirror

To use SSL/TLS with a mirror, each member (failover or async) uses a pair of SSL/TLS configurations that are called `%MirrorClient` and `%MirrorServer`; the Portal allows you to create and edit these configurations.

Note: These configurations must already exist on each member when SSL/TLS is enabled for the mirror.

13.8.2.1 Creating an SSL/TLS Configuration for a Mirror Member

To create the configurations, the procedure is:

1. Enable mirroring for that instance of Caché if it is not already enabled. To do this, use the **Edit Service** page for the **`%Service_Mirror`** service; on this page, select the **Service Enabled** check box. You can reach this page either of two paths:
   - On the **Mirror Settings** page (**System Administration** > **Configuration** > **Mirror Settings**), select **Enable Mirror Service**.

- On the **Services** page (**System Administration** > **Security** > **Services**), select **%Service_Mirror**.

2. Go to the **Create SSL/TLS Configurations for Mirror** page. You can do this either on the **SSL/TLS Configurations** page (**System Administration** > **Security** > **SSL/TLS Configurations**) by clicking **Create Configurations for Mirror** or on the **Create Mirror** page (**System Administration** > **Configuration** > **Mirror Settings** > **Create Mirror**) by clicking **Set up SSL/TLS**.
3. On the **Create SSL/TLS Configurations for Mirror** page (**System Administration** > **Security** > **SSL/TLS Configurations** > **Create SSL/TLS Configurations for Mirror**), complete the fields on the form. The fields on this page are analogous to those on the **New SSL/TLS Configuration** page (as described in the section "Creating or Editing an SSL/TLS Configuration"). Since this page creates both server and client configurations that mirroring automatically enables (%MirrorClient and %MirrorServer), there are no **Configuration Name**, **Description**, or **Enabled** fields; also, for the private-key password, this page allows you to enter or replace one (**Enter new password**), specify that none is to be used (**Clear password**), or leave an existing one as it is (**Leave as is**).

   Since both configurations need the same X.509 credentials, completing this form saves both configurations simultaneously. Fields on this page are:

   - **File containing trusted Certificate Authority X.509 certificate(s)**

     Note: This file must include the certificate(s) that can be used to verify the X.509 certificates belonging to other mirror members. If the file includes multiple certificates, they must be in the correct order, as described in Establishing the Required Certificate Chain, with the current instance's certificate first.

   - **File containing this configuration's X.509 certificate**

     Note:
     - The certificate's distinguished name (DN) must appear in the certificate's subject field.
     - If the certificate to be used includes either Key Usage or Extended Key Usage extensions, see the following section, "Special Considerations for Certificates for Mirror Members."

   - **File containing associated private** key
   - **Private key type**
   - **Password**

     If you select **Leave as is**, the page displays two additional fields, for entering and confirming a new password for the private key associated with the certificate.
   - **Protocols**
   - **Enabled ciphersuites**

   Once you complete the form, click **Save**.

For general information about configuring mirror members, see the "Creating a Mirror" section of the "Mirroring" chapter of the *Caché High Availability Guide*.

13.8.2.2 Editing SSL/TLS Configurations for a Mirror Member

If you have already created a member's %MirrorClient and %MirrorServer configurations, you can edit them on the **Edit SSL/TLS Configurations for Mirror** page (**System Administration** > **Security** > **SSL/TLS Configurations**; click **Edit Configurations for Mirror**). This page displays the same fields as the **Create SSL/TLS Configurations for Mirror** page, as described in the previous section.

13.8.2.3 Special Considerations for Certificates for Mirror Members

When using SSL/TLS with mirroring, the %MirrorClient and %MirrorServer configurations must use the same certificate and private key. Hence, the certificate in use by both configurations must be usable as both a server and a client certificate.

There are certain certificate extensions that are specific to SSL/TLS clients or servers. Because the certificate in use with mirroring must be able to serve both uses (as both a client and a server), if any of these extensions appear in a certificate, then the extensions for client and server must both be present. For example, this is true for the Key Usage and Extended Key Usage extensions. If the Key Usage extension is present, then it must specify both of the following:

- The Digital Signature key usage (for clients)
- The Key Encipherment key usage (for servers)

Similarly, if the Extended Key Usage extension is present, then it must specify both:

- The Client Authentication key usage
- The Server Authentication key usage

If both extensions are present, then each must specify both values. Of course, it is also valid to have neither extension present.

If a certificate only specifies one value (either client or server), the SSL/TLS connection for mirroring fails with an error such as:

```
error:14094413:SSL routines:SSL3_READ_BYTES:sslv3 alert unsupported certificate
```

The way to eliminate this error depends on how you obtained your certificates:

- If you are using self-signed certificates, create new certificates (such as with the OpenSSL library) that adhere to these conditions.
- If you are using a commercial certificate authority tool (such as Microsoft Certificate Services), create new certificates that adhere to these conditions and use the tool to sign your certificate signing requests (CSRs).
- If you are purchasing certificates from a commercial certificate authority (such as VeriSign), include a request along with your CSRs that they adhere to these conditions.

13.9 Configuring Caché to Use SSL/TLS with TCP Devices

This section describes how to use SSL/TLS with a Caché TCP connection. The process is:

1. Creating an SSL/TLS configuration that specifies the characteristics you want.
2. Opening a TCP connection or open a socket for accepting such connections.
3. Securing the connection using SSL/TLS. This can occur either as part of opening the connection/socket or afterwards.

How you invoke the Caché SSL/TLS functionality depends on whether you are using Caché as a client or server and whether you are creating an initially-secured TCP connection or adding SSL/TLS to an existing connection.

This section addresses the following topics:

- Configuring a Client to Use SSL/TLS with a TCP Connection
- Configuring a Server to Use SSL/TLS with a TCP Socket

13.9.1 Configuring a Client to Use SSL/TLS with a TCP Connection

To establish a secure connection from a client, the choices are:

- Opening an SSL/TLS-secured TCP Connection from a Client
- Adding SSL/TLS to an Existing TCP Connection

13.9.1.1 Opening an SSL/TLS-secured TCP Connection from a Client

In this scenario, Caché is part of the client and the TCP connection uses SSL/TLS from its inception. The procedure is:

1. Make sure that the configuration you wish to use is available. If it was created before Caché was last started, it is activated and ready for use; otherwise, you can create a new one or edit an existing one.
2. Open a TCP connection using SSL/TLS.

If Caché is a client, then it connects to the server via the client application. The connection uses the specified configuration to determine its SSL-related behavior.

Opening a TCP Connection Using SSL/TLS

This involves opening a named connection that uses SSL/TLS and communicates with a particular machine and port number. The procedure is:

1. Specify the device that you are connecting to:

   ```
   Set MyConn = "|TCP|1000"
   ```

   The TCP string specifies that this is a TCP device. For more information on initiating a TCP connection, see the section "OPEN Command for TCP Devices" in the "TCP Client/Server Communication" chapter of the *Caché I/O Device Guide*.

2. Open the connection, specifying the use of SSL/TLS with either the `/SSL` or `/TLS` parameter.

   ```
   OPEN MyConn:(SvrID:1000:/SSL="MyCfg")
   ```

   where

   - *MyConn* is the device previously specified
   - *SvrID* can be a string that is a resolvable DNS name or an IP address
   - *MyCfg* is a saved (and activated) SSL/TLS configuration

   This call opens a TCP connection to the loopback processor (that is, the local machine) on port 1000 using SSL. It uses SSL/TLS according to the characteristics specified by the MyCfg configuration.

   Optionally, the call can include a password for the private key file:

   ```
   OPEN MyConn:(SvrID:1000:/SSL="MyCfg|MyPrivateKeyFilePassword")
   ```

   Here, all the arguments are as above and *MyPrivateKeyFilePassword* is the actual password.

   Important: The ability to include a password when opening a TCP connection using SSL/TLS is for real-time interactive use only. You should *never* store a private key password persistently without protecting it. If you need to store such a password, use the PrivateKeyPassword property of the Security.SSLConfigs class.

   For more information on opening a TCP device, see "OPEN and USE Command Keywords for TCP Devices" in the "TCP Client/Server Communication" chapter of the *Caché I/O Device Guide*.

Once the connection is established, you can then use it in the same manner as any other TCP connection.

13.9.1.2 Adding SSL/TLS to an Existing TCP Connection

This scenario assumes that the TCP connection has already been established. The procedure is:

1. Make sure that the configuration you wish to use is available. If it was created before Caché was last started, it is activated and ready for use; otherwise, you can create a new one or edit an existing one.
2. Secure the existing TCP connection using SSL/TLS.

Securing an Existing TCP Connection Using SSL/TLS

This involves adding SSL/TLS to an already-existing connection to a particular machine and port number. The procedure is:

1. Determine the name of the device to which there is a connection. For example, this might have been established using the following code:

   ```
   SET MyConn="|TCP|1000"
   OPEN MyConn:("localhost":1000)
   ```

   The TCP string specifies that this is a TCP device. For more information on initiating a TCP connection, see the section "OPEN Command for TCP Devices" in the "TCP Client/Server Communication" chapter of the *Caché I/O Device Guide*.

2. Specify the use of SSL/TLS as follows with either the `/SSL` or `/TLS` parameter:

   ```
   USE MyConn:(::/TLS="MyCfg")
   ```

   where

   - *MyConn* is the device previously specified
   - *MyCfg* is an SSL/TLS configuration

   Optionally, the call can include a password for the private key file:

   ```
   USE MyConn:(::/TLS="MyCfg|MyPrivateKeyFilePassword")
   ```

   Here, all the arguments are as above and *MyPrivateKeyFilePassword* is the actual password.

   Important: The ability to include a password when securing an existing TCP connection using SSL/TLS is for real-time interactive use only. You should *never* store a private key password persistently without protecting it. If you need to store such a password, use the PrivateKeyPassword property of the Security.SSLConfigs class.

   For more information on opening a TCP device, see "OPEN and USE Command Keywords for TCP Devices" in the "TCP Client/Server Communication" chapter of the *Caché I/O Device Guide*.

Having added SSL/TLS security to the connection, you can continue to use it in the same manner as before.

13.9.2 Configuring a Server to Use SSL/TLS with a TCP Socket

To enable a socket to require a secure connection from a client, you can either:

- Open a TCP socket specifying that this connection requires SSL or TLS.
- Establish the requirement for the use of SSL or TLS on an already-existing socket.

13.9.2.1 Establishing an SSL/TLS-secured Socket

In this scenario, Caché is the server and the TCP socket uses SSL/TLS from its inception. The procedure is:

1. Make sure that the configuration you wish to use is available. If it was created before Caché was last started, it is activated and ready for use; otherwise, you can create a new one or edit an existing one.
2. Open a TCP socket that requires the use of SSL/TLS.

This socket requires the use of SSL/TLS from clients connecting to it. When a client attempts to connect to the server, the server attempts to negotiate a connection that uses SSL/TLS. If this succeeds, the connection is available for normal use and communications are secured using the negotiated algorithm. If it fails, there is no connection available for the client.

Opening a TCP Socket Requiring SSL/TLS

To open a socket that requires SSL/TLS, the procedure is:

1. Specify the device that is accepting connections:

   ```
   SET MySocket = "|TCP|1000"
   ```

   The TCP string specifies that this is a TCP device. For more information on initiating a TCP connection, see the section "OPEN Command for TCP Devices" in the "TCP Client/Server Communication" chapter of the *Caché I/O Device Guide*.

2. Open the connection, specifying the use of SSL/TLS with either the /SSL or /TLS parameter.

   ```
   OPEN MySocket:(:1000:/TLS="MyCfg")
   ```

   Optionally, the call can include a password for the private key file:

   ```
   OPEN MySocket:(:1000:/TLS="MyCfg|MyPrivateKeyFilePassword")
   ```

   This call opens a TCP socket on port 1000 using TLS. For more information on opening a TCP device, see "OPEN and USE Command Keywords for TCP Devices" in the "TCP Client/Server Communication" chapter of the *Caché I/O Device Guide*.

   Important: The ability to include a password when opening a TCP connection using SSL/TLS is for real-time interactive use only. You should *never* store a private key password persistently without protecting it. If you need to store such a password, use the PrivateKeyPassword property of the Security.SSLConfigs class.

13.9.2.2 Adding SSL/TLS to an Existing Socket

This scenario assumes that a connection to the TCP socket has already been established. The procedure is:

1. Make sure that the configuration you wish to use is available. If it was created before Caché was last started, it is activated and ready for use; otherwise, you can create a new one or edit an existing one.
2. Use SSL/TLS to secure the existing TCP connection to the socket.

Securing an Existing TCP Connection to the Socket Using SSL/TLS

This involves adding SSL/TLS to an already-existing connection to a socket on a particular machine and port number. The procedure is:

1. Determine the name of the device on which the socket is open. For example, this might have been established using the following code:

   ```
   SET MySocket = "|TCP|1000"
   OPEN MySocket:(:1000)
   ```

The TCP string specifies that this is a TCP device. For more information on initiating a TCP connection, see the section "OPEN Command for TCP Devices" in the "TCP Client/Server Communication" chapter of the *Caché I/O Device Guide*.

2. Specify the use of SSL/TLS as follows with either the `/SSL` or `/TLS` parameter:

```
USE MySocket:(::/SSL="MyCfg")
```

where

- *MySocket* is the device previously specified
- *MyCfg* is an SSL/TLS configuration

Optionally, the call can include a password for the private key file:

```
USE MySocket:(::/SSL="MyCfg|MyPrivateKeyFilePassword")
```

For more information on opening a TCP device, see "OPEN and USE Command Keywords for TCP Devices" in the "TCP Client/Server Communication" chapter of the *Caché I/O Device Guide*.

Important: The ability to include a password when securing an existing TCP connection using SSL/TLS is for real-time interactive use only. You should *never* store a private key password persistently without protecting it. If you need to store such a password, use the PrivateKeyPassword property of the Security.SSLConfigs class.

Having added SSL/TLS security to the socket, you can continue the connection to it in the same manner as before.

13.10 Configuring the CSP Gateway to Connect to Caché Using SSL/TLS

You can use SSL/TLS to set up a secure, encrypted channel between the CSP Gateway and the Caché server. To do this, you need an SSL/TLS certificate and private key that represents the Gateway. The Gateway can then establish an encrypted connection to the Caché server (which has its own certificate and private key), so that all information is transmitted through the connection.

Note: For information on setting up a connection between the CSP Gateway and the Caché server that is protected by Kerberos, see the "Setting Up a Kerberized Connection from the CSP Gateway to Caché" section of the "Authentication" chapter.

The procedure is:

1. If there is not already a `%SuperServer` SSL/TLS configuration associated with the Caché server, create one as described in the section "Creating or Editing an SSL/TLS Configuration."
2. On the Portal's **System-wide Security Parameters** page (**System Administration** > **Security** > **System Security** > **System-wide Security Parameters**), for the **Superserver SSL/TLS Support** choice, select **Enabled** (*not* **Required**).
3. Go to the CSP Gateway's **Server Access** page (**System Administration** > **Configuration** > **CSP Gateway Management**).
4. On that page, under **Configuration**, select **Server Access**.
5. Next, select **Edit Server** and click **Submit**. This displays the configuration page for the CSP Gateway.

6. On this page, configure the CSP Gateway to use SSL/TLS. Specifically, for the **Connection Security Level** field, select **SSL/TLS**.

   You must also specify values for the **SSL/TLS Protocol**, **SSL/TLS Key Type**, **Require peer certificate verification**, **SSL/TLS Certificate File**, **SSL/TLS Private Key File**, **SSL/TLS CA Certificate File**, and **SSL/TLS Private Key Password** fields. For more details on the fields on this page, see the "Configuring Server Access" section of the "CSP Gateway Operation and Configuration" chapter of the *CSP Gateway Configuration Guide*.

13.11 Establishing the Required Certificate Chain

For a connection to be successfully established using a ciphersuite that uses certificates and keys, the client must be able to verify the server's certificate chain from the server's own certificate to a self-signed certificate from a trusted certificate authority (CA), including intermediate certificates (if any). If the server is authenticating the client user, then the server must also be able to verify the client user's certificate chain from the client user's own certificate to a trusted CA's self-signed certificate, including intermediate certificates (if any).

Since authentication can be bidirectional, the requirements for certificate chains refer to the verifying entity (the side requiring the authentication) and the verified entity (the side being authenticated), rather than the client and the server.

For authentication to be possible, the following conditions must be met:

- The verifying entity must have access to all the certificates that constitute the certificate chain from the verified entity's own certificate to a trusted CA's self-signed root certificate. The certificates in the chain are obtained from the combination of the verified entity's certificate file (the certificates are sent as part of the handshake protocol) and the verifying entity's trusted CA certificate file.
- The verifying entity must have the trusted CA's self-signed root certificate in its CA certificate file.
- The verified entity's own certificate must be the first entry in its certificate file.
- All intermediate CA certificates must be present.
- The certificates in the certificate chain may be divided between the verified entity's certificate file and the verifying entity's trusted CA certificate file. However, each part must be a contiguous partial certificate chain, as described in the following example.

Suppose there are:

- A verified entity (named "VE") with a certificate signed by the certificate authority named "ICA1."
- A certificate for "ICA1" signed by the certificate authority "ICA2," and a certificate for "ICA2" signed by "RootCA".
- A trusted CA (named "RootCA") with a self-signed root certificate.

The following are valid distributions of certificates between the verified entity and the verifying entity:

Table 13–1: Valid Certificate Distribution Schemes

Certificates in the Verified Entity's Certificate File	Certificates in the Verifying Entity's Trusted CA Certificate File
VE	ICA1, ICA2, RootCA
VE, ICA1	ICA2, RootCA
VE, ICA1, ICA2	RootCA

Note that it is *not* valid to have VE and ICA2 in the verified entity's certificate file and ICA1 and RootCert in the verifying entity's trusted CA certificate file

14

The InterSystems Public Key Infrastructure

This chapter covers the following topics:

14.1 About the InterSystems Public Key Infrastructure (PKI)

A Public Key Infrastructure (PKI) provides a means of creating and managing private keys, public keys, and certificates. These are used for cryptographic operations including encryption, decryption, and digital signing and signature verification. Certificates provide a means of associating a public key with an identity. To do this, a PKI includes a trusted third party known as a certificate authority (CA).

The InterSystems PKI implementation gives Caché the ability to serve as a Certificate Authority (CA). An instance of Caché acting as a CA is known as a CA server; an instance of Caché using a CA's services is known as a CA client. A single instance of Caché can be both a CA server and a CA client. As a CA server, an instance can either generate and use a self-signed CA Certificate, or it can use a CA Certificate issued by a commercial third party or product. As a CA client, an instance is associated with a CA server; the CA client's certificate is available for use with SSL/TLS, XML encryption, and signature verification; there is also the option of configuring a CA client to serve as an intermediate CA. Communications involving the PKI occur through web services.

When establishing itself as a CA server, an instance of Caché either creates a public/private key pair and then embeds the public key in a self-signed X.509 certificate or it uses a private key and X.509 certificate signed by an outside CA. X.509 is an industry-standard certificate structure that associates a public key with both identifying information for an entity and other related data; this identifying information is known as a subject Distinguished Name (DN), and consists of various specific information regarding an entity's organization, location, or both. You can use X.509 certificates to provide a high level of public-key–based security if, and only if, appropriate security policies regarding the protection of private keys and the issuance of certificates are enforced, including strict control of the CA server's private key file, preferably stored on removable media which can be physically secured when not in use.

To use the Caché PKI infrastructure from the Management Portal, all actions start from the **Public Key Infrastructure** page (**System Administration** > **Security** > **Public Key Infrastructure**).

Important: The InterSystems PKI is for testing purposes only. Do *not* use it in a production setting.

For more background on PKI and CAs, see the appendix, "About Public Key Infrastructure (PKI)." For technical details about the SSL/TLS calls underlying the InterSystems PKI, see the OpenSSL library. For technical details about X.509 certificates, see RFC 5280, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile."

14.1.1 Help for Management Portal PKI Tasks

The following are links to help for PKI tasks:

- Certificate Authority Client
  - Submit Certificate Signing Request to Certificate Authority server
  - Get Certificate(s) from Certificate Authority server
  - Configure local Certificate Authority client
- Certificate Authority Server
  - Process pending Certificate Signing Requests
  - Configure local Certificate Authority server

14.2 Certificate Authority Server Tasks

A Caché instance can serve as a certificate authority (CA) server. This involves:

1. Configuring a Caché instance as a CA server. This involves providing information that is either related to the certificate or that the CA server uses for processing certificate signing requests.
2. Managing pending certificate signing requests (CSRs). This is the ongoing work of a CA server.

Note: Because these tasks are for the CA server administrator, this section is addressed to those administrators. This differs from the tasks in the CA client tasks, which are addressed to CA client administrators/technical contacts.

14.2.1 Configuring a Caché Instance as a Certificate Authority Server

Before any PKI operations are possible, you need to configure a Caché instance as a Certificate Authority (CA) server. This involves either:

- Configuring a CA Server with a New Private Key and Certificate
- Configuring a CA Server with an Existing Private Key and Certificate

It may also involve reinitializing a CA server.

14.2.1.1 Configuring a CA Server with a New Private Key and Certificate

If you are creating a new private key and certificate, the procedure is:

1. For the selected Caché instance, in the Management Portal, go to the **Public Key Infrastructure** page (**System Administration** > **Security** > **Public Key Infrastructure**).
2. On the **Public Key Infrastructure** page, under **Certificate Authority Server**, select **Configure Local Certificate Authority server**. This displays fields for (1) the file name root for the CA server's certificate and private key and (2) the directory that holds these files.

   Important: If you specify a path and file name root that point to an existing certificate and private key, Caché uses these for the CA server. Otherwise, it creates a new certificate and private key. (Also, if only one of the files exists, Caché renames it by appending the `.old` suffix to it and creates new files.)

   The fields are:

   - **File name root for Certificate Authority's Certificate and Private Key files (without extension)** — Required. Specifies the part of the name of the private key and certificate files that is common to each. This can be for an existing pair of files or for a new pair of files. Hence, if you select `MyCA` as the file name root, the private key is stored in the MyCA.key file and the certificate is stored in the MyCA.cer file. Valid characters for this field are alphanumeric characters, the hyphen, and the underscore. The root cannot be the string "cache".
   - **Directory for Certificate Authority's Certificate and Private Key files** — Required. The path to a directory for storing the CA's certificate and private key files; if the directory does not exist, Caché attempts to create it. This directory should always be on an external device (not a local hard drive or a network server), preferably on an encrypted external device. As this is the directory that holds the CA's private key, it is extremely important that this be a completely secure location. If you provide a relative path here, the path is relative to <install-dir>/mgr/ for the Caché instance.

3. Click **Next** to continue.
4. The fields that appear next depend on whether you are creating a new private key and certificate pair or using an existing private key and certificate. When you are creating a new private key and certificate, Caché displays the following fields:

   - **Password to Certificate Authority's Private Key file** and **Confirm Password** — Required. The password to encrypt and decrypt the CA's private key file.
   - **Certificate Authority Subject Distinguished Name** — The set of one or more name-value pairs that define the distinguished name (DN) that describes the bearer of the CA certificate. You must provide a value for at least one attribute. The attributes are:
     - **Country** — A two-letter code identifying the country, using the ISO country codes.
     - **State or Province** — The name of the CA's state or province. The convention is not to use this field for CA certificates.
     - **Locality** — The name of the CA's municipality. The convention is not to use this field for CA certificates.
     - **Organization** — Name of the organization that is administering the CA. By convention, this value is spelled out in full, such as "InterSystems Corporation," rather than simply "InterSystems" or "InterSystems Corp."

  - **Organizational Unit** — Any other organizational information or special commentary on the CA. Examples of this can include the CA's department, a statement that the CA is for use only within an enterprise, and so on.
  - **Common Name** — A descriptive name for the CA, such as "Documentation Test CA."

- **Validity period for Certificate Authority's Certificate (days)** — Required. The validity period (lifespan) for the CA certificate itself.
- **Validity period for Certificates issued by Certificate Authority (days)** — Required. The validity period (lifespan) for certificates that the CA issues for its clients.
- **Configure email** — Information required for the email account for managing the CA and its tasks.
  - **SMTP server** — The Simple Mail Transfer Protocol (SMTP) server that handles the CA mail in the form of a fully qualified host name, such as "MyMachine.MyDomain.com".
  - **SMTP username** — A username that can be authenticated by the specified SMTP server. This field does not require a fully qualified username.
  - **SMTP password** — The password associated with the SMTP username.
  - **Confirm password** — A confirmation of the password associated with the SMTP username.
  - **Certificate Authority server administrator's email address** — The user who receives certificate signing requests for the CA. This field requires a fully qualified username, such as "CAMgr@MyDomain.com".

5. Complete these fields as required and click **Save**. Caché displays a message indicating success, such as:

```
Certificate Authority server successfully configured.
Created new files: C:\pki\FileNameRoot.cer .key, and .srl.
Certificate Authority Certificate SHA-1 fingerprint:
E3:FB:30:09:53:90:9A:31:30:D3:F0:07:8F:64:65:CD:11:0A:1A:A2
Confirmation email sent to: CAserver-admin@intersystems.com
```

   This indicates that a private key, certificate, and their associated SRL (serial) file have been created. (Otherwise, Caché displays an error message.)

   Once the files have been created, it is *strongly* recommended that you store the private key on removable media that can be physically secured.

   WARNING! It is critical that you properly protect all private keys, and most important that you protect the private key of a CA. The exposure of a private key can result in security breaches, data exposure, financial losses, and legal vulnerability.

   If it has succeeded, Caché has performed the following actions:

   - Creating a key pair.
   - Saving the private key to a file to a specified location with a specified file name root (see below).
   - Creating a self-signed CA certificate containing the public key.
   - Saving the certificate to a file to a specified location with a specified file name root (see below).
   - Creating a counter of the number of certificates issued and storing it in an SRL (serial) file in the same directory as the certificate and the private key. (Each time the CA issues a new certificate, Caché gives the certificate a unique serial number based on this counter and then increments the value in the SRL file.)

Once you have created the CA private key and certificate, you can process certificate signing requests (CSRs). When a CA client creates a CSR, you, as CA administrator, will receive email notification about this.

14.2.1.2 Configuring a CA Server with an Existing Private Key and Certificate

If you are using an existing private key and certificate (such as from another Caché CA, or from an external CA, such as a commercial CA), the procedure is:

1. Create or obtain a private key and certificate. The certificate must be in PEM format, or you must be able to convert it to PEM format.
2. If they do not already have identical file name roots, rename them as *filenameroot*`.cer` for the certificate and *filenameroot*`.key` for the private key, where *filenameroot* is the file name root you wish to use.
3. Store both files in the same directory, making sure that this directory is accessible to the Caché instance that you are configuring as a CA server. This directory should always be on an external device (not a local hard drive or a network server), preferably on an encrypted external device. As this is the directory that holds the CA's private key, it is extremely important that this be a completely secure location.

   WARNING! It is critical that you properly protect all private keys, and most important that you protect the private key of a CA. The exposure of a private key can result in security breaches, data exposure, financial losses, and legal vulnerability.

4. For the selected Caché instance, in the Management Portal, go to the **Public Key Infrastructure** page (**System Administration** > **Security** > **Public Key Infrastructure**).
5. On the **Public Key Infrastructure** page, under **Certificate Authority Server**, select **Configure Local Certificate Authority server**. Complete the fields on this page as follows:
   - **File name root for Certificate Authority's Certificate and Private Key files (without extension)** — Required. The part of the name of the private key and certificate files that is common to each. For this value, use the file name root that the files have or that you selected in step 2 of this procedure.
   - **Directory for Certificate Authority's Certificate and Private Key files** — Required. The path to a directory that holds the CA's certificate and private key files. For this value, use the directory that you selected in step 3. If you provide a relative path here, the path is relative to <install-dir>/mgr/ for the Caché instance.
6. Click **Next** to continue.
7. The fields that appear next depend on whether you are creating a new private key and certificate pair or using an existing private key and certificate. When you are using an existing private key and certificate, Caché displays the following fields:
   - **Validity period for Certificates issued by Certificate Authority (days)** — Required. The validity period (lifespan) for certificates that the CA issues for its clients.
   - **Configure email** — Information required for the email account for managing the CA and its tasks.
     - **SMTP server** — The Simple Mail Transfer Protocol (SMTP) server that handles the CA mail in the form of a fully qualified host name, such as "MyMachine.MyDomain.com".
     - **SMTP username** — A username that can be authenticated by the specified SMTP server. This field does not require a fully qualified username.
     - **SMTP password** — The password associated with the SMTP username.
     - **Confirm password** — A confirmation of the password associated with the SMTP username.
     - **Certificate Authority server administrator's email address** — The user who receives certificate signing requests for the CA. This field requires a fully qualified username, such as "CAMgr@MyDomain.com".

Important: If the Management Portal displays more fields than these, then you have not properly directed it to the private key and certificate that you wish to use. If you complete all the displayed fields, click **Save**, and there is success, Caché will have created a new private key and certificate for the CA server.

8. Click **Save**. When you save the configuration information for the local CA server, Caché uses the existing certificate and private key. (It will also create an SRL file, if one does not exist.) It will display a success message such as:

```
Certificate Authority server successfully configured.
Using existing files: C:\pki\FileNameRoot.cer and .key
Certificate Authority Certificate SHA-1 fingerprint:
E3:FB:30:09:53:90:9A:31:30:D3:F0:07:8F:64:65:CD:11:0A:1A:A2
Confirmation email sent to: CAserver-admin@intersystems.com
```

As with creating a new private key and certificate, at this point, the CA server is configured and is now ready to process certificate signing requests (CSRs). Again, when a CA client creates a CSR, you, as CA administrator, will receive email notification about this.

14.2.1.3 Reinitializing a CA Server

If you have already configured an instance as a CA server, then there is a **Reinitialize** button on the page for configuring a CA. Selecting it has the following effects:

- It deletes all configuration information for the CA server.
- It discards all information for issued certificates.
- It discards all certificate signing requests pending with the CA.

Note: Reinitialization does not delete the files containing the private key or existing certificate for the CA, nor does it delete the CA's existing SRL file; in fact, these are still valid and can be used. Also, it does not render any already signed certificates invalid.

When you click the button, there is a prompt to confirm that you want to reinitialize the CA. After reinitialization, you can configure a new CA server.

14.2.2 Managing Pending Certificate Signing Requests

Once the Certificate Authority (CA) server has been configured, the principal task associated with the CA server is managing certificate signing requests (CSRs) from potential CA clients. This can involve:

- Processing a Certificate Signing Request (CSR)
- Deleting a Certificate Signing Request (CSR)

If processing leads to approving the request, the CA server issues an X.509 certificate signed with the CA's private key, and sends email notification of the issued certificate's serial number to the CA client's technical contact. It is also possible to delete (that is, reject) a request.

A critical step in this process is *verification*, in which the CA administrator uses communications that prevent impersonation to verify the identity of the requester, the authority of the technical contact to hold a certificate with the requested DN, and the purpose for which the certificate is being issued. (To do this, the CA server's administrator uses the contact information received from the potential CA client along with the CSR.)

14.2.2.1 Processing a Certificate Signing Request (CSR)

To process a request is to convert the CSR into a certificate. The procedure is:

1. In the Management Portal, go to the **Public Key Infrastructure** page (**System Administration** > **Security** > **Public Key Infrastructure**).
2. On the **Public Key Infrastructure** page, under **Certificate Authority Server**, select **Process pending Certificate Signing Requests**. This displays a table of CSRs that the CA has received and not processed or deleted; to the right of each CSR there are **Process** and **Delete** links.
3. Mount the media containing the CA server certificate and private key files. (This is the media on which you stored these files while configuring a Caché instance as a CA server.)
4. To process a CSR, click **Process**. This displays the contents of the CSR.
5. Prior to issuing the certificate, you need to specify the use of the certificate. The choices for the **Certificate Usage** radio buttons specify which operations the certificate can perform:
   - **TLS/SSL, XML encryption and signature verification** — For CA clients that are directly using various security capabilities within Caché.
   - **Intermediate Certificate Authority server** — For CA clients that will themselves be serving as CAs for other instances of Caché.
6. **Important:** This step requires that you verify the identity of the technical contact for the potential CA client using a means that prevents impersonation.

   As the instructions on this page specify, you must contact the designated technical contact for the instance that has submitted the CSR. According to the policies of your organization, contact this person by phone or in person and verify:
   - This person's identity
   - This person's authority to hold a certificate containing the Subject Distinguished Name shown above, signed by the CA for which you are responsible
   - That the SHA-1 fingerprint shown above matches the one reported to them when they submitted the certificate signing request
7. Once you have specified the purpose of the certificate and verified the relevant information with the technical contact, you can issue the certificate. To do this, click **Issue Certificate**. This causes the page to display the **Password for Certificate Authority's Private Key file** field.
8. In the **Password for Certificate Authority's Private Key file** field, enter the password to decrypt the CA server's private key file. If you created the private key and certificate with Caché, this is the value you entered in the **Password to Certificate Authority's Private Key file** field; if you created the private key and certificate using other tools, it is the password, if any, that you provided to those tools for this purpose.
9. Click **Finish** to create the certificate. If successful, Caché displays a message such as

   ```
   Certificate number 31 issued for Certificate Signing Request
   "Santiago Development Group"
   ```
10. Remove the media holding the CA server's certificate and private key, and store it in a secure location.

Caché has now created the certificate and notified the technical contact for the CA client by email that the certificate is available for download. The CA client's technical contact can now download the certificate to the client host.

14.2.2.2 Deleting a Certificate Signing Request (CSR)

To delete a request, the procedure is:

1. In the Management Portal, go to the **Public Key Infrastructure** page (**System Administration** > **Security** > **Public Key Infrastructure**).

2. On the **Public Key Infrastructure** page, under **Certificate Authority Server**, select **Process pending Certificate Signing Requests**. This displays a table of CSRs that the CA has received and not processed or deleted; to the right of each CSR there are **Process** and **Delete** links.
3. To delete a CSR, click **Delete**. This displays a confirmation dialog.
4. In the confirmation dialog, click **OK**.
5. Complete these fields as required and click **Save**.

This deletes the CSR.

14.3 Certificate Authority Client Tasks

A certificate authority (CA) client has one-time setup tasks, which are:

1. Configuring the Caché instance as a CA client. This involves providing location about the CA server to the potential CA client; it also includes providing contact information about the CA client's technical contact.
2. Getting a copy of the CA certificate. This allows for verifying other certificates.

After setup tasks, the CA client tasks are:

1. Submitting a certificate signing request (CSR) to the CA server. From the user's perspective, this involves specifying a distinguished name (DN) and other information. (This may happen repeatedly, if the instance has reason to have multiple distinct certificates.)
2. Getting copies of various certificates. In addition to the CA client's own certificate, this includes any other certificates that the CA server has issued.

After performing these tasks, the CA client can then perform the operations that require use of the PKI. These are tasks in which it is known as an *end entity*, since it is at the end of a secured connection.

Note: Because these tasks are for the CA client administrators/technical contacts, this section is addressed to those individuals. This differs from the tasks in the Certificate Authority Server Tasks, which are addressed to CA server administrators.

14.3.1 Configuring a Caché Instance as a Certificate Authority Client

The procedure to configure a Caché instance as a certificate authority (CA) client involves providing location about the CA server to the potential CA client; it also includes providing contact information about the CA client's technical contact. The steps are:

1. In the Management Portal, go to the **Public Key Infrastructure** page (**System Administration** > **Security** > **Public Key Infrastructure**).
2. On the **Public Key Infrastructure** page, under **Certificate Authority Client**, select **Configure Local Certificate Authority Client**. The fields on this page are:
   - **Certificate Authority server hostname** — Required. The fully qualified name of the machine of the CA server. (Specifically, this is a machine on which an instance of Caché is running, and this instance is serving as a CA server. It must be configured as a CA server prior to configuring any instance as a CA client.)
   - **Certificate Authority WebServer port number** — Required. The webserver port number for the instance of Caché serving as the CA server.

- **Certificate Authority server path** — Required. The path of the web service of the CA server. By default, this is /isc/pki/PKI.CAServer.cls. (This value is used along with the server hostname and port number to contact the web service for the CA.) Do not change the value of this field.
- **Local technical contact** — The person who provides verification information to the CA server on behalf of the CA client. For this person, the following information is required:
  - **Name** — Required. The name of the technical contact for the CA client.
  - **Phone number** — The contact's phone number. This is so that the CA administrator can contact the CA client's technical contact to perform verification prior it issuing the CA client's certificate. The phone number is not required, since Caché does not require a particular mechanism of verification; for example, it could happen in person.
  - **Email address** — The contact's email address. This is so that the CA client's technical contact can receive email notification that the CA server has processed the client's CSR and issued a certificate. The email address is not required, since the server administrator can use some other means to contact the client's technical contact about the newly issued certificate.

3. Complete these fields as required and click **Save**.

Caché acknowledges success through a message such as "Certificate Authority client successfully configured." At this point, the next task is to download the CA server's certificate.

14.3.2 Submitting a Certificate Signing Request to a Certificate Authority Server

Once an instance of Caché is configured as a certificate authority (CA) client, you can then submit a certificate signing request (CSR) to the CA server. On the surface, this involves specifying a distinguished name (DN) and other information. Under the covers, the CA client performs several actions:

1. Generating a public/private key pair.
2. Creating a Certificate Signing Request (CSR) containing the public key and a specified DN.
3. Submitting that to the CA server using a web service.

The PKI infrastructure automatically provides the CSR to the CA server, acknowledges the submission, and sends email notification to the CA server's administrator. The submission includes your contact information as the local technical contact for the CA client. The CA administrator then processes the CSR by using communications that prevent impersonation to verify the identity of the requester, the authority of the technical contact to hold a certificate with the requested DN, and the purpose for which the certificate is being issued. If the request is approved, the completion of the process includes the CA server creating a certificate.

To submit a CSR to a CA server, the procedure is:

1. In the Management Portal, go to the **Public Key Infrastructure** page (**System Administration** > **Security** > **Public Key Infrastructure**).
2. On the **Public Key Infrastructure** page, under **Certificate Authority Client**, select **Submit Certificate Signing Request to Certificate Authority Server**. The fields on this page are:
   - **File name root for local Certificate and Private Key files (without extension)** — Required. Specifies the part of the name of the private key and certificate files that is common to each. Hence, if you select `CAClient` as the file name root, the private key is stored in the CAClient.key file and the certificate is stored in the CAClient.cer file. Valid characters for this field are alphanumeric characters, the hyphen, and the underscore. The root cannot be the string "cache".

- **Password to Certificate Authority's Private Key file** and **Confirm Password** — Optional. The password that for encrypting and decrypting the CA client's private key.
- **Subject Distinguished Name** — The set of one or more name-value pairs that define the distinguished name (DN) that describes the bearer of the client certificate. You must provide a value for at least one attribute. The attributes are:
  - **Country** — A two-letter country code for the country, using the ISO country codes.
  - **State or Province** — The name of the state or province, spelled out in full.
  - **Locality** — The name of the municipality, spelled out in full.
  - **Organization** — Name of the organization with which the certificate is associated. By convention, this value is , spelled out in full, such as "InterSystems Corporation," rather than simply "InterSystems" or "InterSystems Corp."
  - **Organizational Unit** — Any other organizational information, such as a department.
  - **Common Name** — A descriptive name for the client, such as "Documentation Test Client."

3. Complete these fields as required and click **Save**. If successful, Caché then displays a message such as:

```
SHA-1 Fingerprint:
0C:DA:5F:06:04:C7:AE:64:61:9C:5C:29:35:49:88:0D:B6:E5:7D:98,
Certificate Signing Request "CAClient060412"
successfully submitted to the Certificate Authority at instance MyCA
on node CATESTCLIENT.CATESTDOMAIN.COM
```

   If Caché has successfully created a CSR, it has performed the following actions:

   - Creating a key pair.
   - Saving the private key to a file in the manager's directory with the specified file name root.
   - Creating a CSR that includes the public key and saving it to a file in the manager's directory with the specified file name root.
   - Submitting that CSR to the CA using the host name, port, and path specified as part of the CA client configuration process.

   (If the process does not succeed, Caché displays an error message.)

   Once the files have been created, it is strongly recommended that you store this sensitive information on encrypted, removable media that can be physically secured.

4. Make a copy of the SHA-1 fingerprint that Caché displays.

   Important: Do not lose this information, as you will need to provide this later, as part of the verification process.

5. At this point, you have used Caché to create and submit the CSR. When the administrator of the CA contacts you, provide the SHA-1 fingerprint that you copied in the last step. The administrator will then create certificate for you, which you can obtain as described in "Get Certificate(s) from Certificate Authority Server."

14.3.3 Getting Certificate(s) from Certificate Authority Server

Once a certificate authority (CA) client has been configured, it can then download any certificate associated with the CA server. This includes:

- The CA server certificate.

- Its own certificate. This is available if the CA client has submitted a certificate signing request (CSR) to the CA server and the CA server has approved the request.
- Any other certificates that the CA server has created for any other CA clients.

The procedure to obtain certificates is:

1. In the Management Portal, go to the **Public Key Infrastructure** page (**System Administration** > **Security** > **Public Key Infrastructure**).
2. On the **Public Key Infrastructure** page, under **Certificate Authority Client**, select **Get Certificate(s) from Certificate Authority server**. This displays a list of available certificates to download, as well as a button that displays certificates issued for the current instance (whether downloaded or not). Ordinarily, you need both the CA server certificate as well as your own. There are several tasks you can perform from this page:
   - To download the CA certificate, click **Get Certificate Authority Certificate**. A confirmation message such as

     ```
     Certificate Authority Certificate (SHA-1 Fingerprint:
     E2:FB:30:09:53:90:9A:31:30:C3:F0:07:8F:64:65:CD:11:0A:1A:A2)
     saved in file "c:\intersystems\myinstnace\mgr\MyCA.cer"
     ```

   - To download any certificate that the CA has issued — including any certificate for the CA client itself, you can locate the certificate by its serial number, the name of the host of the CA client (the **Hostname** column), the name of the instance of the CA client (the **Instance** column), or the root file name of the certificate (the **Filename** column).
   - To view any certificates issued for the current instance, click **Show Certificates for This Instance**. This displays a table of certificates from which you can download a certificate, listing only the **Serial Number** and **Filename** columns.
3. When you click **Get** to download a certificate, Caché displays a confirmation message, such as

   ```
   Certificate number 74 (SHA-1 Fingerprint:
   45:E8:DE:0C:15:BF:A7:89:58:04:5E:68:2E:4D:BB:01:F5:90:94:97)
   saved in file "c:\intersystems\myinstance\mgr\IstanbulAcctsPayable.cer"
   ```

While Caché initially downloads certificates to the manager's directory, once they are on the client host, you can move them anywhere.

15

Using Delegated Authentication

Caché supports the use of user-defined authentication mechanisms. This is known as *delegated authentication*. Delegated authentication allows administrators to implement custom mechanisms to replace the authentication and role-management activities that are part of Caché security.

This chapter covers the following topics:

- Overview of Delegated Authentication
- Creating Delegated (User-Defined) Authentication Code
- Setting Up Delegated Authentication
- After Delegated Authentication Succeeds

15.1 Overview of Delegated Authentication

To use delegated authentication, the steps are:

1. Create the user-defined authentication code in the **ZAUTHENTICATE** routine. This can include the use of two-factor authentication. This routine can also perform basic setup for a user account, such as specifying roles and other user properties.

   If you are using HealthShare Health Connect, create a custom **ZAUTHENTICATE** routine as described in this guide. Do not use the **ZAUTHENTICATE** routine that is part of the HSLIB namespace.

   If you are using HealthShare Unified Care Record, you cannot create a custom version of **ZAUTHENTICATE** to implement delegated authentication because Unified Care Record comes with its own version of the routine. Instead, you must customize methods in the class HS.Local.ZAUTHENTICATE. For more information, see "Unified Care Record's Authentication Mechanism" in the *Unified Care Record Security Guide*.
2. Enable delegated authentication for the Caché instance on the Authentication Options page.
3. Enable delegated authentication for the relevant services, applications, or both, as required.
4. Optionally enable two-factor authentication for the Caché instance and, if required, for web applications and client/server applications.

For example, to use delegated authentication for an instance's Management Portal, the steps are:

1. Create the user-defined authentication code in **ZAUTHENTICATE**.
2. Enable delegated authentication for the Caché instance as a whole.

3. Enable delegated authentication for the set of /csp/sys* applications.

15.1.1 How Delegated Authentication Works

When a user attempts to log in and Caché invokes delegated authentication, the sequence of events is:

1. When a service or application uses delegated authentication, a login attempt automatically results in a call to the **ZAUTHENTICATE** routine. The authentication code in this routine can be any user-defined ObjectScript, class methods, or **$ZF** callout code.
2. The next step depends on whether or not authentication succeeds and whether or not this is the first login using **ZAUTHENTICATE**:
   - If **ZAUTHENTICATE** succeeds and this is the first time that the user has been authenticated through this mechanism, the user is added to the list of Caché users with a type of "Delegated user". If **ZAUTHENTICATE** sets roles or other characteristics, these become part of the user's properties.
   - If **ZAUTHENTICATE** succeeds and this is not the first login, **ZAUTHENTICATE** updates the user's properties.
   - If **ZAUTHENTICATE** fails, then the user receives an access denied error and is unable to access the system. To determine why this has occurred:
     a. Check the **Reason for Failing to Login** field in the User Profile.
     b. For more detailed information, check the audit log for the relevant %System/%Login/LoginFailure event. If auditing or the LoginFailure event are not enabled, you may need to enable both of these and then re-create the circumstances of the login failure.
3. If two-factor authentication is enabled for the instance and the relevant services, then there is a check that the user's PhoneNumber and PhoneProvider properties have been set. If these properties are set, then two-factor authentication proceeds; if they are not set, two-factor authentication cannot proceed and the user is not authenticated.
4. A delegated user is listed as such in the Type column of the list of users on the **Users** page (**System Administration** > **Security** > **Users**). The user's properties are displayed read-only in the Management Portal and are not editable from within Caché (since all the information comes from outside Caché).

   Note: A delegated user cannot also be a Caché password user.

15.2 Creating Delegated (User-Defined) Authentication Code

This section describes various aspects of creating your own **ZAUTHENTICATE** routine:

- Authentication Code Fundamentals
- Signature
- Authentication Code
- Setting Values for Roles and Other User Characteristics
- Return Value and Error Messages

15.2.1 Authentication Code Fundamentals

A system-supplied template of **ZAUTHENTICATE** is available in the SAMPLES namespace in the routine ZAUTHENTICATE.mac for instances with Manager Utility Source Code installed. To install Manager Utility Source Code, select that option as part of a Custom install on the **Setup Type** page of the installation wizard.

To create your own ZAUTHENTICATE.mac:

1. To use ZAUTHENTICATE.mac as a template, copy its contents and save them into a ZAUTHENTICATE.mac routine in the %SYS namespace.
2. Review the comments in the system-supplied code for **ZAUTHENTICATE**. These provide important guidance about how to implement a custom version of the routine.
3. Edit the routine by adding custom authentication code and any desired code to set user account characteristics.

CAUTION: Because Caché places no constraints on the authentication code in **ZAUTHENTICATE**, the application programmer is responsible for ensuring that this code is sufficiently secure.

15.2.2 Signature

The signature of **ZAUTHENTICATE** is:

```
ZAUTHENTICATE(ServiceName, Namespace, Username, Password, Credentials,
        Properties) PUBLIC {

    // authentication code
    // optional code to specify user account properties and roles
}
```

where:

- *ServiceName* — A string representing the name of the service through which the user is connecting to Caché, such as **%Service_Console** or **%Service_CSP**.
- *Namespace* — A string representing the namespace on the Caché server to which a connection is being established. This is for use with the **%Service_Bindings** service, such as with Studio or ODBC.
- *Username* — A string representing the name of the account entered by the user that is to be validated by the routine's code.
- *Password* — A string representing the password entered by the user that is to be validated.
- *Credentials* — *Passed by reference.* Not implemented in this version of Caché.
- *Properties* — *Passed by reference.* An array of returned values that defines characteristics of the account named by *Username*.

When Caché calls **ZAUTHENTICATE**, it has values for these arguments and supplies them to the routine.

15.2.3 Authentication Code

The content of authentication code is application specific. If authentication succeeds, the routine should return the $$$OK macro; otherwise, it should return an error code. See the section "Return Value and Error Messages" for more information on return values.

CAUTION: Because Caché does not and cannot place any constraints on the authentication code in **ZAUTHENTICATE**, the application programmer is responsible for ensuring that this code is sufficiently secure.

15.2.3.1 The GetCredentials Entry Point

ZAUTHENTICATE includes an **GetCredentials** entry point. This entry point is called whenever delegated authentication is enabled for a service, and is called before the user is prompted for a username and password. Instead of getting a username and password from the user, code in the function (created by the application developer) specifies the username and password. The username and password returned are then authenticated in the normal manner as if the user entered them. A possible use of this mechanism is to provide a username and password within the entry point and then, within authentication code, to $roles for the process.

The username and password returned from this entry point can be obtained by any mechanism that the application developer chooses. They can come from a global, an external DLL or LDAP call, or simply set within the routine. The application developer could even provide code to prompt for the username and password, such as in a terminal connection or with a custom CSP login page.

If the entry point returns an error status, then the error is logged in the audit log, and the user is denied access to the system; the one exception to this is if the error status `$SYSTEM.Status.Error($$$GetCredentialsFailed)` is returned, in which the normal username/password prompting proceeds.

In the following example of a **GetCredentials** entry point, the code performs different actions for different services:

- For **`%Service_Console`**, it does not prompt the user for any information and sets the process's username and password to `_SYSTEM` and `SYS`, respectively.
- For **`%Service_Bindings`**, it forces the user to provide a username and password.
- For web applications, it checks if the application in use is the `/csp/`samples application; if it is that application, it sets the username and password to `AdminUser` and `Test`. For all other web applications, it denies access.
- For any other service, it denies access.

Finally, the **Error** entry point performs clean-up as necessary.

The code is:

```
GetCredentials(ServiceName,Namespace,Username,Password,Credentials) Public {

  // For console sessions, authenticate as _SYSTEM.
  If ServiceName="%Service_Console" {
    Set Username="_SYSTEM"
    Set Password="SYS"
    Quit $SYSTEM.Status.OK()
  }

  // For the CSP samples application, authenticate as AdminUser.
  If $isobject($get(%request)) {
    If %request.Application="/csp/samples/" {
      Set Username="AdminUser"
      Set Password="Test"
      Quit $System.Status.OK()
    }
  }

  // For bindings connections, use regular prompting.
  If ServiceName="%Service_Bindings" {
    Quit $SYSTEM.Status.Error($$$GetCredentialsFailed)
  }

  // For all other connections, deny access.
  Quit $SYSTEM.Status.Error($$$AccessDenied)
}
```

For more details, see the comments for this entry point in **ZAUTHENTICATE.mac**.

15.2.3.2 The SendTwoFactorToken Entry Point

ZAUTHENTICATE includes an **SendTwoFactorToken** entry point. This entry point is for use with two-factor authentication. If it is defined and the Caché instance has two-factor authentication enabled, then you can override the default system

setting for the format of the message and token that the instance sends to the user's mobile phone. This allows for messages that can vary by application even on the same Caché instance.

For more details and an example of how to use this entry point, see this entry point in the sample **ZAUTHENTICATE.mac**.

15.2.4 Setting Values for Roles and Other User Characteristics

If initial authentication succeeds, **ZAUTHENTICATE** can establish the roles and other characteristics for the authenticated user. For subsequent logins, **ZAUTHENTICATE** can update these elements of the user record.

For this to happen, code in **ZAUTHENTICATE** sets the values of the *Properties* array. (*Properties* is passed by reference to **ZAUTHENTICATE**.) Typically, the source for the values being set is a repository of user information that is available to **ZAUTHENTICATE**.

15.2.4.1 User Properties

The elements in the *Properties* array are:

- *Properties("Comment")* — Any text
- *Properties("FullName")* — The first and last name of the user
- *Properties("NameSpace")* — The default namespace for a Terminal login
- *Properties("Roles")* — The comma-separated list of roles that the user holds in Caché
- *Properties("Routine")* — The routine that is executed for a Terminal login
- *Properties("Password")* — The user's password
- *Properties("Username")* — The user's username
- *Properties("PhoneNumber")* — The user's mobile phone number, for use with two-factor authentication
- *Properties("PhoneProvider")* — The user's mobile phone's service provider, for use with two-factor authentication

Each of these elements is described in more detail in one of the following sections.

Note: The value of each element in the properties array determines the value of its associated property for the user being authenticated. It is not possible to use only a subset of the properties or to manipulate their values after authentication.

Comment

If **ZAUTHENTICATE** sets the value of *Properties("Comment")*, then that string becomes the value of the user account's *Comment* property in Caché. (This property is described in the section "Properties of Users," in the "Users" chapter.) If no value is passed back to the calling routine, then the value of *Comment* for the user account is a null string and the relevant field in the Management Portal then holds no content.

FullName

If **ZAUTHENTICATE** sets the value of *Properties("FullName")*, then that string becomes the value of the user account's *Full name* property in Caché. (This property is described in the section "Properties of Users," in the "Users" chapter.) If no value is passed back to the calling routine, then the value of *Full name* for the user account is a null string and the relevant field in the Management Portal then holds no content.

NameSpace

If **ZAUTHENTICATE** sets the value of *Properties("Namespace")*, then that string becomes the value of the user account's *Startup Namespace* property in Caché. (This property is described in the section "Properties of Users," in the "Users" chapter.) If no value is passed back to the calling routine, then the value of *Startup Namespace* for the user account is a null string and the relevant field in the Management Portal then holds no content.

Once connected to Caché, the value of *Startup Namespace* (hence, that of *Properties("Namespace")*) determines the initial namespace for any user authenticated for local access (such as for Console, Terminal, or Telnet). If *Startup Namespace* has no value (since *Properties("Namespace")* has no value), then the initial namespace for any user authenticated for local access is determined as follows:

1. If the USER namespace exists, that is the initial namespace.
2. If the USER namespace does not exist, the initial namespace is the %SYS namespace.

Note: If the user does not have the appropriate privileges for the initial namespace, access is denied.

Password

If **ZAUTHENTICATE** sets the value of *Properties("Password")*, then that string becomes the value of the user account's *Password* property in Caché. (This property is described in the section "Properties of Users," in the "Users" chapter.) If no value is passed back to the calling routine, then the value of *Password* for the user account is a null string and the relevant field in the Management Portal then holds no content.

Roles

If **ZAUTHENTICATE** sets the value of *Properties("Roles")*, then that string specifies the *Roles* to which a user is assigned; this value is a string containing a comma-delimited list of roles. If no value is passed back to the calling routine, then the value of *Roles* for the user account is a null string and the relevant field in the Management Portal then holds no content. Information about a user's roles is available on the **Roles** tab of a user's **Edit User** page.

If any roles returned in *Properties("Roles")* are not defined, then the user is not assigned to the role.

Hence, the logged-in user is assigned to roles as follows:

- If a role is listed in *Properties("Roles")* and is defined by the Caché instance, then the user is assigned to the role.
- If a role is listed in *Properties("Roles")* and is not defined by the Caché instance, then the user is not assigned to the role.
- A user is always assigned to those roles associated with the _PUBLIC user. A user also has access to all public resources. For information on the _PUBLIC user, see the section "The _PUBLIC Account" in the "Users" chapter; for information on public resources, see the section "Services and Their Resources" in the "Resources" chapter.

Routine

If **ZAUTHENTICATE** sets the value of *Properties("Routine")*, then that string becomes the value of the user account's *Startup Tag^Routine* property in Caché. (This property is described in the section "Properties of Users," in the "Users" chapter.) If no value is passed back to the calling routine, then the value of *Startup Tag^Routine* for the user account is a null string and the relevant field in the Management Portal then holds no content.

If *Properties("Routine")* has a value, then this value specifies the routine to execute automatically following login on a terminal-type service (such as for Console, Terminal, or Telnet). If *Properties("Routine")* has no value, then login starts the Terminal session in programmer mode.

Username

If the username property is returned by this function, then that username is written to the Caché database. This gives the user a chance to normalize what was entered by the user at the username prompt. Note that the normalized username must only differ by case. If the *Username* property is not passed back to the calling routine, then the username entered by the user at the username prompt will be used as the username written to the Caché security databases (that is, it is not normalized).

If **ZAUTHENTICATE** sets the value of *Properties("Username")*, then that string becomes the value of the user account's *Name* property in Caché. (This property is described in the section "Properties of Users", in the "Users" chapter.) This provides the application programmer with an opportunity to normalize content provided by the end-user at the login prompt.

If there is no explicit call that passes the value of *Properties("Username")* back to the calling routine, then there is no normalization and the value entered by the end-user at the prompt serves as the value of the user account's *Name* property without any modification.

PhoneNumber and PhoneProvider

These are properties associated with two-factor authentication.

If **ZAUTHENTICATE** sets the value of *Properties("PhoneNumber")* and *Properties("PhoneProvider")*, then these then these are written to the Caché database for the user as the user's mobile phone number and mobile phone service provider. If these are not passed back to the calling routine, then the phone number and service provider written to the Cache security database are a null string. Hence, to use two-factor authentication with delegated authentication, you must supply both of these.

15.2.4.2 The User Information Repository

ZAUTHENTICATE can refer to any kind of repository of user information, such as a global or an external file. It is up to the code in the routine to set any external properties in the *Properties* array so that the authenticated user can be created or updated with this information. For example, while a repository can include information such as roles and namespaces, **ZAUTHENTICATE** code must make that information available to Caché.

If information in the repository changes, this information is only propagated back into the Caché user information if there is code in **ZAUTHENTICATE** to perform this action. Also, if there is such code, changes to users' roles must occur in the repository; if you change a user's roles during a session, the change does not become effective until the next login, at which point the user's roles are re-set by **ZAUTHENTICATE**.

15.2.5 Return Value and Error Messages

The routine returns one of the following values:

- Success — $$$OK. This indicates that username/password combination was successfully authenticated
- Failure — $SYSTEM.Status.Error($$$ERRORMESSAGE). This indicates that authentication failed.

ZAUTHENTICATE can return system-defined or application-specific error messages. All these messages use the **Error** method of the %SYSTEM.Status class. This method is invoked as **$SYSTEM.Status.Error** and takes one or two arguments, depending on the error condition.

The available system-defined error messages are:

- **$SYSTEM.Status.Error($$$AccessDenied)** — Error message of "Access Denied"
- **$SYSTEM.Status.Error($$$InvalidUsernameOrPassword)** — Error message of "Invalid Username or Password"
- **$SYSTEM.Status.Error($$$UserNotAuthorizedOnSystem,Username)** — Error message of "User *Username* is not authorized"
- **$SYSTEM.Status.Error($$$UserAccountIsDisabled,Username)** — Error message of "User *Username* account is disabled"
- **$SYSTEM.Status.Error($$$UserInvalidUsernameOrPassword,Username)** — Error message of "User *Username* invalid name or password"
- **$SYSTEM.Status.Error($$$UserLoginTimeout)** — Error message of "Login timeout"
- **$SYSTEM.Status.Error($$$UserCTRLC)** — Error message of "Login aborted"
- **$SYSTEM.Status.Error($$$UserDoesNotExist,Username)** — Error message of "User *Username* does not exist"
- **$SYSTEM.Status.Error($$$UserInvalid,Username)** — Error message of "Username *Username* is invalid"
- **$SYSTEM.Status.Error($$$PasswordChangeRequired)** — Error message of "Password change required"

- **$SYSTEM.Status.Error($$$UserAccountIsExpired,Username)** — Error message of "User *Username* account has expired"
- **$SYSTEM.Status.Error($$$UserAccountIsInactive,Username)** — Error message of "User *Username* account is inactive"
- **$SYSTEM.Status.Error($$$UserInvalidPassword)** — Error message of "Invalid password"
- **$SYSTEM.Status.Error($$$ServiceDisabled,ServiceName)** — Error message of "Logins for Service *Servicename* are disabled"
- **$SYSTEM.Status.Error($$$ServiceLoginsDisabled)** — Error message of "Logins are disabled"
- **$SYSTEM.Status.Error($$$ServiceNotAuthorized,ServiceName)** — Error message of "User not authorized for service"

To generate a custom message, use the **$SYSTEM.Status.Error()** method, passing it the `$$$GeneralError` macro and specifying any custom text as the second argument. For example:

```
$SYSTEM.Status.Error($$$GeneralError,"Any text here")
```

Note that when an error message is returned to the caller, it is logged in the audit database (if LoginFailure event auditing is turned on). However, the only error message the user sees is $SYSTEM.Status.Error($$$AccessDenied). However, the user also sees the message for the `$$$PasswordChangeRequired` error. Return this error if you want the user to change from the current to a new password.

15.3 Setting Up Delegated Authentication

Once you have created a **ZAUTHENTICATE** routine to perform authentication (and, optionally, authorization tasks), the next step is to enable it for the instance's relevant services or applications. This procedure is:

1. Enable delegated authentication for entire instance. On the **Authentication/CSP Session Options** page (**System Administration** > **Security** > **System Security** > **Authentication/CSP Session Options**), select **Allow Delegated authentication** and click **Save**.

   With delegated authentication enabled for the instance, a **Delegated** check box appears on the **Edit Service** page for relevant services and the **Edit Web Application** page for those applications.
2. Enable delegated authenticated for services and applications, as appropriate.

The following services support delegated authentication:

- **%Service_Bindings**
- **%Service_CSP**
- **%Service_CacheDirect**
- **%Service_CallIn**
- **%Service_ComPort**
- **%Service_Console**
- **%Service_Login**
- **%Service_Terminal**
- **%Service_Telnet**

These fall into several categories of access modes:

- Local access —

  %Service_CallIn, **%Service_ComPort**, **%Service_Console**, **%Service_Login**, **%Service_Terminal**, **%Service_Telnet**

  To use delegated authentication with local connections, enable it for the service.

- Client/Server access —

  %Service_Bindings, **%Service_CacheDirect**,

  To use delegated authentication with client/server connections, enable it for the service.

- CSP access —

  %Service_CSP

  To use delegated authentication with web-based connections (through CSP or Zen), enable it for the web application. You may also enable it for the CSP gateway by enabling the service **%Service_CSP**

15.4 After Delegated Authentication Succeeds

Once the user has authenticated, two important topics are:

- The State of the System
- Changing Passwords

15.4.1 The State of the System

Any user who is initially authenticated using delegated authentication is listed in the table of users on the **Users** page (**System Administration** > **Security** > **Users**) as having a type of "Delegated user". If a system administrator has explicitly created a user through the Management Portal (or using any other native Caché facility), that user has a type of "Caché password user". If a user attempts to log in using delegated authentication and is successfully authenticated, Caché determines that this user already exists as a Caché user — not a Delegated user — and so login fails.

15.4.2 Changing Passwords

The **ZAUTHENTICATE** routine also includes an entry point, **ChangePassword**, to include code to change a user's password. The signature of this entry point is:

```
ChangePassword(Username,NewPassword,OldPassword,Status) Public {}
```

where

- *Username* is a string specifying the user whose password is being changed.
- *NewPassword* is a string specifying the new value of the user's password.
- *OldPassword* is a string specifying the old value of the user's password.
- *Status* (passed by reference) receives a Caché status value indicating either that the password change has been successful or specifying the error that caused the routine to fail.

16

Using LDAP

Caché provides support for authentication and authorization using LDAP, the Lightweight Directory Access Protocol. LDAP systems have a central repository of user information, from which Caché retrieves information. For example, on Windows, a domain controller using Active Directory is an LDAP server. This means that an instance of Caché running on this type of Windows domain can use LDAP for its authentication; if an instance is using LDAP authentication, it then also has the option of using LDAP authorization.

This chapter covers the following topics:

- Overview of Using LDAP with Caché
- Configuring Caché to Use an LDAP Server
- Setting Up LDAP-Based Authentication
- After Authentication — The State of the System
- Configuring the LDAP Server to Use Registered LDAP Properties
- Using LDAP Authorization with OS-Based Authentication

A Note on Securing Outbound LDAP Connections

This chapter describes using LDAP for authentication and authorization when connecting to Caché. If you are establishing an outbound connection from Caché to an LDAP server and need to secure the connection, Caché includes support for TLS for these purposes. This is described in the class documentation for %SYS.LDAP, in the content for the Init method.

16.1 Overview of Using LDAP with Caché

The procedure for configuring a Caché service or application to use an existing LDAP server for authentication (and optional authorization) is as follows:

1. Configure the LDAP server to use the registered LDAP properties.
2. On the **LDAP Options** page, configure Caché to use the LDAP server. This includes specifying the names of LDAP user properties to be used for setting the values of properties of Caché users.
3. Set up Caché to use LDAP. This involves enabling LDAP for the entire instance of Caché and then enabling it for the relevant services or applications.

Note: This chapter primarily discusses the use of LDAP for both authentication and authorization. For information on using OS-based authentication in conjunction with LDAP authorization, see the Using LDAP Authorization with OS-Based Authentication section.

When a user attempts to authenticate to an instance of Caché that uses LDAP authentication, the process is:

1. The user is prompted for a user name and password. This user, who is trying to authenticate, is known as the "target user."
2. Caché establishes a connection to the LDAP server using the values specified for the *LDAP username to use for searches* and *LDAP username password*. This user, who has privileges to search the LDAP database so that Caché can retrieve information, is known as the "search user."
3. Once the connection is established, the next step is to look up the user in the LDAP database using the *LDAP Unique search attribute*.
4. If Caché locates the target user in the LDAP database, it retrieves the attributes associated with the user, such as the user's initial roles and namespace.
5. Caché then attempts to authenticate the user to the LDAP database, using the user name and password provided in step 1.
6. If authentication succeeds, the user can then interact with Caché based on the privileges associated with the user's roles and any publicly available resources. The user's properties are displayed read-only in the Management Portal and are not editable from within Caché (since all the information is coming from outside Caché).

Notes on Using LDAP with Caché

Note the following points when using LDAP with Caché:

- If you have more complex authentication and authorization requirements, it is also possible to use delegated authentication to authenticate using the LDAP server. This requires that you use calls to the %SYS.LDAP class as part of the custom authentication code in the **ZAUTHENTICATE** routine. For sample code that performs these actions, see the **LDAP.mac** routine in the SAMPLES namespace. For more details about delegated authentication and the **ZAUTHENTICATE** routine, see the "Delegated Authentication" chapter.
- Caché supports LDAP version 3 protocols. Earlier LDAP protocols are not supported.
- If you using the Caché LDAP APIs with certificates on UNIX® and need detailed debugging information, you may wish to use the ldapsearch program that is part of the OpenLDAP package. Once you have corrected any problems with certificates, the Caché LDAP APIs should also be corrected. The ldapsearch program may also be useful for debugging other LDAP connection problems.
- If you need to authenticate to LDAP or use Caché authentication after collecting credentials through another mechanism, call **$SYSTEM.Security.Login** with those credentials to authenticate the user.

16.1.1 Using LDAP Authorization

In addition to performing authentication with LDAP, Caché supports LDAP authorization. To perform LDAP authorization with Caché, InterSystems recommends the use of LDAP groups.

LDAP groups are a way to assign privileges to users using an LDAP or Active Directory server. The names of groups are specified by the schema on the local LDAP server; typically, the local LDAP administrator defines these names (though Caché uses the predefined name structure described below). Each group has a distinguished name (DN) that uniquely identifies it. You can then assign each user to an LDAP group, and that group then determines the user's access to Caché roles, routines, and namespaces. (Note that when defining these groups on your LDAP server, they should be created as *security* groups, and not *distribution* groups.)

Caché´ supports three predefined structures for group names. Each of these uses a prefix for the group, where the prefixes are `intersystems-Role-`, `intersystems-Namespace-`, and `intersystems-Routine-`, so conforming group names are of the form `intersystems-Role-`*role-name*, `intersystems-Namespace-`*namespace-name*, and `intersystems-Routine-`*routine-name*, and these names specify groups for initial roles, an initial namespace, and an initial routine, respectively. To use this feature, the names must follow this structure. (Names are not case-sensitive.)

Note: A user can only belong to one routine group and one namespace group.

For example, a user might have groups defined such as:

```
CN=intersystems-Role-Admin,OU=Groups,OU=Brasilia,DC=mysite,DC=com
CN=intersystems-Role-%Operator,OU=Groups,OU=Brasilia,DC=mysite,DC=com
CN=intersystems-Namespace-USER,OU=Groups,OU=Brasilia,DC=mysite,DC=com
CN=intersystems-Routine-INTEGRIT,OU=Groups,OU=Brasilia,DC=mysite,DC=com
```

In this case, after logging in, the user would get the roles **`Admin`** and **`%Operator`**, with a default namespace of USER, and a default routine of **INTEGRIT**.

If the user does not belong to any group with a name in this form, then Caché sets the value of that attribute for the user to ""; hence, if the user is not a member of a group that defines `intersystems-Routine-` as the initial portion of its name, then Caché sets the startup routine for that user to "" (and Caché places the user in programmer mode). If a user is assigned to a group that specifies a role for the user, but that role is not defined on the instance where they are logging in, then they do not have that role on that instance.

For example, suppose that a user belongs to the following groups:

```
intersystems-Role-Admin
intersystems-Role-Application1
intersystems-Role-Application2
```

If the role **`Application2`** is not defined on the system they are logging into, Caché only assigns the **`Admin`** and **`Application1`** roles.

Note: While InterSystems recommends the use of LDAP groups rather than the use of user-defined attributes for managing role, routine, and namespace definitions, Caché *does* support the use of user-defined attributes.

16.1.1.1 Setting Up a Role Required for Login

If you are using LDAP authentication or OS-based LDAP authorization — especially if your organization has multiple instances of Caché, then InterSystems *strongly* recommends that each instance have a role that is required for connecting to it:

1. For each instance, create the role that is required for login. Do this according to the instructions in the "Creating Roles" section of the "Roles" chapter.
2. For each instance, enter the newly created role in the **Role required to connect to this system** field on the **System Security Settings** page (**System Administration** > **Security** > **System Security** > **System-wide Security Parameters**).
3. Add an LDAP group with a name of form `intersystems-Role-`*rolename* that includes the name of the newly created role.

This mechanism prevents users from accessing instances where they are insufficiently privileged. Otherwise, a user who holds various roles on one instance may then have those same roles on an instance where this is not intended.

Important: If the **Role required to connect to this system** field is left blank, a user with sufficient general privileges can connect and gain access to a system that uses OS-based LDAP authorization, provided that they had enough roles assigned to them to log in.

For example, suppose there are two systems, *TEST* and *PRODUCTION*. To secure each of these productions, create a role on **TEST** called **TESTACCESS** and a role on **PRODUCTION** called **PRODUCTIONACCESS**. On **TEST**, set the value of the **Role required to connect to this system** field to **TESTACCESS**; on **PRODUCTION**, set it to **PRODUCTIONACCESS**. Then, if a user is only allowed to access the **TEST** system, assign that user the **TESTACCESS** role *only* and do not assign the **PRODUCTIONACCESS** role to the user. For users who can access either system, assign them both **PRODUCTIONACCESS** and **TESTACCESS** roles.

16.1.1.2 Using Nested Groups

On Active Directory, LDAP groups include support for what are known as *nested groups*. A nested group is a group that is a member of a second group, which means that all the users who are members of the nested group are also members of the second group. For example, suppose that there are two LDAP groups defined, known as ABC and DEF. You can make the ABC group a member of the DEF group. This means that if a user is a member of the ABC group, then they are also a member of the DEF group without explicitly assigning the user to that group.

Note: Depending on the structure and size of the local LDAP database, using nested groups can slow down users' login time. Hence, when configuring Caché to use an LDAP server, if you select **Search nested Groups for Roles/Routine/Namespace**, this can have an effect on performance: selecting this returns nested groups and has slower logins, while not selecting it does not return nested groups and is faster.

16.2 Configuring Caché to Use an LDAP Server

This section describes the following topics:

- Specifying Configuration Information for LDAP in Caché
- Specifying a Certificate File on Windows
- Searching the LDAP Database

16.2.1 Specifying Configuration Information for LDAP in Caché

To use LDAP authentication with Caché, the first step is to configure Caché for its interactions with the LDAP server. This includes information required to:

- Connect to and query the LDAP server
- Retrieve the required information about the user being authenticated

All this information is specified on the Management Portal **LDAP Options** page (**System Administration** > **Security** > **System Security** > **LDAP Options**).

When this page is first displayed, only a single field is visible, the **Allow LDAP authentication** field. Checking this field displays all the fields for configuration information for connecting the Caché instance to the LDAP server (false by default). There are a standard set of fields and there can also be customized fields. When connecting to an LDAP server, Caché refers to various fields for its information; the end-user can do the same. The fields are:

- **LDAP server is a Windows Active Directory server** — *Windows only.* Specifies whether or not the LDAP server is a Windows Active Directory server (true by default).
- **LDAP domain name** — *Windows only.* Specifies the name of the domain where the LDAP server is located. This field is only editable for a Windows Active Directory server. This field is populated with a value based on the IP address of the DNS name of the current LDAP server.

- **LDAP host names** — Specifies the name(s) of the host(s) on which the LDAP server is running. The complexity of each host name can range from an unqualified host name to fully-qualified host name with a port number; the required form of the host name(s) depends on the particular configuration. This field is populated with a value based on the IP address of the DNS name of the current LDAP server.

  To specify multiple host names, separate the names by spaces. If the LDAP server is configured to use a particular port, you can specify it by appending ":*portname*" to the host name; typical usage is not to specify a port and to let the LDAP functions use the default port, such as:

  ```
  ldapserver.example.com
  ldapserver.example.com ldapbackup.example.com
  ```

- LDAP search information — varies by operating system:

  – **LDAP username to use for searches** — *Windows only.* The user name provided to the LDAP server to establish an initial connection and which is used to perform LDAP searches and lookups. This user is also known as the "search user."

    The search user must have permission to read the entire LDAP database. It is important to ensure that the search user has uninterrupted access to the LDAP database; for example, to establish this on Windows, set its *User cannot change password* and *Password never expires* attributes and set the value of its *Account expires* attribute to `Never`.

    For more information on searching the LDAP database, see the section "Searching the LDAP Database."

  – **LDAP DN to use for searches** — *UNIX® only.* The Distinguished Name (DN) of the user provided to the LDAP server to establish an initial connection and which is used to perform LDAP searches and lookups. This user is also known as the "search user."

    The search user must have permission to read the entire LDAP database. It is also important to ensure that the search user has uninterrupted access to the LDAP database. For example, the user's operating-system account should be set so that:

    - The user cannot change the account's password
    - The password never expires
    - The account never expires

    For example, if the search user is "ldapsearchuser", the Distinguished Name might be as follows:

    ```
    uid=ldapsearchuser,ou=People,dc=example,dc=com
    ```

    For more information on searching the LDAP database, see the section "Searching the LDAP Database."

- **LDAP username password** — Specifies the password associated with account used for the initial connection.
- **LDAP Base DN to use for searches** — Specifies the point in the directory tree from which searches begin. This typically consists of domain components, such as `DC=intersystems,DC=com`.
- **LDAP Unique search attribute** — Specifies a unique identifying element of each record, which therefore makes it appropriate for searches. For more information on searching the LDAP database, see the section "Searching the LDAP Database."
- **Use TLS/SSL encryption for LDAP sessions** — Specifies if TLS/SSL encryption protect data being passed between the Caché server and the LDAP server (false by default).
- **TLS/SSL certificate file** — *UNIX® only.* Specifies the location of the file containing any TLS/SSL certificates (in PEM format) being used to authenticate the server certificate, if needed. For Windows, see "Specifying a Certificate File on Windows."

- **Use LDAP Groups for Roles/Routine/Namespace** — Specifies if the user's roles, routine, and namespace come from the user's group memberships (true by default); if not, then they come from the attribute fields of the user's LDAP record. Selecting the use of groups enables and disables other fields (see each subsequent field for details).

  Note: InterSystems recommends the use of LDAP groups for authorization, rather than LDAP attributes (including InterSystems registered LDAP properties). If you have existing code or are otherwise required to use registered properties, see the "Configuring the LDAP Server to Use Registered LDAP Properties" section for details.

- **Search nested Groups for Roles/Routine/Namespace** (only active if LDAP groups are selected and only relevant for Active Directory) — Specifies if search returns all a user's nested groups. See the "Using Nested Groups" section for more information on nested groups.
- **User attribute to retrieve comment attribute** — Specifies the attribute whose value is the source for the *Comment* property for a user. This property is described in the section Properties of Users, in the "Users" chapter.
- **User attribute to retrieve full name from** — Specifies the attribute whose value is the source for the *Full name* property for a user. This property is described in the section Properties of Users, in the "Users" chapter.
- **User attribute to retrieve default namespace** (not active if LDAP groups are selected) — Specifies the attribute whose value is the source for the *Startup namespace* property for a user. This property of a Caché user is described in the section Properties of Users, in the "Users" chapter; this LDAP property is described in the section "Registered LDAP Properties."
- **User attribute to retrieve default routine** (not active if LDAP groups are selected) — Specifies the attribute whose value is the source for the *Tag^Routine* property for a user. This property of a Caché user is described in the section Properties of Users, in the "Users" chapter; this LDAP property is described in the section "Registered LDAP Properties."
- **User attribute to retrieve roles** (not active if LDAP groups are selected) — Specifies the attribute whose value determines the roles to which a user is assigned. When creating this attribute, it must be specified as an LDAP multivalued attribute. For information about a Caché user's roles, see the **Roles** tab of a user's **Edit User** page; this LDAP property is described in the section "Registered LDAP Properties."
- **LDAP attributes to retrieve for each user** — Specifies any attributes whose values are the source for any application-specific variables. Application code can then use the **Get** method of the Security.Users class to return this information.

16.2.2 Specifying a Certificate File on Windows

On Windows, to specify the location of a file containing any TLS/SSL certificates (in PEM format) being used to authenticate the server certificate to establish a secure LDAP connection, use Microsoft Certificate Services.

Note: Certificates must be installed in the Certificates (Local Computer)\Trusted Root Certification Authorities certificate store.

16.2.3 Searching the LDAP Database

Once Caché has established a connection to the LDAP server as the search user, the next step is to retrieve information about the target user. To do this, Caché checks the user name provided at login against values in the LDAP database for the *LDAP Unique search attribute*. (The name of this attribute is often "sAMAcccountName" for a Windows LDAP server and "uid" for a UNIX® LDAP server.)

When looking up a user with the *LDAP Unique search attribute*, the form of the user name depends on the operating system of both the Caché instance and the LDAP server:

- When connecting from Windows to Windows, it can simply be login name of the user, such as "testuser".

- For any other type of connection, it must include the full DN (distinguished name) of the user. When connecting from UNIX® to a Windows LDAP server, this is case-sensitive. For example:
  - When connecting from UNIX® to Windows, it might be `CN=testuser,OU=Users,OU=Cambridge`.
  - When connecting from any operating system to UNIX®, it might be `uid=testuser,ou=people`.

Once Caché has located the user, it retrieves attribute information. It retrieves information about every named attribute in the Caché LDAP configuration fields (described in "Specifying Configuration Information for LDAP in Caché"), and it retrieves all values associated with each attribute. Note that Caché retrieves all values associated with all attributes specified for the user in the Caché LDAP configuration fields; it is not possible to configure it to retrieve only a subset of these.

16.3 Setting Up LDAP-Based Authentication

Once Caché has been configured for use with LDAP, the next step is to enable it for the instance's relevant services or applications. This procedure is:

1. Enable LDAP authentication for entire instance:
   a. From the Management Portal home page, go to the **Authentication/CSP Session Options** page (**System Administration** > **Security** > **System Security** > **Authentication/CSP Session Options**), select **Allow LDAP authentication**.
   b. On the **Authentication Options** page, you also have the option of selecting **Allow LDAP cache credentials authentication**. If this is selected and Caché is unable to connect to the LDAP server, then Caché uses the copy of the LDAP credentials that it last cached to authenticate the user. This can be useful if there is a failure in the connection to the LDAP server or if the LDAP server becomes unavailable.

      Note: If a user password has changed on the LDAP server after the last login and Caché uses the LDAP credentials cache for authentication, authentication can only occur with the old password.

   c. Click **Save** to apply the changes.

   With LDAP authentication enabled for the instance, an **LDAP** check box appears on the **Edit Service** page for relevant services and the **Edit Web Application** page for those applications.
2. Enable LDAP authentication for services and applications, as appropriate.

The following services support LDAP authentication:

- **`%Service_Bindings`**
- **`%Service_CSP`**
- **`%Service_CacheDirect`**
- **`%Service_CallIn`**
- **`%Service_ComPort`**
- **`%Service_Console`**
- **`%Service_Login`**
- **`%Service_Terminal`**
- **`%Service_Telnet`**

These fall into several categories of access modes:

- Local access —

  %Service_CallIn, **%Service_ComPort**, **%Service_Console**, **%Service_Login**, **%Service_Terminal**, **%Service_Telnet**

  To use LDAP authentication with local connections, enable it for the service.

- Client/Server access —

  %Service_Bindings, **%Service_CacheDirect**,

  To use LDAP authentication with client/server connections, enable it for the service.

- Web access —

  %Service_CSP

  To use LDAP authentication with web connections (through CSP or Zen), enable it for the web application. Enabling it for the service also allows the CSP Gateway itself to authenticate using LDAP authentication.

16.4 After Authentication — The State of the System

Any user who is initially authenticated using LDAP authentication is listed in the table of users on the **Users** page (**System Administration** > **Security** > **Users**) as having a Type of "LDAP user". If a system administrator has explicitly created a user through the Management Portal (or using any other native Caché facility), that user has a type of "Caché password user". If a user attempts to log in using LDAP authentication and is successfully authenticated, Caché determines that this user already exists as a Caché user — not an LDAP user — and so login fails.

If you wish to have any LDAP attributes of the current user stored on the Caché server, specify these in the LDAP attributes to retrieve for each user field of the **LDAP Options** page (**System Administration** > **Security** > **System Security** > **LDAP Options**). Your application can then use the **Get** method of the Security.Users class to return this information.

16.5 Configuring the LDAP Server to Use Registered LDAP Properties

Important: InterSystems recommends using LDAP groups for LDAP authorization.

InterSystems has registered three field names that are available for use with an LDAP schema to store authorization information. Each has its own dedicated purpose:

- *intersystems-Namespace* — The name of the user's default namespace (LDAP OID 1.2.840.113556.1.8000.2448.2.1).
- *intersystems-Routine* — The name of the user's default routine (LDAP OID 1.2.840.113556.1.8000.2448.2.2).
- *intersystems-Roles* — The name of the user's login roles (LDAP OID 1.2.840.113556.1.8000.2448.2.3).

To use these attributes, the procedure on the LDAP server is:

1. Enable the attributes for use. To do this, modify the value of *objectClass* field in the LDAP schema by appending the `intersystemsAccount` value to its list of values. (`intersystemsAccount` has an LDAP OID of 1.2.840.113556.1.8000.2448.1.1.)
2. Add the fields (as few or as many as required) to the schema.

3. Populate their values for the entries in the LDAP database.

Note: It is not required to use the registered LDAP schema names. In fact, you may already use existing attributes from your LDAP schema.

For example, with a UNIX® LDAP server, to define the schema for using LDAP authentication with Caché, use the content that appears the following definitions:

```
# Attribute Type Definitions

attributetype ( 1.2.840.113556.1.8000.2448.2.1 NAME 'intersystems-Namespace'
        DESC 'InterSystems Namespace'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 SINGLE-VALUE )

attributetype ( 1.2.840.113556.1.8000.2448.2.2 NAME 'intersystems-Routine'
        DESC 'InterSystems Routine'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{128} SINGLE-VALUE )

attributetype ( 1.2.840.113556.1.8000.2448.2.3 NAME 'intersystems-Roles'
        DESC 'InterSystems Roles'
        EQUALITY caseIgnoreMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

# Object Class Definitions

objectclass ( 1.2.840.113556.1.8000.2448.1.1 NAME 'intersystemsAccount' SUP top AUXILIARY
        DESC 'Abstraction of an account with InterSystems attributes'
        MAY ( intersystems-Routine $ intersystems-Namespace $ intersystems-Roles ) )
```

This content goes to two locations:

- Place it in the intersystems.schema file in the /etc/openldap/schema/ directory.
- Include it, along with any other content, in the /etc/openldap/slapd.conf file.

16.6 Using LDAP Authorization with OS-Based Authentication

Caché allows you to configure your system to support operating-system–based authentication, and then to perform authorization via LDAP. This is known as *Operating System LDAP authorization*. It allows a user authenticate to Caché using credentials from the operating system login and to have the appropriate roles in Caché by having them automatically retrieved from an LDAP server. Operating system LDAP authorization is available in the Terminal on UNIX®, Linux, and MacOS, and in the Caché Console on Windows.

17

Using Delegated Authorization

Caché supports the use of user-defined authorization code. This is known as *delegated authorization*. Topics in this chapter include:

- Overview of Delegated Authorization
- Creating Delegated (User-Defined) Authorization Code
- Configuring an Instance to Use Delegated Authorization
- After Authorization — The State of the System

17.1 Overview of Delegated Authorization

Delegated authorization allows administrators to implement custom mechanisms to replace the role-assignment activities that are part of Caché security. For example, user-defined authorization code might look up a user's roles in an external database and provide that information to Caché.

To use delegated authorization, there are the following steps:

1. Creating Delegated (User-defined) Authorization Code in the **ZAUTHORIZE** routine.
2. Configuring an Instance to Use Delegated Authorization for the Caché instance.

Note: Delegated authorization is only supported with Kerberos and Operating-System–based authentication.

Interactions between Delegated Authentication and Delegated Authorization

Delegated authorization through **ZAUTHORIZE.mac** is not supported for use with delegated authentication. The routine for delegated authentication (**ZAUTHENTICATE**, which is described in the chapter "Using Delegated Authentication") provides support for authorization. When using **ZAUTHENTICATE**, you have the option to segregate authentication and authorization code.

17.2 Creating Delegated (User-defined) Authorization Code

Topics associated with creating delegated authorization code include:

17.2.1 Working from the ZAUTHORIZE.mac Template

A system-supplied copy of the **ZAUTHORIZE** routine is available in the SAMPLES namespace in the routine ZAUTHORIZE.mac for instances with the manager utility source code installed. To install the manager utility source code, select an installation type of Development, Server, or a Custom installation that includes the Caché Database Engine.

To create your own ZAUTHORIZE.mac:

1. To use ZAUTHORIZE.mac as a template, copy its contents and save them into a ZAUTHORIZE.mac routine in the %SYS namespace.
2. Review the comments in the system-supplied code for **ZAUTHORIZE**. These provide important guidance about how to implement a custom version of the routine.
3. Edit the routine by adding custom authorization code and any desired code to set user account characteristics.

CAUTION: Because Caché places no constraints on the authorization code in **ZAUTHORIZE**, the application programmer is responsible for ensuring that this code is sufficiently secure.

Upgrading Delegated Authorization Code

Before upgrading to a new version of Caché, check ZAUTHORIZE.mac to determine if your current authorization code needs any changes to support new functionality.

17.2.2 ZAUTHORIZE Signature

When configured for delegated authorization, the system automatically calls **ZAUTHORIZE** after authentication occurs. Caché supplies values for the parameters defined in the **ZAUTHORIZE** signature as necessary. The signature of **ZAUTHORIZE** is:

```
ZAUTHORIZE(ServiceName, Namespace, Username, Password,
        Credentials, Properties) PUBLIC {

  // authorization code
  // optional code to specify user account properties and roles
}
```

where:

- *ServiceName* — A string specifying the name of the service through which the user is connecting to Caché, such as **%Service_Console** or **%Service_CSP**.
- *Namespace* — A string specifying the namespace on the Caché server to which a connection is being established. This is for use only with **%Service_Bindings**, such as connections for Studio or ODBC; for any other service, the value should be "" (the empty quoted string).
- *Username* — A string specifying the user whose privileges are being determined.
- *Password* — A string specifying the password associated with account in use. This is for use only with the Kerberos K5API authentication mechanism; for any other mechanism, the value should be "" (the empty quoted string).
- *Credentials* — *Passed by reference.* Not implemented in this version of Caché.

- *Properties — Passed by reference.* An array of returned values that specifies characteristics of the account named by *Username*. For more information, see "ZAUTHORIZE and User Properties."

17.2.3 Authorization Code with ZAUTHORIZE

The purpose of **ZAUTHORIZE** is to establish or update the roles and other characteristics for the authenticated user. The content of authorization code is application-specific. It can include any user-written ObjectScript code, class method, or $ZF callout.

ZAUTHORIZE specifies role information by setting the values of the *Properties* array, which is passed by reference to **ZAUTHORIZE**. Typically, the source for the values being set is a repository of user information that is available to **ZAUTHORIZE**.

CAUTION: Because Caché does not and cannot place any constraints on the authorization code in **ZAUTHORIZE**, the application programmer is responsible for ensuring that this code is sufficiently secure.

17.2.3.1 ZAUTHORIZE and User Properties

Elements of the *Properties* array specify values of attributes associated with the user specified by the *Username* parameter. Typically, code within **ZAUTHORIZE** sets values for these elements. The elements in the *Properties* array are:

- *Properties("Comment")* — Any text.
- *Properties("FullName")* — The first and last name of the user.
- *Properties("NameSpace")* — The default namespace for a Terminal login.
- *Properties("Roles")* — The comma-separated list of roles that the user holds in Caché.
- *Properties("Routine")* — The routine that is executed for a Terminal login. A value of "" specifies that the Terminal run in programmer mode.
- *Properties("Password")* — The user's password.
- *Properties("Username")* — The user's username.

Each of these elements is described in more detail in one of the following sections.

Note: It is not possible to manipulate the value of any member of the *Properties* array after authorization.

Comment

If **ZAUTHORIZE** returns a value for *Properties("Comment")*, then that string becomes the value of the user account's *Comment* property in Caché. (This property is described in the section "Properties of Users," in the "Users" chapter.) If no value is passed back to the calling routine, then the value of *Comment* for the user account is a null string and the relevant field in the Management Portal holds no content.

FullName

If **ZAUTHORIZE** returns a value for *Properties("FullName")*, then that string becomes the value of the user account's *Full name* property in Caché. (This property is described in the section "Properties of Users," in the "Users" chapter.) If no value is passed back to the calling routine, then the value of *Full name* for the user account is a null string and the relevant field in the Management Portal holds no content.

NameSpace

If **ZAUTHORIZE** sets the value of *Properties("Namespace")*, then that string becomes the value of the user account's *Startup Namespace* property in Caché. (This property is described in the section "Properties of Users," in the "Users"

chapter.) If no value is passed back to the calling routine, then the value of *Startup Namespace* for the user account is a null string and the relevant field in the Management Portal holds no content.

Once connected to Caché, the value of *Startup Namespace* — as specified by the value of *Properties("Namespace")* — determines the initial namespace for any user authenticated for local access (such as for Console, Terminal, or Telnet). If *Startup Namespace* has no value, then the initial namespace for any user authenticated for local access is determined as follows:

1. If the USER namespace exists, that is the initial namespace.
2. If the USER namespace does not exist, the initial namespace is the %SYS namespace.

Note: If the user does not have the appropriate privileges for the initial namespace, access is denied.

Password

If **ZAUTHORIZE** sets the value of *Properties("Password")*, then that string becomes the value of the user account's *Password* property in Caché. (This property is described in the section "Properties of Users," in the "Users" chapter.) If no value is passed back to the calling routine, then the value of *Password* for the user account is a null string and the relevant field in the Management Portal then holds no content.

If **ZAUTHORIZE** returns a password, this allows the user to log into the system via Password authentication if it is enabled. This is a possible mechanism to help migrate from delegated authentication to Password authentication, though with the usual cautions associated with the use of multiple authentication mechanisms; see "Cascading Authentication" in the Authentication chapter for more details.

Roles

If **ZAUTHORIZE** sets the value of *Properties("Roles")*, then that string specifies the *Roles* to which a user is assigned; this value is a string containing a comma-delimited list of roles. If no value is passed back to the calling routine, then there are no roles associated with the user account and the Management Portal indicates this. Information about a user's roles is available on the **Roles** tab of a user's **Edit User** page and a user's profile.

If any roles returned in *Properties("Roles")* are not defined, then the user is not assigned to the role.

Hence, the logged-in user is assigned to roles as follows:

- If a role is listed in *Properties("Roles")* and is defined by the Caché instance, then the user is assigned to the role.
- If a role is listed in *Properties("Roles")* and is not defined by the Caché instance, then the user is not assigned to the role.
- A user is always assigned to those roles associated with the _PUBLIC user. A user also has access to all public resources. For information on the _PUBLIC user, see the section "The _PUBLIC Account" in the "Users" chapter; for information on public resources, see the section "Services and Their Resources" in the "Resources" chapter.

Routine

If **ZAUTHORIZE** sets the value of *Properties("Routine")*, then that string becomes the value of the user account's *Startup Tag^Routine* property in Caché. (This property is described in the section "Properties of Users," in the "Users" chapter.) If no value is passed back to the calling routine, then the value of *Startup Tag^Routine* for the user account is a null string and the relevant field in the Management Portal then holds no content.

If *Properties("Routine")* has a value, then this value specifies the routine to execute automatically following login on a terminal-type service (such as Console, Terminal, or Telnet). If *Properties("Routine")* has no value or a value of "", then login starts the Terminal session in programmer mode, subject to whether they have the privilege to access programmer mode or not.

Username

If **ZAUTHORIZE** sets the value of *Properties("Username")*, then that string becomes the value of the user account's *Name* property in Caché. (This property is described in the section "Properties of Users", in the "Users" chapter.) This provides the application programmer with an opportunity to normalize content provided by the end-user at the login prompt (while ensuring that the normalized username only differ by case).

If there is no explicit call that passes the value of *Properties("Username")* back to the calling routine, then there is no normalization and the value entered by the end-user at the prompt serves as the value of the user account's *Name* property without any modification.

17.2.3.2 The User Information Repository

ZAUTHORIZE can refer to any kind of repository of user information, such as a global or an external file. It is up to the code in the routine to set any external properties in the *Properties* array so that the authenticated user can be created or updated with this information. For example, while a repository can include information such as roles and namespaces, **ZAUTHORIZE** code must make that information available to Caché.

If information in the repository changes, this information is only propagated back into the Caché user information if there is code in **ZAUTHORIZE** to perform this action. Also, if there is such code, changes to users' roles must occur in the repository; if you change a user's roles during a session, the change does not become effective until the next login, at which point the user's roles are reset by **ZAUTHORIZE**.

17.2.4 ZAUTHORIZE Return Value and Error Messages

The routine returns one of the following values:

- Success — **$SYSTEM.Status.OK()**. This indicates that **ZAUTHORIZE** has successfully executed. Depending on the code in the routine, this can indicate successful authentication of the user associated with *Username* and *Password*, successful authorization of the user associated *Username*, or both.
- Failure — **$SYSTEM.Status.Error($$$ERRORMESSAGE)**. This indicates that authorization failed. When **ZAUTHORIZE** returns an error message, it appears in the audit log if the LoginFailure event auditing is enabled; the end-user only sees the **$SYSTEM.Status.Error($$$AccessDenied)** error message.

ZAUTHORIZE can return system-defined or application-specific error messages. All these messages use the **Error** method of the %SYSTEM.Status class. This method is invoked as **$SYSTEM.Status.Error** and takes one or two arguments, depending on the error condition.

The available system-defined error messages are:

- **$SYSTEM.Status.Error($$$AccessDenied)** — Error message of "Access Denied"
- **$SYSTEM.Status.Error($$$InvalidUsernameOrPassword)** — Error message of "Invalid Username or Password"
- **$SYSTEM.Status.Error($$$UserNotAuthorizedOnSystem,Username)** — Error message of "User *username* is not authorized"
- **$SYSTEM.Status.Error($$$UserAccountIsDisabled,Username)** — Error message of "User *username* account is disabled"
- **$SYSTEM.Status.Error($$$UserInvalidUsernameOrPassword,Username)** — Error message of "User *username* invalid name or password"
- **$SYSTEM.Status.Error($$$UserLoginTimeout)** — Error message of "Login timeout"
- **$SYSTEM.Status.Error($$$UserCTRLC)** — Error message of "Login aborted"
- **$SYSTEM.Status.Error($$$UserDoesNotExist,Username)** — Error message of "User *username* does not exist"

- **$SYSTEM.Status.Error($$$UserInvalid,Username)** — Error message of "Username *username* is invalid"
- **$SYSTEM.Status.Error($$$PasswordChangeRequired)** — Error message of "Password change required"
- **$SYSTEM.Status.Error($$$UserAccountIsExpired,Username)** — Error message of "User *username* account has expired"
- **$SYSTEM.Status.Error($$$UserAccountIsInactive,Username)** — Error message of "User *username* account is inactive"
- **$SYSTEM.Status.Error($$$UserInvalidPassword)** — Error message of "Invalid password"
- **$SYSTEM.Status.Error($$$ServiceDisabled,ServiceName)** — Error message of "Logins for Service *username* are disabled"
- **$SYSTEM.Status.Error($$$ServiceLoginsDisabled)** — Error message of "Logins are disabled"
- **$SYSTEM.Status.Error($$$ServiceNotAuthorized,ServiceName)** — Error message of "User not authorized for service"

To use these error codes, uncomment the `#Include %occErrors` statement that appears in **ZAUTHORIZE.mac**.

To generate a custom message, use the **$SYSTEM.Status.Error()** method, passing it the `$$$GeneralError` macro and specifying any custom text as the second argument. For example:

```
$SYSTEM.Status.Error($$$GeneralError,"Any text here")
```

Note that when an error message is returned to the caller, it is logged in the audit database (if LoginFailure event auditing is turned on). However, the only error message the user sees is $SYSTEM.Status.Error($$$AccessDenied). However, the user also sees the message for the `$$$PasswordChangeRequired` error. Return this error if you want the user to change from the current to a new password.

17.3 Configuring an Instance to Use Delegated Authorization

Once you have created a customized **ZAUTHORIZE** routine, the next step is to enable it for the instance's relevant services or applications. This procedure is:

1. Run the **^SECURITY** routine from the %SYS namespace in a Terminal or Console window.
2. In **^SECURITY**, choose **System parameter setup**; under that, choose **Edit authentication options**; and under that, choose either **Allow Kerberos authentication** or **Allow Operating System authentication**. (Delegated authorization is only supported for these two authentication mechanisms.).
3. If you have selected **Allow Operating System authentication**, choose **Allow Delegated Authorization for O/S authentication**. If you have selected **Allow Kerberos authentication**, choose **Allow Delegated Authorization for Kerberos** authentication.

Selecting either of these choices causes Caché to invoke the **ZAUTHORIZE.mac** routine, if one exists, in the %SYS namespace.

Important: Caché only calls **ZAUTHORIZE** after user authentication.

17.3.1 Delegated Authorization and User Types

When a user first logs in to Caché with an authentication mechanism that uses delegated authorization, the system creates a user account either of Type OS (for Operating System) or Kerberos. (Note that this value does not appear in the Type column of the table of users on the **Users** page (**System Administration** > **Security** > **Users**).) At the time of account creation and, for subsequent logins, the **ZAUTHORIZE** routine specifies the roles for the user.

Any attempt to log in without using delegated authorization will result in a login failure. This is because only delegated authorization specifies the user Type as OS or Kerberos. When using these authentication mechanisms without delegated authorization, the user is authenticated as being of Type "Caché Password User"; the login fails because a user can only have one type and a user of one type cannot log in using mechanisms associated with another type. (Delegated authentication and LDAP authentication also both fail for the same reason.)

For general information about user types, see the section "About User Types" in the "Users" chapter.

17.4 After Authorization — The State of the System

If the user is successfully authorized, the Cache security database is updated in one of the following ways:

1. If this is the first time the user has logged in, a user record is created in the security database for the entered username, using properties returned by **ZAUTHORIZE**.
2. If the user has logged in before, the user record is updated in the security database, using properties returned by this function.

Whether for a first-time user or not, the process that logs in has the value of the *$ROLES* system variable set to the value of Properties("Roles"). For a terminal login, the namespace is set to the value of Properties("NameSpace") and the startup routine is set to the value of Properties("Routine").

A

Tightening Security for a Caché Instance

To provide increased security for a Caché instance, you can configure it to more tightly constrain user access. This can prevent unauthorized users from using Caché tools or from gaining access to sensitive resources. This appendix describes various actions that reduce the attack surface of a Caché instance or otherwise increase its security.

There are multiple actions for tightening an instance's security. They are presented here roughly in the sequence in which they should be performed:

- Enabling auditing
- Changing the authentication mechanism for an application
- Limiting the number of public resources
- Restricting access to services. This involves:
  - Limiting the number of enabled services
  - Limiting the number of public services
  - Restricting access to services by IP address or machine name
- Restricting public privileges
- Limiting the number of privileged users
- Disabling the _SYSTEM user
- Restricting access for UnknownUser
- Configuring third-party software

The Caché Security Advisor also provides an automated analysis of the instance and recommendations for actions to increase the security of an instance.

Important: A Caché instance has many interdependent elements. Because of this, it is recommended that you only do what is specified for a change, and not more or less. For example, simply removing UnknownUser from the **%All** role — without doing anything else — will cause problems for a minimal-security installation.

A.1 Enabling Auditing

The primary elements of security are often described as the "Three A's": authentication, authorization, and auditing. Auditing, including the auditing provided with Caché, provides two functions:

- It provides data about what has occurred if there is a security event.
- The knowledge of its existence can be a deterrent for an attacker, given that the attack will be tracked and there will be evidence of any malicious actions.

To enable auditing for key events for a Caché instance, the procedure is:

1. From the Management Portal home page, select **System Administration** > **Security** > **Auditing** > **Enable Auditing**. If the choice is not available, auditing is already enabled.
2. From the Management Portal home page, go to **Configure System Events** page (**System Administration** > **Security** > **Auditing** > **Configure System Events**).
3. On the **Configure System Events** page, enable the following events if they are not already enabled by clicking **Change Status** in the event's row:
   - %System/%DirectMode/DirectMode — Provides information on console/terminal use. For sites that extensively use command-line utilities, can create large amounts of data. Recommended if increased data is not an issue.
   - %System/%Login/Login — Provides information on logins. For large sites, can create large amounts of data. Recommended if increased data is not an issue.
   - %System/%Login/LoginFailure — Provides feedback on possible attempted unauthorized logins. Recommended.
   - %System/%Security/Protect — Provides data on attempts to read, write, or use protected data. Recommended.

A.2 Changing the Authentication Mechanism for an Application

A key element of restricting access to Caché is configuring the instance to use a stricter authentication mechanism for its applications. This section describes how to perform this procedure, using the Management Portal as an example application and with the change from unauthenticated access (as in a minimal-security installation) to requiring a password as an example of moving to a stricter authentication mechanism.

Important: Performing the following procedure may affect aspects of the instance being modified beyond access to the Portal. The specifics depend on (1) the instance's configuration and (2) whether you are performing just this procedure or all the procedures in this appendix. Specifically:

- Making %Service_CSP:Use not public means that all users of web applications will need to be granted **`%Service_CSP:Use`** by some other means.
- Removing UnknownUser from the %All role can have many effects.

To provide properly functioning authentication for an application, there must be consistent authentication mechanisms for both the application and any service that it uses. For a web application, the CSP Gateway must also be configured to match the CSP service. Hence, to provide authentication for the Management Portal, there are three layers that all need to work together:

- The **`%Service_CSP`** service
- The CSP Gateway
- The Management Portal application

If these layers do not have matching authentication mechanisms, this usually results in a denial of access — for example, there may be a “This page cannot be displayed” error instead of a login page or access to the Management Portal.

Important: If (1) a web application uses a more powerful authentication mechanism than the CSP Gateway and **%Service_CSP** and (2) authentication succeeds, then the system’s security is only that of the less powerful mechanism.

For an instance with a minimal-security installation, the CSP Gateway, **%Service_CSP**, and the Management Portal application are all set up for unauthenticated access. To provide password-level authentication for the Portal, various Caché elements must be configured as follows:

- The CSP service must require password authentication.
- The CSP Gateway must provide a username and password for that authentication.
- The user representing the Gateway must have sufficient privilege to use the CSP service.
- The Management Portal must require password authentication.
- All the Portal’s users must have sufficient privilege to use the Portal.

Important: Complete the following set of procedures during a single session in the Portal. Otherwise, you may lock yourself out of the Portal and have to perform the remaining procedures through the ^SECURITY routine.

An overview of the procedure to make these changes is:

1. Optionally, turn on auditing to track the changes to the instance. This is described in the Enabling Auditing section of this appendix.
2. Give the %Service_CSP:Use privilege to the CSPSystem user.
3. Change the password of the CSPSystem user.
4. Configure the CSP Gateway to provide a username and password for authentication.
5. Configure %Service_CSP to require password authentication.
6. Remove the public status of the %Service_CSP:Use privilege.
7. Configure the Management Portal application to require password authentication only.
8. Specifying the appropriate privilege level for the instance’s users.
9. Optionally, make the documentation or samples available.
10. Begin enforcement of the new policies.

Once this process is complete, then a user’s next attempt to connect to the Portal will result in a login prompt.

CAUTION: A Caché instance has many interdependent elements. Because of this, it is recommended that you only do what is specified for a change, and not more or less. Otherwise, you may lock yourself out of Caché or could even render the instance temporarily inoperative.

A.2.1 Giving the %Service_CSP:Use Privilege to the CSPSystem User

The Caché installation process creates a CSPSystem user, which represents the CSP Gateway in its interactions with the **%Service_CSP** service. Since the service is going to have restricted access, this user needs to hold the **%Service_CSP:Use** privilege for the authentication process.

Note: There is a *service* called **%Service_CSP** and a *resource* called **%Service_CSP**. The resource regulates access to the service. Therefore, to gain access to the service, a user must have Use permission for the resource — that is, the **%Service_CSP:Use** privilege.

To associate the **%Service_CSP:Use** privilege with the CSPSystem user, the procedure is:

1. From the Management Portal home page, go to the **Roles** page (**System Administration** > **Security** > **Roles**).
2. On the **Roles** page, click **Create New Role**. This displays the **Edit Role** page, where the **Name** field is editable.
3. Enter a name for the role to include the **%Service_CSP:Use** privilege (such as "GatewayRole").
4. Click **Save**. Caché has now created the role.
5. In the **Privileges** section on the **General** tab of the **Edit Role** page, click **Add**, which displays a list of available resources for the role.
6. From this list, click **%Service_CSP** and then click **Save**. The newly created role now includes the **%Service_CSP:Use** privilege.
7. Select the **Members** tab of the **Edit Role** page.
8. On this tab, you can assign the CSPSystem user to the newly created role. Click CSPSystem from the users in the **Available** list and move it to the **Selected** by clicking the right arrow.
9. Click **Assign** to assign CSPSystem to the role. (In other words, CSPSystem is now a member of the role.) This means that CSPSystem holds the **%Service_CSP:Use** privilege.

Note: The system creates the CSPSystem user to represent the CSP Gateway. If you prefer, a different user can perform this function. This procedure refers only to the CSPSystem user; if you use a different user, replace CSPSystem with that username where relevant.

A.2.2 Changing the Password of the CSPSystem User

Because a minimal-security installation gives the CSPSystem user a password of "SYS", it is important to change this to a new password — one that an attacker would not know or be able to guess. The procedure is:

1. In the Management Portal, go to the **Users** page (**System Administration** > **Security** > **Users**).
2. On the **Users** page, in the row for the CSPSystem user, select the name **CSP Gateway User**. This displays the **Edit User** page.
3. Enter the new password for CSPSystem in the **Password** field. Since no user has to remember this password, you can make it as long and complex as you wish.
4. Reenter the new password in the **Confirm Password** field and click **Save**. If the Portal does not display an error message or dialog, then the password change has succeeded.

If you wish, you can also confirm that CSPSystem is assigned to the role created for authentication in the previous procedure. To do this, click on the **Roles** tab. The table with the column heading **CSPSystem is Assigned to the Following Roles** should list the newly-created role.

A.2.3 Configuring the CSP Gateway to Provide a Username and Password

Because you are going to configure **%Service_CSP** to require password authentication, the CSP Gateway needs to provide a username-password pair. Having set up a user with the appropriate level of privilege, you have established a username-password pair that the Gateway can provide. The next step is to configure the Gateway to provide this username-password pair when the Caché server challenges it for them. The procedure is:

1. In the Management Portal, go to the **Caché Server Pages - Web Gateway Management** page (**System Administration** > **Configuration** > **CSP Gateway Management**).
2. On the **Caché Server Pages - Web Gateway Management** page, select **Server Access** from the list on the left side. This displays the **Server Access** frame.
3. In the **Server Access** frame, the LOCAL server should be highlighted. Click **Submit** to edit it, which displays a page with Server Access and Error Pages parameters.
4. On this page, there is a **Connection Security** section.
5. Ensure that the **Connection Security Level** drop-down has "Password" displayed.
6. In the **User Name** field, enter `CSPSystem`.
7. In the **Password** and **Password (Confirm)** field, enter the password that you selected in the previous section.
8. Click **Save Configuration** near the bottom of the page.
9. To return to the Management Portal, click **Back to Management Portal** from the bottom of the list in the left pane.

A.2.4 Configuring %Service_CSP to Require Password Authentication

Now that the Gateway is configured to provide a username and password and you have given the CSPSystem user the necessary level of privilege, the next step is to configure the service that manages CSP (**`%Service_CSP`**) so that it requires password authentication. The procedure is:

1. From the Management Portal home page, go to the **Services** page (**System Administration** > **Security** > **Services**).
2. On the **Services** page, click **%Service_CSP**. This displays the **Edit Service** page for **`%Service_CSP`**.
3. On the **Edit Service** page, under **Allowed Authentication Methods**, make sure that **Unauthenticated** access is disabled and that **Password** access is enabled (also known as "Caché login"). Click **Save**.

A.2.5 Removing the Public Status of the %Service_CSP:Use Privilege

With **`%Service_CSP`** requiring password authentication and the Gateway able to authenticate with an appropriately authorized user, the next step is to exclude **`%Service_CSP:Use`** from public availability. The procedure is:

1. From the Management Portal home page, go to the **Resources** page (**System Administration** > **Security** > **Resources**).
2. On the **Resources** page, in the row for **`%Service_CSP`**, click **Edit**. This displays the **Edit Resource** page for **`%Service_CSP`**.
3. In the **Public Permission** section, clear the **Use** box. Click **Save**.

Important: Once **`%Service_CSP:Use`** is not a public privilege, only those users who have been explicitly granted it will be able to use CSP. You may need to assemble a list of these users and grant them this privilege through other means.

A.2.6 Configuring the Management Portal to Accept Password Authentication Only

Once the connection between the Gateway and the Caché server has a new authentication mechanism, the next task is to configure the Management Portal application to use a matching mechanism. In this example, this mechanism is Caché login. The procedure for changing the Portal's authentication mechanism is:

1. From the Management Portal home page, go to the **Web Applications** page (**System Administration** > **Security** > **Applications** > **Web Applications**).
2. On the **Web Applications** page, the /csp/sys application represents the Management Portal home page. Select the name **/csp/sys** in this row to edit the application. This displays the **Edit Web Application** page for the /csp/sys application.
3. Under **Allowed Authentication Methods**, disable **Unauthenticated** access and enable **Password** access. Click **Save**.
4. Also disable **Unauthenticated** access and enable **Password** access for all the applications that compose the other pages and choices of the Portal. These applications are:
   - /csp/sys/bi
   - /csp/sys/exp
   - /csp/sys/mgr
   - /csp/sys/op
   - /csp/sys/sec

This configures the Portal to require password authentication (also known as "Caché login") and not to allow unauthenticated access, and so that all its parts behave consistently. The next step is to ensure that all relevant users have appropriate access to the Portal.

A.2.7 Specifying the Appropriate Privilege Level for the Instance's Users

When the Portal is configured to accept unauthenticated connections, any user can connect as the UnknownUser. Because a minimal-security installation makes UnknownUser a member of the **%All** role, there is no danger of being locked out of the Portal. Now that the Portal requires password authentication, its legitimate users need to be members of the **%Operator** role, the **%Manager** role, or the **%All** role.

In a minimal-security installation, SuperUser, Admin, _SYSTEM, and UnknownUser all have this level of privilege; further, these all have passwords of "SYS".

To properly secure users, the procedure is:

1. Either disable UnknownUser or remove UnknownUser from the **%All** role.
   - To disable UnknownUser, the procedure is:
     a. On the Users page (**System Administration** > **Security** > **Users**), click **UnknownUser** under the **Name** column. This displays the **Edit User** page for UnknownUser.
     b. Clear the **User Enabled** field and click **Save**.
   - To remove UnknownUser from the **%All** role:
     a. On the **Users** page (**System Administration** > **Security** > **Users**), click **UnknownUser** under the **Name** column. This displays the **Edit User** page for UnknownUser.
     b. Go to the **Roles** tab on the **Edit User** page.
     c. In the **UnknownUser is Assigned to the Following Roles** table, on the **%All** row, click **Remove**, then click **Save**.

Important: Limiting access through UnknownUser can have widespread effects, particularly if an instance's users are not sufficiently privileged.

2. Ensure that any other potentially unauthorized users are not members of **%All**, **%Developer**, **%Manager**, **%Operator**, **%SQL**, or any user-defined role that grants privileges. This involves a process analogous to removing UnknownUser from the **%All** role.

   (A user-defined role that grants privileges might have Use permission on any of the **%Admin...** resources, **%Development**, or any of the **%Service** or **%System** resources, or Write permission on **%DB_CACHELIB** or **%DB_CACHESYS**.)

3. Ensure that any user who *should* have access to the Portal is assigned to **%All**, **%Developer**, **%Manager**, **%Operator**, **%SQL**, or any user-defined role that grants Portal access. The procedure, for each of these users, is:

   a. On the **Users** page (**System Administration** > **Security** > **Users**), click the name of the user under the **Name** column. This displays the **Edit User** page for that user.

   b. Go to the **Roles** tab on the **Edit User** page.

   c. Move the desired role(s) from the **Available** to the **Selected** list by selecting the role, clicking the right arrow button, and then clicking **Assign** to assign the user to the role(s).

4. Change the passwords for SuperUser and Admin users from the default. To do this:

   a. On the **Users** page (**System Administration** > **Security** > **Users**), click the name of the user under the **Name** column. This displays the **Edit User** page for that user.

   b. Select **Enter new password**.

   c. Enter the new password in the **Password** field.

   d. Confirm it in the **Password (confirm)** field and click **Save**.

Important: Make sure that you know the password for at least one user who administers the Portal. Otherwise, you may lock yourself out of the Portal and have to log in using emergency access so that you can reset one or more passwords using the **^SECURITY** routine.

A.2.8 Making the Documentation or Samples Available

Once you finish configuring the service, the CSP Gateway, and the Portal application, you may wish to ensure that the documentation or sample programs are available. The procedure is:

1. From the Management Portal home page, go to the **Web Applications** page (**System Administration** > **Security** > **Applications** > **Web Applications**).

2. To make the sample applications available:

   a. On the **Web Applications** page, the /csp/samples application represents the Caché sample applications. Select **/csp/samples** in this row to edit the application. This displays the **Edit Web Application** page for the /csp/samples application.

   b. Under **Allowed Authentication Methods**, disable **Unauthenticated** access and enable **Password** access. Click **Save**.

3. To make the documentation available:

   a. On the **Web Applications** page, the /csp/docbook application represents the Caché DocBook documentation application. Select the name **/csp/docbook** to edit the application. This displays the **Edit Web Application** page for the /csp/docbook application.

   b. Under **Allowed Authentication Methods**, disable **Unauthenticated** access and enable **Password** access. Click **Save**.

   c. Return to the **Web Applications** page.

d. On the **Web Applications** page, the /csp/documatic application represents the Caché Documentation class reference application. Select **/csp/documatic** in this row to edit the application. This displays the **Edit Web Application** page for the /csp/documatic application.

e. Under **Allowed Authentication Methods**, disable **Unauthenticated** access and enable **Password** access. Click **Save**.

Note: Because the documentation is compose of two individual applications — /csp/docbook and /csp/documatic — that are separate from each other and from the Portal, each has a separate password prompt.

If you do not perform this procedure, the service requires a password prompt but the application attempts to use unauthenticated access. This prevents all users — including those assigned to **%All** — from reaching the documentation or samples.

A.2.9 Beginning Enforcement of New Policies

At this point, the Caché instance is fully configured to operate properly. However, all existing connections are still using unauthenticated access. To begin enforcement of the new policies, the following events must occur:

- The CSP Gateway must establish an authenticated connection.
- All users must also establish authenticated connections.

A.2.9.1 Establishing an Authenticated CSP Gateway Connection

To force the CSP Gateway to establish an authenticated connection, the procedure is:

1. From the Management Portal home page, select **System Administration** > **Configuration** > **CSP Gateway Management**. This displays the **Caché Server Pages - Web Gateway Management** page.
2. On the **Caché Server Pages - Web Gateway Management** page, select **Close Connections** from the list on the left side. This displays the **Close Connections** frame.
3. Click **Close Connection(s)**. This displays a message indicating that all connections between the Gateway and Caché server have been closed.

The next time that a user requests a page, the Gateway will reestablish a connection to the Caché server. This connection will use the selected authentication mechanism.

A.2.9.2 Establishing Authenticated User Connections

At this point, all connections to the Management Portal are still using unauthenticated access. If there is no pressing need to require authenticated access, then there is nothing else to do. Users will gradually end their connections to the Portal and will have to authenticate when they reconnect. (Connections may be ended due to machine reboots, stopping and restarting browsers, clearing browser caches, Portal logouts, etc.)

If there is a need to force connections to use authenticated access, you can stop and restart Caché. For example, on Windows, if you have Caché available through the default Start menu page:

1. From the Windows Start menu, select **Programs** > **Caché**, then the Caché instance to restart.
2. On the submenu for the instance of Caché, choose **Stop Caché**.
3. On the dialog that appears, select **Restart** and click **OK**.

Note: If you are using a production instance of Caché, you may want to choose a low-traffic time for the restart, since users will temporarily not have access to either Caché as a whole or the Portal.

A.3 Limiting the Number of Public Resources

Any resource can be specified as a public resource. This means that any user has the ability to read, write, or use the resource, depending on its public settings. The following should always be public:

Table I–1: Required Public Resources and Their Permissions

Resource	Permission
%DB_CACHE	R
%DB_CACHELIB	R
%DB_CACHETEMP	RW

Note: You may also want to make the Read permission on the **%DB_DOCBOOK** database public, so that all users have access to the documentation.

To tighten the security of an instance, limit the number of public resources. To do this, the procedure is:

1. Ensure that all users who genuinely require access to these resources have been given privileges for them.

   Important: If you do not provide privileges for **%Service_CSP:Use** to the appropriate users, then this procedure can result in a widespread lockout from the Management Portal and other CSP applications.

2. From the Management Portal home page, go to the **Resources** page (**System Administration** > **Security** > **Resources**).
3. On the **Resources** page, each resource for which there is one or more public permissions has those permissions listed in the **Public Permissions** column of the table of resources. Select the resource by clicking **Edit**. This displays the resource's **Edit Resource** page.
4. On the **Edit Resource** page, clear any checked **Public Permission** fields and click **Save**. The resource is no longer public.

Perform this procedure for all public resources.

A.4 Restricting Access to Services

There are various pathways by which users can interact with Caché. Services regulate access to these pathways. To limit access to Caché services, the available choices are:

- Limiting the number of enabled services to only those required for the applications in use
- Limiting the number of public services to only those required for the applications in use
- Restricting access to services by IP address or machine name

A.4.1 Limiting the Number of Enabled Services

To limit the number of enabled services, the procedure is:

1. Determine the required services for the Caché instance. Typically, these are:
   - Whatever service is required for each form of user access

- Whatever services are required for any automated access
- Either **%Service_Console** (on Windows) or **%Service_Terminal** (on UNIX®), for local programmer-mode access

2. From the Management Portal home page, go to the **Services** page (**System Administration** > **Security** > **Services**).
3. On the **Services** page, for each service that is not required, select the service by clicking on its name. This displays the service's **Edit Service** page.
4. On the **Edit Service** page, clear the **Service Enabled** field and click **Save**. The service is now disabled.

Once you have disabled all unnecessary services, the only pathways to Caché are the required services.

A.4.2 Limiting the Number of Public Services

Each service is associated with a resource. In most cases, the resource has the same name as the service, such as **%Service_CSP**; the exception to this is the **%Service_Bindings** service, which is associated with the **%Service_Object** and **%Service_SQL** resources. Services are public because of the settings for the resources associated with them. Because of this, the procedure for making a service non-public is the same as for making any other resource non-public. This is described in the section, "Limiting the Number of Public Resources."

A.4.3 Restricting Access to Services by IP Address or Machine Name

For certain services, you have the option of restricting access to the service according to IP address or machine name. This is known as the ability to limit "allowed incoming connections." The services that support this feature are:

- **%Service_Bindings**
- **%Service_CSP**
- **%Service_CacheDirect**
- **%Service_ECP**
- **%Service_Monitor**
- **%Service_Shadow**
- **%Service_Weblink**

By default, a service accepts connections from all machines. If a service has no associated addresses or machine names, then it accepts connections from any machine. If one or more addresses or machine names are specified from which a service accepts connections, then the service only accepts connections from these machines.

This feature is not available for **%Service_CallIn**, **%Service_ComPort**, **%Service_Console**, **%Service_DataCheck**, **%Service_Login**, **%Service_MSMActivate**, **%Service_Mirror**, **%Service_Telnet**, and **%Service_Terminal**.

To restrict access to a service by IP address, the procedure is:

1. Determine the IP addresses of those machines with legitimate access to the service.
2. From the Management Portal home page, go to the **Services** page (**System Administration** > **Security** > **Services**).
3. On the **Services** page, for each service for which you are restricting access by IP address, select the service by clicking on its name. This displays the service's **Edit Service** page.
4. On the **Edit Service** page, in the Allowed Incoming Connections section, click **Add New**.
5. In the displayed dialog, enter an IP address for an allowed incoming connection. Click **OK**.

6. Click **Add** and enter other addresses as needed.

Perform this procedure for each service that is restricting the IP addresses from which it accepts connections.

A.5 Restricting Public Privileges

Whenever any user logs in, that user receives all the privileges associated with the _PUBLIC user. With a minimal-security installation, the _PUBLIC user is not assigned to any roles, but holds a number of SQL privileges. These have been granted by the _SYSTEM user, which is created by the Caché installation in accordance with the SQL standard as the SQL root user. This means that they must be revoked by the _SYSTEM user.

To revoke SQL privileges from the _PUBLIC user, the procedure is:

1. Log into the Management Portal as the _SYSTEM user:
   a. Enable the _SYSTEM user if that user is not already enabled. (After completing this procedure, you may wish to disable the _SYSTEM user according to the instructions in the section "Limiting the Number of Privileged Users.")
   b. For a minimal-security installation, change the authentication method for the Management Portal so that it requires logins. This is described in the section "Changing Authentication Methods for the Management Portal."
   c. Log in as _SYSTEM. In a minimal-security installation, the default password for this user is "SYS"; in normal and locked-down installations, the default password is whatever was selected during the installation process.
2. From the Management Portal home page, go to the **Execute SQL Query** page (**System Explorer** > **SQL** > **Execute SQL Query**) for the SAMPLES namespace. (On the **Execute SQL Query** page, from the list of namespaces on the left, click SAMPLES.)
3. Revoke all the _PUBLIC user's privileges on tables in the SAMPLES database. In the page's editing form, enter the following SQL command:

   ```
   REVOKE ALL PRIVILEGES ON * FROM _PUBLIC
   ```

   and click **Execute Query**.
4. Revoke all the _PUBLIC user's privileges to invoke stored procedures in the SAMPLES database. In the page's editing form, enter the following SQL command:

   ```
   REVOKE EXECUTE ON * FROM _PUBLIC
   ```

   and click **Execute Query**.
5. To prevent general use of the InterSystems documentation, you can revoke the _PUBLIC user's privileges to search the DocBook database. To do this:
   a. On the **Execute SQL Query** page, click DOCBOOK from the list of namespaces.
   b. Select the **SQL Tables** tab.
   c. In the page's editing form, enter by the following SQL command:

      ```
      REVOKE ALL PRIVILEGES ON * FROM _PUBLIC
      ```

      and click **Execute Query**.

A.6 Limiting the Number of Privileged Users

Every instance of Caché must have at least one user who is assigned to the **%All** role. In fact, if there is only one user assigned to this role, then Caché prevents the user from being removed from the role. However, over time, an instance can end up having more users assigned to **%All** than are necessary. This can arise from assigned users leaving an organization but their accounts not being disabled, from temporary assignments not being revoked, and so on.

Along with the **%All** role, the system-defined roles of **%Manager**, **%Developer**, **%Operator**, and **%SQL** can give users undue privilege. There also may be user-defined roles that do this. Users assigned to such roles are sometimes known as "privileged users."

To limit the number of privileged users, determine which users are assigned to each privileged role and remove those who are not needed. The procedure is:

1. From the Management Portal home page, go to the **Roles** page (**System Administration** > **Security** > **Roles**).
2. On the **Roles** page, click the name of the role. This displays the **Edit Role** page for that role.
3. On the **Edit Role** page, click the **Members** tab, which displays a list of the users and roles assigned to the role.
4. To remove any user from the specified role, click **Remove** on the row for the user or role to be removed. Click **OK** in the confirmation dialog.

Perform this procedure for each privileged role, including **%All** and the others listed previously. It is also important to disable the _SYSTEM user; the procedure for this is described in the next section, "Disabling the _SYSTEM User."

Important: Certain seemingly non-privileged roles may have what could be called "privileges by proxy." This occurs when a seemingly non-privileged role is assigned to a privileged role. In this case, any user who is assigned to role with privileges by proxy holds all the privileges associated with the privileged role.

Avoid creating privileges by proxy whenever possible. When not possible, have as few users as possible assigned to the roles with privileges by proxy.

A.7 Disabling the _SYSTEM User

The Caché installation program creates the _SYSTEM user. This user is created in accordance with the SQL standard as the SQL root user. In a minimal-security installation, the default password for this user is "SYS"; in normal and locked-down installations, the default password is whatever was selected during the installation process. Because this user and the password of "SYS" are both publicly specified by the SQL standard, and because of this user's SQL privileges, disabling _SYSTEM is important for tightening access to a Caché instance.

To do this, the procedure is:

1. From the Management Portal home page, go to the **Users** page (**System Administration** > **Security** > **Users**)).
2. On the **Users** page, click the name **_SYSTEM** to open the **Edit User** page for _SYSTEM.
3. On the **Edit User** page for _SYSTEM, clear the **User Enabled** field. Click **Save**.

Note: If you need to check root-level SQL privileges after disabling _SYSTEM, you will have to temporarily enable the user to perform the required action.

A.8 Restricting Access for UnknownUser

In instances that support unauthenticated access, connections that do not use authentication are established with the UnknownUser account. In minimal-security installations, the default behavior is that:

- All connections use UnknownUser.
- UnknownUser is assigned to the **%All** role.
- UnknownUser holds all SQL privileges.

To restrict access for UnknownUser, disable unauthenticated access for all enabled services. (Other actions may not be effective or may result in a lockout from the Management Portal.)

A.8.1 Potential Lockout Issue with the UnknownUser Account

If an instance has been installed with Minimal security, UnknownUser has the **%All** role; the instance also has unauthenticated access available for all services and applications. If you simply change the user's role (from **%All** to something else) and still allow unauthenticated access, you may be unable to use basic features.

This is because, under these conditions, Caché establishes a connection to the selected tool without performing authentication. When there is no authentication, the system automatically sets the user account to UnknownUser. The next step is to check user privileges. If UnknownUser has insufficient privileges, access to the tool is limited or not available. For example, under these circumstances, the Terminal displays an "Access Denied" message and then shuts down; the Portal displays its main page, but no options can be selected.

To correct this condition:

1. Start Caché in emergency access mode.
2. Restore sufficient privileges to the UnknownUser account.

If you wish to prevent use of UnknownUser, you must upgrade the authentication mechanism for the Management Portal.

A.9 Configuring Third-Party Software

InterSystems products often run alongside and interact with non-InterSystems tools, including virus scanners. For important information about the effects these interactions can have, see the appendix "Configuring Third-Party Software to Work in Conjunction with InterSystems Products" in the *Caché System Administration Guide*.

B

Using the cvencrypt Utility

Caché allows you to use encrypted databases, as described in the chapter "Managed Key Encryption." There are occasions when you may need to perform encryption management operations on a database, such as:

- Converting an Unencrypted Database to Be Encrypted
- Converting an Encrypted Database to Be Unencrypted
- Converting an Encrypted Database to Use a New Key
- Using Command-line Options with cvencrypt

To perform these operations, Caché supplies cvencrypt, a standalone utility for use with 8-KB format databases. Its available operations are described in the following sections.

Note: cvencrypt is not for use with journal files that belong to a running system.

B.1 Converting an Unencrypted Database to be Encrypted

This describes the procedure for making an unencrypted database encrypted.

1. Back up the data in the database to be encrypted.

   The cvencrypt utility encrypts data in place. This means that it uses on-disk space for its operations (not copying the database elsewhere and restoring it to its current disk location after successful completion). If the utility is interrupted before completion, the database will be partly encrypted and partly unencrypted, rendering it unusable.

   CAUTION: It is critical that you back up the database before encrypting it. Failure to do so can result in data being lost.

2. Have an existing database-encryption key for your Caché instance.
3. If the database is mounted, dismount it. You can do this from the Management Portal **Databases** page (**System Operation** > **Databases**).
4. In the directory containing the CACHE.DAT file for the database to encrypt, run the cvencrypt utility, providing a path to it. For example, if your database is in the C:\MyDBs directory and is called test1 and Caché is installed in the C:\InterSystems\MyCache directory, then you would run the utility as follows:

   ```
   C:\MyDBs\test1>..\..\InterSystems\MyCache\bin\cvencrypt CACHE.DAT
   ```

The utility then provides an informational display message about itself, and then information on the database passed to it.

5. The utility then prompts for what action you wish to take:

```
1) Encrypt
2) Quit
```

To encrypt the database, enter "1" and press **Enter**.

6. This displays a prompt to activate an already-existing encryption key:

```
Access database encryption key.
Keyfile:
```

At this prompt, enter the full path of the key for encrypting the database. For example, suppose that you have stored the database-encryption keyfile, dek1, in the C:\InterSystems\MyCache\Mgr directory.

```
Keyfile: C:\InterSystems\MyCache\Mgr\dek1
```

7. The utility then prompts for the username and password of a keyfile administrator:

```
Username: dek-admin1
Password: <characters shielded during entry>
```

When it receives these, the utility then reminds you that it modifies data in place, so that you should be sure that your data is adequately backed up before continuing:

```
This utility will modify your database in place.
Be sure that your data is adequately backed up before proceeding.
Continue? [Y/N]:
```

If you wish to continue, enter "Y".

it encrypts the database, stating the key that it is using for encryption:

```
Using database encryption key (ID = 5DBC532D-4D1F-A3A4-30CDA34BB66B).
```

During encryption, the utility displays a progress indicator, as well as a reminder not to interrupt the process.

CAUTION: If you interrupt the process while it is incomplete, the database may be in a state in which some of its data is encrypted and some of it is unencrypted. It is impossible for Caché to read such a database, and *all the data will be lost*.

8. When finished, the utility announces this, such as:

```
Processed:
118400 blocks (done!)
```

B.2 Converting an Encrypted Database to be Unencrypted

This describes the procedure for making an encrypted database unencrypted.

1. Back up the database to be unencrypted.

The cvencrypt utility unencrypts data in place. This means that it uses on-disk space for its operations (not copying the database elsewhere and restoring it to its current disk location after successful completion). If the utility is interrupted before completion, the database will be partly encrypted and partly unencrypted, rendering it unusable.

CAUTION: It is critical that you back up the database before decrypting it. Failure to do so can result in data being lost.

2. If the database is mounted, dismount it. You can do this from the Management Portal **Databases** page (**System Operation** > **Databases**).
3. In the directory containing the CACHE.DAT file for the database to decrypt, run the cvencrypt utility, providing a path to it. For example, if your database is in the C:\MyDBs directory and is called test1 and Caché is installed in the C:\InterSystems\MyCache directory, then you would run the utility as follows:

   ```
   C:\MyDBs\test1>..\..\InterSystems\MyCache\bin\cvencrypt CACHE.DAT
   ```

   The utility then provides an informational display message about itself, and then information on the database passed to it.
4. If the database is encrypted, the utility displays information about this:

   ```
   Database is encrypted
   (Key ID = E1D00BC2-07F4-4495-9562-394B26A2B05B).
   ```

   It then prompts for an action:

   ```
   1) Decrypt
   2) Re-encrypt with a different key
   3) Quit
   ```

5. To decrypt the database, enter "1" and press **Enter**. This displays a prompt to activate the database's encryption key:

   ```
   Access original database encryption key
   (key ID = E1D00BC2-07F4-4495-9562-394B26A2B05B)
   Keyfile:
   ```

   Enter the keyfile including its full path, such as C:\encdb\dek; relative pathnames are not valid.
6. The utility then prompts for the username and password of a keyfile administrator:

   ```
   Username: dek-admin1
   Password: <characters shielded during entry>
   ```

7. The utility next states that it is ready to perform the decryption:

   ```
   Prepared to decrypt database
   (key ID = E1D00BC2-07F4-4495-9562-394B26A2B05B).
   ```

   Before decrypting the data, the utility reminds you that it modifies data in place, so that you should be sure that your data is adequately backed up before continuing:

   ```
   This utility will modify your database in place.
   Be sure that your data is adequately backed up before proceeding.
   Continue? [Y/N]:
   ```

   If you wish to continue, enter "Y". Once you confirm your decision, the utility decrypts the database.

   During decryption, the utility displays a progress indicator, as well as a reminder not to interrupt the process, such as:

```
Decrypting database
(key ID = E1D00BC2-07F4-4495-9562-394B26A2B05B).
Do not interrupt this process!
Processed:
25000 blocks (16%)
```

CAUTION: If you interrupt the process while it is incomplete, the database may be in a state in which some of its data is encrypted and some of it is unencrypted. It is impossible for Caché to read such a database, and *all the data will be lost*.

8. When the process is complete, the progress indicator changes to a completion message:

```
Processed:
153600 blocks (done!)
```

B.3 Converting an Encrypted Database to Use a New Key

This describes the procedure for re-encrypting an encrypted database using a new key.

1. Back up the data in the database to be re-encrypted.

   The cvencrypt utility re-encrypts data in place, that is, using the on-disk space for its operations (not copying the database elsewhere and restoring it to its current disk location after successfully completing the decryption). If the utility is interrupted before completion, the database will be partly encrypted with two different keys, rendering it unusable.

   CAUTION: It is critical that you back up the database before decrypting it. Failure to do so can result in data being lost.

2. If the database is mounted, dismount it. You can do this from the Management Portal **Databases** page (**System Operation** > **Databases**).

3. Make sure that the key with which the database is being re-encrypted already exists. You can create this key using the Management Portal **Database Encryption** page (**System Administration** > **Encryption** > **Database Encryption**).

4. In the directory containing the CACHE.DAT file for the database to decrypt, run the cvencrypt utility, providing a path to it. For example, if your database is in the C:\MyDBs directory and is called test1 and Caché is installed in the C:\InterSystems\MyCache directory, then you would run the utility as follows:

   ```
   C:\MyDBs\test1>..\..\InterSystems\MyCache\bin\cvencrypt CACHE.DAT
   ```

   When it first starts, the utility displays an information message about itself and the database passed to it.

5. The utility displays information about the encryption in use for the database:

   ```
   Database is encrypted
   (Key ID = E1D00BC2-07F4-4495-9562-394B26A2B05B).
   ```

   It then prompts for an action:

   ```
   1) Decrypt
   2) Re-encrypt with a different key
   3) Quit
   ```

6. To re-encrypt the database with a new database-encryption key, enter "2" and press **Enter**.

   To begin the process of re-encrypting the database, you must have access to the key in which the database is currently encrypted. The utility prompts for the absolute path for this keyfile, and, receiving that, the username of an administrator and that administrator's password:

```
Access original database encryption key.
(key ID = E1D00BC2-07F4-4495-9562-394B26A2B05B)
Keyfile: C:\encdb\dek1
Username: dek-admin1
Password: <characters shielded during entry>
```

7. Once this information has been successfully entered, the utility prompts for the absolute path of the keyfile for re-encrypting the database. Receiving that, it prompts for the username of an administrator and that administrator's password:

   ```
   Access new database encryption key.
   Keyfile (full path): C:\encdb\dek2
   Username: dek-admin2
   Password: <characters shielded during entry>
   ```

   When it receives information on this second key, the utility reminds you that it modifies data in place, so that you should be sure that your data is adequately backed up before continuing.

   If you wish to continue, enter "Y".

   Once you confirm that you want to perform this process, the utility encrypts the database. During encryption, the utility displays a progress indicator, as well as a reminder not to interrupt the process, such as:

   ```
   Do not interrupt this process!
   Processed:
   16000 blocks (10%)
   ```

   CAUTION: If you interrupt the process while it is incomplete, the database may be in a state in which some of its data is encrypted with one key and some of it is encrypted with another. It is impossible for Caché to read such a database, and *all the data will be lost*.

8. When the process is complete, the progress indicator changes to a completion message:

   ```
   Processed:
   153600 blocks (done!)
   ```

B.4 Using Command-line Options with cvencrypt

You can invoke cvencrypt with various command-line options. These allow you to encrypt, decrypt, or re-encrypt one or more databases or journal files with a single command. You must specify one of the following options: `-dbfile`, `-dbfilelist`, `-jrnfile`, or `-jrnfilelist`.

Note: You can also specify the path of a journal or database file as the only argument to cvencrypt, which directs the program to decrypt or encrypt the file in interactive mode. In this situation, cvencrypt determines whether it is operating on a journal file or database file. It provides a confirmation prompt before modifying the file.

The behavior of the `-inkeyfile` and `-outkeyfile` options depends on whether an encrypted or unencrypted database is being processed, and, if an encrypted database is present, if one or both of the options is present. For details, see the description of each option.

The available command-line options are:

-dbfile *argument*

The database file to encrypt, decrypt, or re-encrypt. *argument* can be a simple file name, a file name containing a relative path, or a file name containing an absolute path.

-dbfilelist *argument*

The name of a file containing a list of databases to process. *argument* can be a simple file name, a file name containing a relative path, or a file name containing an absolute path. In the file containing the list, each entry is the name of a database file, which can also be a simple file name, a file name containing a relative path, or a file name containing an absolute path; place the name of each database file on its own line without any additional punctuation or delimiting characters.

-inkeyfile *argument*

The input key file, which contains the database key for decrypting any encrypted input to cvencrypt. When processing an unencrypted database, the `-inkeyfile` option has no effect. When processing an encrypted database, cvencrypt uses the input key file to decrypt the database. If the input key file and the output key file are both present, then, when processing an encrypted database, cvencrypt uses the input key file to decrypt the database and the output key file to re-encrypt it; if input key file and output key file hold the same database encryption key, cvencrypt does not run.

-inpass *argument*

The password to use (along with the `-inuser` username) to extract the database encryption key from the input key file. If you do not use the `-inpass` option with cvencrypt, it prompts you for a password.

Important: If you use the `-inpass` option, then this password may be visible to operating system tools as part of the data associated with the cvencrypt process. For example, on certain versions of UNIX®, the **ps** command displays the value of the `-inpass` password. If you have any concerns about exposing this information to those who should not have it, do not use the `-inpass` option; when you invoke cvencrypt, you will be prompted for the password, which will then *not* appear as part of the data associated with the process.

-jrnfile *argument*

The journal file to encrypt, decrypt, or re-encrypt. *argument* can be a simple file name, a file name containing a relative path, or a file name containing an absolute path.

The output journal file preserves the endianness of the input file, which may differ from the native endianness.

-jrnfilelist *argument*

The name of a file containing a list of journal files to process. *argument* can be a simple file name, a file name containing a relative path, or a file name containing an absolute path. In the file containing the list, each entry is the name of a journal file, which can also be a simple file name, a file name containing a relative path, or a file name containing an absolute path. Place the name of each journal file on its own line without any additional punctuation or delimiting characters.

The output journal files preserve the endianness of the input files, which may differ from the native endianness.

-inuser *argument*

The administrator name to use (along with the `-inpass` password) to extract the database encryption key from the input key file. If you do not use the `-inuser` option with cvencrypt, it prompts you for a username.

-outkeyfile *argument*

The output key file, which contains the database key for encrypting the output from cvencrypt. When processing an unencrypted database, cvencrypt uses the output key file to encrypt it. When processing an encrypted database where there is a value for the `-outkeyfile` option but no value provided for the `-inkeyfile` option, cvencrypt

does nothing. When processing an encrypted database, if there are values for both the input and output key files, then cvencrypt uses the input key file to decrypt the database and the output key file to re-encrypt it; if the input and output key files hold the same database encryption key, cvencrypt does not run.

-outpass *argument*

The password to use (along with the `-outuser` administrator name) to extract the database encryption key from the output key file. If you do not use the `-outpass` option with cvencrypt, it prompts you for a password.

Important: If you use the `-outpass` option, then this password may be visible to operating system tools as part of the data associated with the cvencrypt process. For example, on UNIX®, the **ps** command displays the value of the `-outpass` password. If you have any concerns about exposing this information to those who should not have it, do not use the `-outpass` option; when you invoke cvencrypt, you will be prompted for the password, which will then not appear as part of the data associated with the process.

-outuser *argument*

The administrator name to use (along with the `-outpass` password) to extract the database encryption key from the output key file. If you do not use the `-outuser` option with cvencrypt, it prompts you for a username.

C

Frequently Asked Questions about Caché Security

Troubleshooting

When users attempt to use the Management Portal, they are either prompted to log in as they move among its sections or unexpectedly lack privileges on certain pages or are not allowed to perform certain operations. Why is this and how can I correct it?

The Management Portal consists of several separate web applications. The main page of the Portal is associated with the /csp/sys application and other pages are associated with various /csp/sys/* applications (such as the security-related content, which is associated with the /csp/sys/sec application). If the applications do not all have a common set of authentication mechanism(s) in use, users going from one Portal page to another may encounter login prompts or sudden shifts in their level of privilege.

For example, if the /csp/sys application is using password authentication exclusively, while other related Portal applications are using unauthenticated access exclusively, then, as users move from one Portal page to another, they go from unauthenticated access to requiring authentication. Another possible case is this: the /csp/sys application supports only password authentication, the other applications support only unauthenticated access, and UnknownUser has no special privileges; in this case, when users go from the Portal's main page to its other pages, they may not have sufficient privileges to perform any action.

To check and configure the authentication mechanism for a web application, select the application from the **Web Applications** page in the Portal (**System Administration** > **Security** > **Applications** > **Web Applications**) and, for the displayed application, make selections under **Allowed Authentication Methods** as appropriate (typically, so that /csp/sys and /csp/sys/* share a common set of authentication mechanisms).

Upgrading

These questions address questions for users of Caché 5.0 and previous versions who are considering the security implications of upgrading to Caché 5.1 or more recent versions.

- Must I use Caché security?
- What do I need to be aware of when upgrading to the Caché security in version 5.1 or later?
- What operational aspects of Caché security (5.1 or later) differ from previous versions of Caché?

Must I use Caché security?

Yes. Security in Caché is always enabled. However, you can configure an instance's security to mimic the openness of an older system and to support legacy systems without any visible effects.

What do I need to be aware of when upgrading to the Caché security in version 5.1 or later?

The following items require attention when upgrading:

- All users require new passwords assigned to them after an upgrade installation.

  The password hash function used is more robust than those used in earlier versions of Caché. Since Caché only stored (and stores) the hashed form of the password for comparison, there is no way to invert the hashed form (giving a plaintext password) and replace it with the hashed value using the newer function. As a result, to take advantage of this robustness, users need to enter new passwords.

- By default, developers do not have privileges on many of the Caché services they did under prior versions.

  The default installation of Caché is configured with a relatively limited set of features accessible by default. The pre-defined roles do not include privileges for legacy resources such as COM ports, which most customers do not need. As necessary, administrators can alter the predefined roles or create new roles that provide a different set of privileges to meet the needs of each site.

What operational aspects of Caché security (5.1 or later) differ from previous versions of Caché?

An instance of Caché 5.1 or later operates slightly differently from prior versions of Caché. Differences include:

- As of Caché 5.1, ObjectScript has two system variables, *$USERNAME* and *$ROLES* that help applications manage their security needs. Both these variables can be used programmatically in routines, methods, and SQL statements.

- For CSP and Zen applications, security information is maintained as part of the CSP session. The values of *$USERNAME* and *$ROLES* are preserved across page requests, even if different processes are used to execute those requests. More specifically, when processing begins for a CSP page, *$ROLES* contains the user's roles as well as roles defined for the application.

  The session does not contain roles that were dynamically added during processing of a previous page by setting the value of *$ROLES* or invoking **$SYSTEM.Security.AddRoles**. This is true for both stateless sessions and those that maintain state.

D

Relevant Cryptographic Standards and RFCs

The following are standards and RFCs (requests for comment) that define the cryptographic primitives and algorithms used in Caché security:

- AES (Advanced Encryption Standard) encryption — FIPS (Federal Information Processing Standards) 197
- AES Key Wrap —
  - NIST (National Institute of Standards and Technology) document "AES Key Wrap Specification" (http://csrc.nist.gov/CryptoToolkit/kms/AES_key_wrap.pdf)
  - IETF (Internet Engineering Task Force) RFC 3394
- Base64 encoding — RFC 3548
- Block padding — PKCS (Public-Key Cryptography Standards) #7 and RFC 2040
- CBC (Cipher Block Chaining) cipher mode — NIST 800-38A
- Deterministic random number generator —
  - FIPS PUB 140-2, Annex C
  - FIPS PUB 186-2, Change Notice 1, Appendix 3.1 and Appendix 3.3
- GSS (Generic Security Services) API —
  - The Kerberos Version 5 GSS-API Mechanism — RFC 1964
  - Generic Security Service Application Program Interface, Version 2, Update 1 — RFC 2743
  - Generic Security Service API Version 2: C Bindings — RFC 2744
  - Generic Security Service API Version 2: Java Bindings — RFC 2853
- Kerberos Network Authentication Service (V5) — RFC 1510
- Hash-based Message Authentication Code (HMAC) — FIPS 198 and RFC 2104
- Message Digest 5 (MD5) hash — RFC 1321
- Password-Based Key Derivation Function 2 (PBKDF2) — PKCS #5 v2.0 and RFC 2898
- Secure Hash Algorithm (SHA-1) — FIPS 180-2 and RFC 3174

All these documents are available online:

- FIPS documents
- NIST documents
- PKCS documents
- RFCs (IETF)

E

About PKI (Public Key Infrastructure)

This appendix covers the following topics:

E.1 The Underlying Need

There are many situations involving computing that need security. One familiar example is how to protect information – such as for a business transaction – that is traveling across a network that is not secure – such as the Internet. Another is when there needs to be some kind of verifiable, legally binding digital signature.

One of the most common ways of providing security is through the use of *public-key cryptography*. Public-key cryptography is a mathematical algorithm that enables the encryption and decryption of data. Everyone who is using public-key cryptography holds two keys – one of which is public and one of which is private; the algorithm can use either key to perform an operation, such as encryption, which then means that the other key can perform the complementary operation, in this case, decryption. Using these keys and operations, public-key cryptography provides the means of providing security, such as protecting data in transit or establishing the provenance of a document.

However, public-key cryptography alone does not provide sufficient confidence of the identity of the participants in activities, particularly if those participants do not know each other personally. To provide enough confidence to be able to use public key cryptography in an unsecured setting, such as over the Internet, there needs to be a larger structure that also provides trustworthy and verifiable identification information for entities involved. Such a structure is known as a *Public Key Infrastructure (PKI)*.

A PKI provides a means so that individuals or organizations – generally speaking, known as entities – without any direct personal knowledge of or contact with each other can be confident of each other's identities. This depends on each of the entities trusting a third party who vouches for the identity of the other *entity* (also known as the other *peer*). This allows the entities to perform meaningful and legally binding cryptographic operations; the operations include encryption, decryption, and digital signing and signature verification.

You can use a PKI for multiple purposes:

- Protecting communications via SSL/TLS (Secure Sockets Layer or its successor Transport Layer Security)
- XML digital signatures, with or without communication
- Data encryption
- Digital signing

E.2 About Public-Key Cryptography

Public-key cryptography operates on data controlled by distinct entities, where entities can be people, applications, organizations, and so on. Each entity has a *private key*, which is a closely held secret, and a *public key*, which is made widely available. (Usually, the public key is encapsulated in a *public key certificate*, which holds both the public key itself and identifying information about the key holder; see below for more information about certificates and certificate authorities.)

A public-key algorithm uses the two keys for the complementary operations, where either key can perform either operation. (The most commonly used public-key algorithm is the RSA algorithm, developed by Ron Rivest, Adi Shamir, and Leonard Adleman.) All functionality associated with public-key cryptography is based on this fundamental principle: data encrypted with one key can only be decrypted with the other.

Hence, if you use your private key to encrypt content, only your public key can be used to decrypt it; conversely, if someone else encrypts content with your public key, only your private key – and therefore, only you – will be able to decrypt it. This means that public-key cryptography provides a means for secure and private communications between two entities. If I send you a message that is signed with my private key and encrypted with your public key, this gives you content that you can trust as being from me (since only I have access to my private key) and that only you can read (since only your private key can decrypt it).

(Note that there is also a simpler case, in which only one entity has a key pair and certificate. This is an arrangement for situations where only the identity of that peer needs to be verified.)

The uses of public-key cryptography include:

- Encryption of data with the public key, so that only the private key can decrypt it. This use enables the transfer of data in secret. For example, the encrypted data may be some medical records, so that one peer can securely transfer such records to the other peer, and can be confident that only the other peer is able to decrypt them.
- Encryption of data (or of a hash based on the data) with a known signer's private key, so that anyone can decrypt the data or the hash with the signer's public key. This use establishes the provenance of the data. For example, the encrypted data may be a legal document, so that the action of encrypting the data with a private key associated with a known individual constitutes a legally binding digital signature on that document.
- Encryption of data with the private key when the identity of the signer is unknown, so that it is possible to establish the identity of the signer. This use is based on the unique linkage of a private key to a public key, the use of a certificate to bind a public key to an entity, and the ability to look up an entity based on a certificate. For example, using only a message signed with a private key, it is possible to determine the provenance of that message (that is, its origin and, most specifically, its signer).

E.3 Authentication, Certificates, and Certificate Authorities

For public-key cryptography to be useful among users who do not know each other and cannot easily perform out-of-band *authentication* (verification of identity), there needs to be a way to determine what public key is associated with what user. The mechanism to attain this end is the certificate, which is issued by a *certificate authority (CA)*. A certificate is a document

that is digitally signed by a trusted third party (the CA) and that ties the public key to a set of identifying information about its owner, such as a name, organization, location, and so on.

Typically, a CA is an independent organization, such as VeriSign, or within an organization. For either external or internal CAs, there can also be *intermediate CAs*; these serve as subordinate CAs to the uppermost or *root CA*. For example, an entire organization may have a root CA, and divisions or offices may each have their own intermediate CAs.

Entities do not need to use the same CA – each (or, more likely, each one's software) simply needs to trust the other's CA. This relationship of trusting a CA is usually established without any user intervention, such as by having a browser ship with a set of pre-approved CA certificates. For example, in Firefox 14.0, you can see a list of trusted CAs in the Tools > Options dialog, on the Advanced tab, by clicking the View Certificates button and the Authorities tab. If a user attempts to connect to a site that uses SSL/TLS and that has a trusted CA, then the user's browser is able to authenticate the site.

Sometimes, the entity receiving the connection may also want to authenticate the entity initiating the connection. This is known as *mutual authentication*. Again, it does not typically require human intervention.

When two entities need to authenticate each other, they can do so using their certificates and the CA's trusted relationship to them. Hence, when Alice and Bob attempt to communicate via, say, SSL/TLS, the SSL/TLS handshake performs authentication for each of them as follows:

- Alice ends up with Bob's certificate. Alice can trust this certificate because Bob's CA has signed it. Bob's CA is either a trusted CA or itself has a certificate that, going up a chain of certificates, is ultimately signed by a trusted CA.
- The same is true with Alice's certificate and Bob.

As another example, suppose that a large company has its headquarters in Sao Paolo, as well as offices in Tokyo and Istanbul. This company might have its root CA in Sao Paolo, with intermediate CAs in Tokyo and Istanbul. Suppose that there are entities in Tokyo and Istanbul that need to authenticate each other. When the one in Tokyo receives the other's certificate, it sees that the certificate is signed by the Istanbul CA, which, in turn, has a certificate signed by the root CA in Sao Paolo. Likewise for the entity in Istanbul verifying the certificate signed by the Tokyo CA.

E.4 How the CA Creates a Certificate

When an entity obtains a certificate from a CA, a number of events have occurred – frequently without being visible to the user.

First, an algorithm creates the key pair. It then obtains necessary information to describe the entity using the key pair, which has to do with the entity's location, organization, and so on. Taken all together, this identifying information comprises a *distinguished name (DN)*. The entity provides the public key and DN information to the CA in the form of a *certificate signing request (CSR)*; it does not provide the private key because, again, this is a closely held secret.

The CA receives the CSR and then processes it according to its procedures (which vary by CA and by the degree of authenticity to which the certificate attests). Ultimately, the CA signs a document that binds the public key to the DN information, thereby creating a certificate (specifically, a certificate that conforms to X.509 standard).

E.5 Limitations on Certificates: Expiration and Revocation

A certificate has an expiration date. This requires the owner of the certificate to renew it regularly. And even CA certificates have expiration dates, which is one reason why you may occasionally see messages asking if you are willing to accept a non-valid certificate for a site – it may be because the site's owner has let its certificate lapse.

If a certificate has been compromised, CAs also have a means declaring it non-valid, using what are called *certificate revocation lists (CRLs)*. A CRL is a document that a CA publishes and that specifies all certificates for which it revokes its certification. A CRL is analogous to a list of people from whom a store will not accept a personal check; in fact, the analogy holds well, since the default case both for the CA and the store is that certificate or the check is valid, and the CRL or the list specifies the certificates or checks that are not acceptable.

E.6 Recapping PKI Functionality

The different activities of the CA and its clients are made possible because these are part of a public-key infrastructure (a PKI). To review:

- A PKI is an implementation of a system to create and manage private keys and certificates containing public keys. Certificates provide a means of associating a public key with an entity, so that there can be assurance of the identity of entities; to do this, a PKI includes a trusted third party known as a certificate authority (CA).
- A PKI associates an ID and other important information with a public key. Since a public key implies the existence of an associated private key, the ID associated with the public key is thereby also associated with the private key.
- Taken all together, a PKI provides the functionality so that entities in an unsecured environment can have sufficient confidence to use public-key cryptography in meaningful and legally binding ways.

F

Using Character-based Security Management Routines

The preferred and recommended way to manage a Caché installation is the Management Portal. The portal provides a convenient, browser-based interface for controlling the system. However, to cover those instances when the system cannot be managed this way, Caché also has several character-based routines that collectively provide many of the same functions on the Terminal.

The utilities described in this appendix are:

- ^SECURITY — addresses the setup and maintenance of the data essential to the proper functioning of Caché security.
- ^EncryptionKey — supports operations for encryption key management, database encryption, and data element encryption.
- ^DATABASE — is used to manage databases; it also allows you to set values related to Caché security.
- ^%AUDIT — allows the reporting of data from the logs, and the manipulation of entries in the audit logs as well as the logs themselves.

Each of the routines is described in its own section along with its top-level functionality. In most cases, the initial menu choice will lead to further requests for information until the routine has sufficient information to accomplish its task. To use any routine from the Terminal, the user must be in the %SYS namespace and have at least the **%Manager** role. The routine, for example ^SECURITY, is invoked as expected with the command:

```
DO ^SECURITY
```

When the routine runs, it presents you with a list of options. Select an option by entering its number after the "Option?" prompt.

CAUTION: As previously noted, the preferred way to manage a Caché system is via the Management Portal. Administrators who elect to use the routines described in this documents are assumed to have a detailed operating knowledge of how Caché works and what parameter values are appropriate for the options they choose.

General notes about prompts

The following are characteristics of prompts when using the character-based facilities:

- Each option has a numeric prefix. Select an option by typing its number. The option-number pattern is used throughout the routines.
- All option lists have an item to exit this level of menu and return to the previous level. You may also reply to the "Option?" prompt with **Enter**. This is interpreted as if you had chosen the "Exit" option, that is, you are finished with

that section and you are presented with the next "upper" level of options. An **Enter** reply to the top-level of options exits the **^SECURITY** routine.

- Many of the prompts for information have a default value which is selected by typing the **Enter** key. When there is a default value available, it is shown after the prompt message and followed by the characters "=>" as in

  ```
  Unsuccessful login attempts before locking user? 5 =>
  ```

  where the default value is 5 for the number of times a user may try to login and fail before the system locks their username.
- Prompts whose defaults are "Yes" or "No" also accept any matching partial response such as "yE" or "n". The match is done ignoring the case of the response.
- In options whose intent is to alter the characteristics of existing user, roles, services, and so on, the existing value of the item is displayed as the default. Typing **Enter** preserves that value and moves on to the next prompt.
- Some prompts ask for a pattern to use when matching items such as user names. The default pattern is usually "*" that matches all items. In such patterns the asterisk matches any sequence of characters, much like it does in DOS. A pattern may also consist of a comma-separated list of items each of which is treated as its own pattern. An item is treated as being selected if it matches any pattern in the list.

CAUTION: There is nothing to prevent multiple instances of the same routine from being executed at the same time by different system administrators (or even the same administrator). If this happens, it is the responsibility of the administrators to coordinate their activity to avoid conflicts and achieve their objectives with regard to the coherence of the affected data.

F.1 ^SECURITY

This routine addresses the setup and maintenance of the data essential to the proper functioning of Caché security. The initial menu includes:

1. User setup

   Users represent actual people or other entities who are permitted access to the system. This is the section for define the characteristics of users for the instance.

   Note: User definitions for Caché 2014.1 and later versions are not compatible with user definitions for 2013.1 and previous versions, due to the introduction of the AccountNeverExpires and PasswordNeverExpires fields. If you attempt to import newer definitions into an older version, Caché skips them.

2. Role setup

   Caché users are given permission to perform an action by their assignment to one or more roles. This section is where the characteristics of roles are defined.

3. Service setup

   Services control the ability to connect to Caché using various connection technologies. They are predefined by InterSystems. The parameters governing their use are set through this option.

4. Resource setup

   Resources represent assets, such as databases or applications, whose use is to be managed. A resource may represent a single asset such as a database, or it may protect multiple (usually related) assets such as a suite of applications.

5. Application setup

Application definitions serve as proxies for the actual application code. Permissions on the definition are interpreted by the security system as granting the same permission on the application associated with the definition.

6. Auditing setup

   Auditing is the means by which Caché keeps a record of security-related events. This section deals with the definition and management of events whose occurrence is to be noted in the audit log.

7. Domain setup

   Domains permit a community of users to be partitioned into several groups. This option allows an administrator to set up Caché security to accept users from multiple domains. The domains defined via this option exist only within the Caché system for the purpose of recognizing valid users. When multiple domains have been defined, usernames should include the domains they will be attempting access from, for example, president@whitehouse.gov. If a user's name is given without the domain identification, Caché uses the default domain (if any) set up in the system parameters section.

8. SSL configuration setup

   SSL/TLS provides authentication and other functionality. This section provides configuration tools if the instance uses Caché support for the SSL/TLS protocol; this includes the use of SSL/TLS with mirroring, such as for creating and editing SSL/TLS configurations for use with mirroring.

9. Mobile phone service provider setup

   With two-factor authentication, authenticating users receive a one-time security code on their mobile phone that they then enter at a prompt. This section provides the tools for configuring the mobile phone service providers in use for the Caché instance.

10. OpenAM Identity Services setup

    OpenAM identify services allow Caché to support single-sign on (SSO); by using this feature, users that have already successfully authenticated do not have to re-authenticate. This section deals with managing OpenAM identity services for the Caché instance.

11. Encryption key setup

    Caché uses keys to encrypt databases or user-specified data elements. This section provides tools for working with keys for both database and managed encryption.

12. System parameter setup

    The system parameters are a collection of security-related values that apply system-wide. This section includes the ability to export and import all an instance's security settings, including those for SQL privileges.

    Note: Considerations related to importing settings:

    - If you are importing security settings from a source instance configured with multiple domains to a target instance not configured to allow multiple domains *and* the source instance's default domain differs from that of the target instance, then the import does not update the target's default domain — you must explicitly set this value. To do this, use the **Default security domain** drop-down on the **System-wide Security Parameters** page (**System Administration** > **Security** > **System Security** > **System-wide Security Parameters**).
    - When importing *all* security settings, the import/export file includes web application settings; each web application has a *Path* setting. Before importing settings onto a new drive, VM, or hardware, for each web application, ensure that the value of the *Path* setting is accurate for that environment. If the web applications associated with the Management Portal do not have correct *Path* values, the Management Portal will not display correctly.

      To locate the *Path* setting for each web application in the import/export file (SecurityExport.xml), look in the *ApplicationsExport* section; in each *Applications* section, identify the application by the value of the *Name* setting; then update the value of the *Path* setting as appropriate.

13. X509 User setup

    X.509 is the standard for certificates that a public key infrastructure (PKI) uses. Caché uses X.509 certificates for its PKI, and each user associated with an X.509 certificate is known as an X.509 user. This section provides tools for working with X.509 users, such as creating them, editing them, deleting them, and so on.

14. Exit

F.2 ^EncryptionKey

The **^EncryptionKey** routine is for use with managed key encryption; it supports operations for encryption key management (technology for creation and management of encryption keys and key files), database encryption, and data element encryption.

1. Create new encryption key and key file

   Allows you to create a new database-encryption key, which it stores in a key file.

2. Manage existing encryption key file

   Allows you to list administrators associated with a key file, add an administrator to a key file, remove an administrator from a key file, and convert a version 1.0 key file to a version 2.0 key file.

3. Database encryption

   Allows you to activate a database encryption key, display the unique identifier for the currently activated database encryption key (if there is one), deactivate the activated database encryption key, and configure Caché startup options related to database encryption.

4. Data element encryption for applications

   Allows you to activate a data element encryption key, list the unique identifier for any currently activated data element encryption keys (if there are any), and deactivate the activated data element encryption key.

F.3 ^DATABASE

The **^DATABASE** routine is used to manage databases; it also allows you to set values related to Caché security.

1. Create a database

   Allows you to create a new database.

2. Edit a database

   Allows you to change the characteristics of an existing database, for example, by adding additional volumes.

3. List databases

   Displays the characteristics of one or more databases.

4. Delete a database

   Allows you to delete a Caché database. This action is irreversible.

5. Mount a database

Makes a database ready for use by Caché. Databases must be mounted to Caché in order to be usable. Databases can be set to be automatically mounted at startup.

Note: You can use the **Mount a database** option to mount any CACHE.DAT file accessible to the instance by specifying the directory containing it. However, if you do this with a database that was deleted from, or was never added to, the Management Portal database configuration (see Configuring Databases in the "Configuring Caché" chapter of the *Caché System Administration Guide*), the database is not added to the Management Portal configuration and is therefore unavailable for portal database operations and for some routines, for example **^Integrity** (see Checking Database Integrity Using the ^Integrity Utility in the "Introduction to Data Integrity" chapter of the *Caché Data Integrity Guide*).

6. Dismount a database

   Permits you to quiesce a database and remove it from use by Caché.

7. Compact globals in a database

   Reorganizes the data inside *CACHE.DAT*. Note that this option does not reduce the size of the database file; to reduce the size of the database, see option 13.

8. Show free space for a database

   Displays the available space for a database. This is calculated as the difference between its current contents and its current declared size.

9. Show details for a database

   Displays detailed information on a specified database including location, size, status, and other controlling parameters.

10. Recreate a database

    Creates a new, empty database with the parameters of the original database. The new database is the same size as the original database.

11. Manage database encryption

    Removes all the logical data from a database while preserving the properties of the database for reuse.

12. Return unused space for a database

    Frees either a specified amount of or all available extra space associated with a database, reducing it from its current size to its smallest possible size.

13. Compact freespace in a database

    Specifies the desired amount of freespace (unused space) that is in a database after the end of the database's data. You can also eliminate this freespace using the Return unused space for a database option (#12).

14. Defragment globals in a database

    Defragments a database, which organizes its data more efficiently. Defragmentation may leave freespace in a database (see options #12 and #13).

F.4 ^%AUDIT

This routine allows the reporting of data from the logs, and the manipulation of entries in the audit logs as well as the logs themselves.

1. Audit reports

Permits you to specify selection criteria (date ranges, events, affected users, and so on) and display characteristics, then extracts the data from the audit log and formats it for presentation.

2. Manage audit logs

   Allows the extraction of log entries to another namespace, the export and import of audit log data to and from external files, and maintenance activities against the audit log itself.

3. Exit