Authorization Guide

Version 2023.1
2024-07-11

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

Authorization Guide
InterSystems IRIS Data Platform Version 2023.1 2024-07-11

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 Tables

1

About InterSystems Authorization

Once a user has authenticated, the next security-related question to answer is what assets that person is allowed to use, view, or alter. Assets include:

- Databases — Physical files containing data or code.
- Services — Tools for connecting to InterSystems IRIS, for example, client-server services, telnet.
- Applications — InterSystems IRIS programs, for example, Web applications.
- Administrative actions — Sets of tasks, for example, starting and stopping InterSystems IRIS or creating backups.

You probably do not want all of the users in your organization to be able to see and modify every asset on your system. The determination and control of access to assets is known as *authorization*.

Authorization manages the relationships of users and assets, which are represented within the InterSystems IRIS® data platform as *resources*. InterSystems IRIS employs *Role-Based Access Control (RBAC)* as its authorization model: a system administrator assigns a user to one or more task-based *roles*; each role is authorized to perform a particular set of activities with a particular set of resources. *Applications* can temporarily expand the roles a user has.

This page provides an overview of the RBAC authorization model implemented within InterSystems IRIS. For an interactive introduction to InterSystems RBAC, try Configuring Role-Based Access.

1.1 Resources, Permissions, and Privileges

The primary goal of security is the protection of *assets* — information or capabilities in one form or another. With InterSystems IRIS data platform, assets can be databases, services, applications, tools, and even administrative actions.

Each asset is represented in InterSystems IRIS by a *resource*, and a single resource can represent more than one asset.

The system administrator controls access to an asset by assigning *permissions* to a resource. Granting or revoking a permission enables or disables access to an activity which can be performed upon the asset the resource represents. For databases, the permissions are Read and Write; for most other resource types, the relevant permission is Use.

Together, a pairing of a resource and an associated permission is known as a *privilege*. This is often described using the following shorthand: `Resource-Name:Permission`. For example, a privilege granting read and write permissions on the EmployeeInfo database is represented as `%DB_EmployeeInfo:Read,Write` or `%DB_EmployeeInfo:RW`.

See Using Resources to Protect Assets and Privileges and Permissions for more details.

1.2 Users and Roles

Within the InterSystems role-based access control model, a user gains the ability to manipulate resources as follows:

1. *Resources* are associated with *permissions* to establish *privileges*, as described in the preceding section.
2. A set of *privileges* is assigned to a *role*.
3. *Roles* have members, such as *users*.

A *user* connects to InterSystems IRIS to perform some set of tasks. A *role* describes a set of privileges that a user holds, and thus the tasks that user may perform.

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 can be a member of more than one role. Using the same example, the medical director for the hospital may require capabilities used by doctors across all departments. This user can be assigned both the **`RoundsDoctor`** and **`ERDoctor`** roles. Alternatively, the system administrator can create a **`MedicalDirector`** role which is itself a member of both these roles, and inherits privileges accordingly.

The native InterSystems implementation of role-based access control is available with every type of authentication mechanism that InterSystems IRIS supports, including LDAP, Kerberos, and OS-based. If you prefer, you can also choose to assign roles using LDAP or delegated authorization. See Roles and User Accounts for more details.

1.3 Applications

InterSystems security provides a flexible application security model. The ability to use an application is a resource, so you can restrict the use of an application to a particular set of users, or open it to all users. For users who can use an application, the security model supports a role escalation model. This means that while using an application, users can access specific resources that they could not generally access.

See Applications for more information about the multiple types of applications.

2

Using Resources to Protect Assets

Authorization protects InterSystems IRIS® data platform components from inappropriate use. InterSystems uses the following terminology when discussing these components:

- An *asset* is something that is protected. For instance, an InterSystems IRIS database is an asset, the ability to connect to InterSystems IRIS using SQL is an asset, and the ability to perform a backup is an asset.
- *Resources* protect assets. 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 example, a variety of system management functions are protected by a single resource.
- 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 Privileges and Permissions.

This topic addresses issues related to resources and the assets that they protect. InterSystems IRIS includes a set of resources for assets that it protects — and provides access for users based on the rights that they hold. You can also define your own resources.

2.1 Types of Resources

There are multiple resource types:

- System resources — For controlling the ability to perform various actions for an InterSystems IRIS instance. For more information on these resources, see System Resources.

  These resources are **`%Admin_ExternalLanguageServerEdit`**, **`%Admin_Journal`**, **`%Admin_Manage`**, **`%Admin_Operate`**, **`%Admin_RoleEdit`**, **`%Admin_Secure`**, **`%Admin_Task`**, **`%Admin_UserEdit`**, **`%Development`**, **`%DocDB_Admin`**, **`%IAM`**, **`%System_Callout`**, **`%System_Attach`**, and **`%Secure_Break`**.
- Database resources — For controlling read and write access to InterSystems IRIS databases. For more information on these resources, see Database Resources.

  The database resources for a newly installed InterSystems IRIS instance are **`%DB_IRISLOCALDATA`**, **`%DB_IRISAUDIT`**, **`%DB_IRISLIB`**, **`%DB_IRISLOCALDATA`**, **`%DB_IRISSYS`**, **`%DB_IRISTEMP`**, **`%DB_ENSLIB`**.
- Gateway resources — For controlling access to external language servers. For more information on these resources, see Gateway Resources.

The gateway resources for a newly installed InterSystems IRIS instance are **%Gateway_Object**, **%Gateway_SQL**, and **%Gateway_ML**.

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

  Not all services have associated privileges — only those services for which InterSystems IRIS provides user-based access; other services, such as data check, are not user-based and, as a result, do not have associated security resources. For more information on managing services, see Services.

  The service resources are **%Service_CallIn**, **%Service_ComPort**, **%Service_Console**, **%Service_DocDB**, **%Service_Login**, **%Service_Native**, **%Service_Object**, **%Service_SQL**, **%Service_Telnet**, **%Service_Terminal**, and **%Service_WebGateway**.

- 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 Application Resources. For information on creating these resources, see Create or Edit a Resource.

2.2 System Resources

InterSystems IRIS comes with a set of built-in resources that govern actions in relation to the installed InterSystems IRIS instance. System resources include:

- Administrative Resources
- The %Development Resource
- The %DocDB_Admin Resource
- The %IAM Resource
- The %System_Callout Resource
- The %System_Attach Resource
- The %Secure_Break Resource
- The %Service_Native Resource

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

2.2.1 Administrative Resources

The administrative resources are:

- %Admin_ExternalLanguageServerEdit
- %Admin_Journal
- %Admin_Manage
- %Admin_Operate
- %Admin_RoleEdit
- %Admin_Secure
- %Admin_Tasks

- %Admin_UserEdit

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

2.2.1.1 %Admin_ExternalLanguageServerEdit

This resource controls the ability to create, modify, or delete an external language server (also known as a gateway), including changing the gateway resource associated with it.

This resource takes the Use permission.

By default, the **%Manager** role holds the **%Admin_ExternalLanguageServerEdit:USE** privilege.

2.2.1.2 %Admin_Journal

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.

This resource takes the Use permission (not Read or Write).

2.2.1.3 %Admin_Manage

This resource controls multiple sets of privileges:

- It controls access to various pages in the Management Portal, including the System Administration page.
- It controls the ability to:
  - Create, modify, and delete InterSystems IRIS 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.

This resource takes the Use permission.

2.2.1.4 %Admin_Operate

This resource controls multiple sets of privileges:

- It controls access to various pages in the Management Portal, including the System Operation page.
- It controls the ability to:
  - Start and stop InterSystems IRIS.

  - 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 an ObjectScript utility) or implicitly (such as when making a global reference to an un-mounted database).

This resource takes the Use permission.

2.2.1.5 %Admin_RoleEdit

This resource controls the following privileges:

- For SQL, it controls the ability to:
  - Create or delete a role.

This resource takes the Use permission.

2.2.1.6 %Admin_Secure

This resource controls multiple sets of privileges:

- It controls access to various pages in the Management Portal.
- When working with the RBAC security model, it controls the ability to:
  - Create, modify, or delete a user.
  - Create, modify, or delete a role.
  - Create, modify, or delete application definitions and application resources.
  - Modify audit settings.
  - Modify services.
- For SQL, it controls the ability to:
  - Create, modify, or delete a user.
  - Create, modify, or delete a role.
  - See the privileges granted to a user.
  - See the privileges granted to a role.
  - Revoke SQL privileges that were granted by another user.

This resource takes the Use permission.

2.2.1.7 %Admin_Tasks

This resource's privileges include the ability to create, modify, or run a task, such as through the Management Portal's Task Manager (**System Operation** > **Task Manager**).

This resource takes the Use permission.

2.2.1.8 %Admin_UserEdit

This resource controls the following privileges:

- For SQL, it controls the ability to:
  - Create, modify, or delete a user.

This resource takes the Use permission.

2.2.2 The %Development Resource

The **%Development** resource controls access to InterSystems IRIS 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 InterSystems IRIS system manager utility. (This privilege is also required to call any APIs that provide programmatic access to this functionality.)
- Use the debugging facilities of InterSystems IRIS, including the **BREAK** and **ZBREAK** commands and the debug option of the process display in the InterSystems IRIS system manager utility.

The **%Development:Use** privilege works in conjunction with database privileges to control developer access to InterSystems IRIS 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 globals 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 routines 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 InterSystems IRIS 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 InterSystems IRIS 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 an InterSystems IRIS 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

2.2.3 The %DocDB_Admin Resource

The **%DocDB_Admin** resource controls the ability to manage a document database application. For more information on this feature, see the Using Document Database guide.

2.2.4 The %IAM Resource

The *%IAM* resource controls the ability to obtain a license from InterSystems IRIS to run the InterSystems API Manager (IAM).

2.2.5 The %System_Callout Resource

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

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

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

2.2.6 The %System_Attach Resource

The %System_Attach resource enables the ability to attach the Studio debugger to a running process.

2.2.7 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 Secure Debug Shell.

2.2.8 The %Service_Native Resource

The **%Service_Native** resource controls whether the user can issue Native SDK calls via Python, Java, .NET, and Node.js. The user must have the **Use** permission. The system-defined roles **%Developer** and **%Manager** have this privilege by default.

2.3 Database Resources

Database resources control access to the contents of InterSystems IRIS 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.

2.3.1 Database Resource Privileges

The available database privileges are:

Table 2–1: 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. InterSystems 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. InterSystems IRIS 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 SQL Users, Roles, and Privileges.

2.3.2 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 IRISSYS database is specified using the **%DB_IRISSYS** 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.

2.3.3 Default Database Resource

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

Table 2–2: %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.

2.3.4 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 InterSystems IRIS 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).

2.3.5 Namespaces

Users and applications interact with InterSystems IRIS 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 InterSystems IRIS 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).

2.3.6 IRISSYS, the Manager's Database

InterSystems IRIS ships with a database that provides a repository for administrative routines and globals. This is the IRISSYS 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 an InterSystems IRIS 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 routines 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 IRISSYS 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 IRISSYS. 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**); for a description of this page, see System-wide Security Parameters.

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

2.3.6.1 Special Capabilities

There are special capabilities available to code located in the IRISSYS 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 User Account Properties and the instance-specific password constraints described in the Password Strength and Password Policies.
- Invoking one of the $ZF functions, which allow you to call non-ObjectScript programs or functions from ObjectScript routines.

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 IRISSYS 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_IRISSYS** resource.

2.4 Gateway Resources

Gateway resources control access to the external language servers (also known as gateways) provided in InterSystems IRIS. The gateway resources and the types of external language server they are associated with by default are listed in the following:

- **%Gateway_ML** — IntegratedML
- **%Gateway_Object** — .NET, Java, Python, R, and XSLT
- **%Gateway_SQL** — JDBC

These resources take the Use permission.

The %Admin_ExternalLanguageServerEdit resource controls the ability to create, delete, and modify external language servers, including changing the gateway resources associated with them. You can replace the default gateway resource associated with a gateway with a user-defined resource, or remove it without replacing it, in which case it the gateway is public and can be used by anyone.

Important: InterSystems strongly recommends protecting all gateways by associating a gateway resource.

2.5 Application Resources

InterSystems IRIS 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 Use Custom Resources with the Management Portal.
- Authorization checking at a particular point in an application — for more information, see the next section, Create or Edit a Resource.
- Authorization for the whole of an application

For the whole of an application, InterSystems IRIS 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 several types of applications:

- Web application definitions
- Privileged Routine application definitions
- Client application definitions
- Document database definitions

Application resources provide a means of controlling access to applications. To use this feature, create a custom resource (as described in Create or Edit a Resource) and use it in association with the application as described in either Edit a Web Application: The General Tab or Edit a Privileged Routine Application, a Client Application, or Document Database Application: The General Tab.

For example, if a web 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 Applications.

2.6 Create or Edit 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 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 Check the Privileges of a Process for more information.

2.6.1 Resource Naming Conventions

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

Resource names are case-sensitive, but also prohibit naming as if insensitive. This means that:

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

For example, if there is a resource named **Accounting**, then you cannot create a resource named **ACCOUNTING**, **accounting** or any combination of letter casing therein. References to the **Accounting** resource using capitalization such as **accounting** or **ACCOUNTING** do not succeed. If you delete the **Accounting** resource and subsequently make an **accounting** resource, references to **Accounting** fail. You must specify the correct letter casing when referencing a resource.

2.7 Use 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:

- Define and Apply a Custom Resource to a Page
- Remove 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.

2.7.1 Define and Apply 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**.

2.7.2 Remove 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**.

3

Privileges and Permissions

Permissions allow users to perform some action, such as reading or writing data, or using a tool. Permissions are associated with resources, forming privileges. A privilege is written as a resource name followed by a permission separated by a colon, such as `%DB_Sales:Read`, which describes an action a user can perform.

A group of privileges, in turn, is called a role. Performing an action requires a user to be a member of a role that holds the appropriate privilege. This model provides precision when specifying what tasks a user (or group of users) can do – to make an adjustment, simply adjust the privileges in that user's role.

3.1 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 an InterSystems 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.

3.2 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 **`%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 3–1: Default Public Privileges

Resource	Permission
%DB_IRIS	Read
%DB_IRISLIB	Read
%DB_IRISTEMP	Read, Write

3.3 Check the Privileges of a Process

InterSystems IRIS® data platform 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. (There are also methods with built-in privilege checks, described in the next section.)

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_TESTDATABASE")
```

returns `READ,WRITE` if the process holds Read and Write permissions for %DB_TESTDATABASE. 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_TESTDATABASE", "WRITE")
```

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

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

```
$SYSTEM.Security.Check("%DB_TESTDATABASE", "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_TESTDATABASE", "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.

3.4 Use Methods with Buillt-In Privilege Checks

InterSystems IRIS allows methods to require that the process that calls them has certain specified privileges.

This feature uses the Requires method keyword. The Requires method keyword has a quoted string value that is a comma-delimited list of privileges. Each privilege names a resource and its associated permission (Use, Read, or Write) in standard format.

For example, if the **MyAction** method requires the **`Service_FileSystem:Use`** privilege, its signature might be:

```
ClassMethod MyAction() [ Requires="Service_FileSystem:Use"]
{
 // Method content
}
```

If the Requires keyword has a value, the method may only run if the calling process has the required privilege at the time that it invokes the method. If the process does not have the required privilege, the system generates a <PROTECT> error.

Methods that inherit this keyword may require additional resources by overriding and setting a new value for the keyword. There is no way to remove requirements.

3.5 When Changes in Privileges Take Effect

InterSystems IRIS maintains a persistent database of the security settings. When InterSystems IRIS 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.

4

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.

Note: For SQL access to data in tables, InterSystems IRIS® data platform supports row-level security. For information on setting this up, see Adding Row-Level Security.

4.1 About Roles

Every role has the following properties:

Table 4–1: Role Properties

Property Name	Property Description
Name	Unique role identifier. See 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 Roles, Users, Members, and Assignments.

4.1.1 About Role Assignment

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

The available role-assignment mechanisms are:

Table 4–2: Authentication Mechanisms and Role-Assignment Mechanisms

Authentication Mechanism	Role-Assignment Mechanisms
Delegated authentication (**ZAUTHENTICATE**)	**ZAUTHENTICATE**
Instance authentication	Native authorization (the primary approach described by this guide)
LDAP	LDAP
Kerberos	Options of: • Delegated authorization (**ZAUTHORIZE**) • Native authorization
Operating System authentication	Options of: • Delegated authorization (**ZAUTHORIZE**) • LDAP • Native authorization

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 UnknownUser Account and The _PUBLIC Account respectively.

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

4.1.2 Maximum Number of Roles

Each instance of InterSystems IRIS can have up to 10,240 roles.

4.2 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.

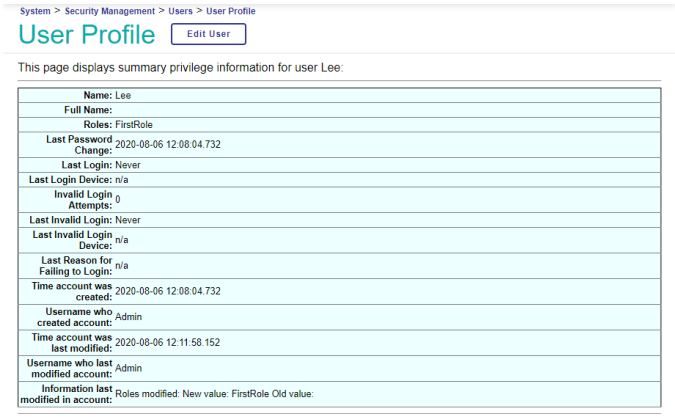
4.2.1 An Example of Multiple Role Assignment

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

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**.

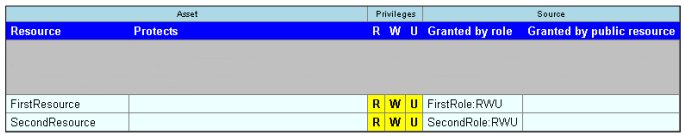
When first created, Lee is not a member of any roles. This is reflected in Lee's profile:

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:



4.3 Create 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 Manage 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.

4.3.1 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 InterSystems IRIS predefined roles.
- It can include Unicode characters.
- It cannot be the same as an existing username.

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, InterSystems IRIS ignores differences in case.

A role name can be up to 64 characters long.

For example, if there is a role named **`BasicUser`**, then you cannot create a role named **`BASICUSER`**. References to the **`BasicUser`** role using capitalization such as **`basicuser`** or **`BASICUSER`** will succeed.

4.4 Manage 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:
  - View Existing Roles
  - Delete a Role
- Creating, modifying, and removing a role's privileges. This includes:
  - Give New Privileges to a Role
  - Modify Privileges for a Role

  - Remove Privileges From a Role

- Creating and removing assignments among roles and users. This includes:
  - Assign Users or Roles to the Current Role
  - Remove Users or Roles From the Current Role
  - Assign the Current Role to Another Role
  - Remove the Current Role From Another Role
- Modify 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.

4.4.1 View 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

4.4.2 Delete 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. InterSystems IRIS displays a confirmation dialog. Click **OK** to delete the role and **Cancel** otherwise.

4.4.3 Give 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).

4.4.4 Modify 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.

4.4.5 Remove 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.

4.4.6 Assign 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.

4.4.7 Remove 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 **Remove** 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.

4.4.8 Assign 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.

4.4.9 Remove 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 **Remove** 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.

4.4.10 Modify 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

4.4.10.1 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 VIEW command.

4.4.10.2 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.

4.4.10.3 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).

4.4.10.4 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).

4.5 Predefined Roles

InterSystems IRIS 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 System Explorer, 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 Secure Debug Shell.

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 4–3: 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_IRISLOCALDATA	Read	Read	Read		
%DB_IRISAUDIT		Read			
%DB_IRISLIB	Read	Read, Write			
%DB_IRISSYS		Read, Write	Read, Write		
%DB_IRISTEMP	Read, Write	Read, Write	Read, Write		
%DB_%DEFAULT	Read, Write	Read, Write			
%Development	Use	Use			
%DocDB_Admin	Use	Use			
%Secure_Break					Use
%Service_Console					
%Service_DocDB	Use	Use	Use		
%System_Native	Use	Use			
%Service_Object	Use	Use			
%Service_SQL	Use	Use		Use	
%Service_Telnet	Use	Use			
%Service_Terminal	Use	Use			
%Service_WebGateway	Use	Use	Use		

The definitions of these predefined roles are set during a new InterSystems IRIS 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_IRISSYS: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.

4.5.1 %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 InterSystems IRIS 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 Adding Row-Level Security.

4.5.2 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. The **%DB_<database-resource-name>** roles are read-only and therefore cannot be modified; hence, for each of these roles, you cannot add privileges for other resources in addition to the RW access to the database resource for which the role is named.

4.6 Login Roles and Added Roles

Each InterSystems IRIS 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.

4.6.1 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.

4.7 Programmatically Manage 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 IRISSYS database or if the current privileges held include Write permission for the IRISSYS database (**%DB_IRISSYS: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.

5

User Accounts

User accounts represent the actual users of InterSystems IRIS® data platform. The roles a user account is a member of determine what resources that use can access. You can also give user accounts specific SQL privileges.

5.1 User Account Properties

Each user account in InterSystems IRIS has a number of properties. To view the properties for an account, navigate to the **Users** page of the Management Portal (**System Administration** > **Security** > **Users**) and select the user account you want to view.

The **General** tab for the user account contains the following properties. For information about the other tabs, see Modify a User's Roles and Modifying a User's SQL-Related Options.

Table 5–1: User Account Properties

Property Name	Property Description
Name	Unique user identifier that is up to 128 characters long. Once the user account is created, this cannot be changed. This can include any character except "@" or "*". A name is not case-sensitive. All usernames can include Unicode characters. A username cannot be the same string as an existing role.
Full Name	The user account'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 is case-sensitive, may include Unicode characters, and 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**).
Confirm Password	Confirmation of new password value.

Property Name	Property Description
Change password on next login	A check box specifying whether or not the user is required to change the password at the next login.
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.
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.
Account expiration date	The last date on which the account can be used.
Startup Namespace	The namespace in which to begin execution following login from a terminal-type service or the Portal. This property overrides any namespace value provided via the command invoking InterSystems IRIS.
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 InterSystems IRIS.
Email Address	The email address associated with this account.
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 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 `Password user`, `Delegated user`, `Kerberos user`, `LDAP user`, or `OS user`. For more information on user types, see About User Types below.

5.1.1 About User Types

The user account *Type* can be one of the following:

- Password user — This type is authenticated through Instance Authentication, Kerberos (without delegated authorization), or the operating system (without delegated authorization). The InterSystems IRIS tools for editing or otherwise altering users are for use with Password users.
- Delegated user — This type is authenticated through a user-defined authentication mechanism. The InterSystems IRIS 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, InterSystems IRIS 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 Delegated Authorization. If a user is authenticated through Kerberos without using delegated authorization, then the user is of the Password user type.

- LDAP user — This type is authenticated through LDAP. The InterSystems IRIS 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, InterSystems IRIS 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 Delegated Authorization. If a user is authenticated through the operating system without using delegated authorization, then the user is of the Password user type.

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

5.2 Manage User Accounts

To view a list of the existing user accounts, 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 User Account Properties section):

- **User** — A unique identifier for the user account
- **Full Name** — The account's displayable name
- **Enabled** — Whether or not the user account 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 account, which is determined by the authentication and role-assignment mechanisms in use

You can perform the following actions from the **Users** page:

- Create a New User Account
- Edit an Existing User Account
- View a User Profile
- Disable/Enable a User Account
- Delete a User Account

5.2.1 Create a New User Account

To create a new user account:

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 user accounts.
3. On the **Edit User** page, set values for the user properties described in the User Account Properties section.

Note: As a shortcut, if you wish to create multiple accounts with similar characteristics, you can use the **Copy from** field to begin the process. Select an existing user account from the **Copy from** drop-down menu to fill in the following fields with values from the selected account:

- Full Name
- Expiration Date
- Default Namespace
- Default Tag^Routine

4. Click the **Save** button to create the new user account.

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

5.2.2 Edit an Existing User Account

Once you have created a user account, you can alter any of its basic properties:

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 user accounts. To edit an existing account, select the name of the account from the table. This displays the **General** tab of the **Edit User** page for creating and configuring user accounts.
3. On the **Edit User** page, you can alter values for the properties described in the User Account Properties section.
4. Click the **Save** button to save the new values for the user account.

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

- Roles — Lists the roles that the user account currently holds. You can also give a user account new roles (or take them away) on this page.
- SQL Properties — This includes:
  - **SQL Privileges** — Lists all the SQL privileges that a user account 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 account 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 account 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 account 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.

5.2.2.1 Modify a User's Roles

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

- To assign a user account 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 account to the role.

- To assign a user account to all roles, click the double-arrow pointing from the **Available** list to the **Selected** list; click the **Assign** button to assign the user account to all the roles.

  Note: If you assign a user account to all roles, this includes the predefined %SecureBreak role, which limits (and does not expand) the user account's abilities. If a user account is assigned to the **%SecureBreak** role, this enables the InterSystems IRIS secure debug shell, which restricts the commands that the user may issue. This may also have unexpected consequences in other areas such as <COMMAND> errors after reaching a BREAK or another error. To resolve this, simply remove the **%SecureBreak** role from the user.

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

5.2.2.2 Modifying a User's SQL-Related Options

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

- General SQL 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 account:

- To add a privilege to a user account, 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 account. To also add the privilege of being able to grant the added privilege to other user accounts, click the relevant button below the **Available** list.
- To add all privileges to a user account, click the double-arrow pointing from the **Available** list to the **Selected** list; click the **Assign** button to give the privileges to the user account. To also add the privileges of being able to grant the added privileges to other user accounts, click the relevant button below the **Available** list.
- To remove a privilege from a user account, click the **Remove** link to the right of privilege name.
- To remove all privileges from a user account, 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 account:

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 account.

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 account'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 account has the privilege to execute the stored procedure.
5. Optionally, you can grant the users the ability to grant this privilege on other user accounts; 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 account.

To remove a user account'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 account.

5.2.3 View a User Profile

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

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

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

Table 5–2: 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 account's displayable name. This is editable on the **Edit User** page.
Roles	A comma-separated list of roles assigned to user account. These are editable on the **Roles** tab of the **Edit User** page.
Last Password Change	The date and time of user account's most recent password change.

Property Name	Property Description
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.

5.2.4 Disable/Enable a User Account

You can disable or enable a user account. For example, you can disable an account to make it temporarily unavailable; when you enable it later, you then do not have to reconstruct its properties, roles, and so on.

To disable or enable a user account:

1. From the Management Portal home page, go to the **Users** page (**System Administration** > **Security** > **Users**).
2. On the **Users** page, for the user account you wish to disable or enable, click the name of the user account. This displays the**General** tab of the **Edit User** page for that user.
3. On the **Edit User** page, clear or select the **User enabled** field.
4. Click the **Save** button to save the user account in its new state.

5.2.5 Delete a User Account

To delete a user account:

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

5.3 Predefined User Accounts

Every instance of InterSystems IRIS automatically includes the following accounts:

Table 5–3: Predefined User Accounts

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 and disable the account prior to going into production.
CSPSystem	(None)	Default account representing the Web Gateway when it connects to InterSystems IRIS via Instance Authentication 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. This user account is used internally by HealthShare and should not be disabled. **Note:** If you change this user's password for the instance, you must also change it for the web gateway.
IAM	IAM_API	Default account required to obtain a license for InterSystems API Manager (IAM) from InterSystems IRIS. To use the IAM user, you must enable it and change its password; setting up the IAM user is part of setting up IAM generally.
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 and disable the account 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	Interoperability manager (not a login account). Only on InterSystems IRIS instances. This account is used internally by HealthShare and should not be disabled.
HS_Services	%HS_ServiceRole	This account is used by healthcare products for internal purposes. By default, it is disabled for InterSystems IRIS for Health™ and HealthShare® Health Connect, and must be enabled before using FHIR®-based IHE profiles with these products. For other HealthShare solutions, the account is enabled by default and must not be disabled.

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

CAUTION: New installations of InterSystems IRIS use the same password for all the predefined accounts, as described in Initial User Account Passwords. The default password is a security vulnerability, particularly in a Minimal Security installation. To address this issue, disable the accounts or change their passwords. InterSystems recommends disabling the accounts after creating a unique account with the **%All** role, as InterSystems IRIS requires at least one enabled account with the **%All** role.

This is a critical concern with containerized instances in particular; see Authentication and passwords for more information, including ways in which you can address the issue.

Additionally, there is a user account called %System, which is not visible and which exists for InterSystems IRIS to use internally. You cannot log in to, edit, or delete this account. This account runs with the **%All** role. Certain routines, such as **%ZSTART** and **%ZSTOP**, are run as this user account; to run such a routine as a different user, call **$SYSTEM.Security.Login()**.

While it is not recommended, you can delete predefined user accounts. However, there must be at least one account with the **%All** role in addition to %System.

5.3.1 Notes on Various Accounts

5.3.1.1 The UnknownUser Account

For certain applications, or certain parts of an application, unauthenticated users may have a legitimate reason to use InterSystems IRIS, such as for a retail system to display availability of products, prior to the user initiating a purchase. For this type of situation, InterSystems IRIS 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 InterSystems IRIS via terminal and supplies a username and password that fails authentication. Here, the user is not connected to InterSystems IRIS, even if unauthenticated access is permitted; on the other hand, if unauthenticated access is permitted and that same user connects to InterSystems IRIS 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.

5.3.1.2 The _PUBLIC Account

The predefined user account, _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.

5.4 Validate 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 account is valid; if the login fails, the user account is not valid. When the routine exits (regardless of the login's success or failure), the current user account will be the one that invoked the routine.

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

ObjectScript

```
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 account 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 account must have the **`IRISSYS: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:

ObjectScript

```
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 an InterSystems IRIS 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>
```

6

Applications

Applications are the primary way that most users interact with InterSystems IRIS® data platform — and application security provides a key set of tools for regulating user access and user actions. Application security uses InterSystems authorization tools to ensure that only the appropriate users can use an application. An application can also escalate its users' privileges.

This topic covers the following topics:

- Applications, Their Properties, and Their Privileges
- Application Types
- Create and Edit Applications
- Built-In Applications

6.1 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 InterSystems authorization tools, which manage how an application and its users can interact with InterSystems products and other security resources. There are several kinds of applications:

- Web Applications
- Privileged Routine Applications
- Client Applications
- Document Database Applications

This section covers:

- Applications and Their Properties
- Associating Applications with Resources

- Applications and Privilege Escalation
- Check for Privileges Programmatically

6.1.1 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, InterSystems IRIS supports what is called an *application definition*, which is a set of information that represents the application within InterSystems IRIS. (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 InterSystems IRIS. 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 a forward slash ("/") and must be followed by an alphanumeric or any of the following characters:

/ , - , _ , . , or %.

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 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 Applications and Privilege Escalation.

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

6.1.2 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 InterSystems 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.

6.1.3 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, InterSystems IRIS 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 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 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.

6.1.3.1 User-Based and Application-Based Security

The InterSystems 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 InterSystems IRIS to provide a very flexible model:

Table 6–1: 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.

6.1.4 Check 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:

ObjectScript

```
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:

ObjectScript

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

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

6.2 Application Types

There are several types of applications:

- Web Applications
- Privileged Routine Applications
- Client Applications
- Document Database Applications

6.2.1 Web Applications

These applications connect to InterSystems IRIS using the **%Service_WebGateway** service.

For web applications, security information is maintained as part of the Web 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, 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 web session, then there is already a connection, so no further security checking is required.
3. If the Use permission for **%Service_WebGateway** is not public and the user does not hold this permission, there is no connection.
4. If the application or **%Service_WebGateway** requires authentication and the user has not already been authenticated, then InterSystems IRIS checks if the request includes *IRISUsername* and *IRISPassword* parameters:
    a. If *IRISUsername* and *IRISPassword* are present, InterSystems IRIS 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 *IRISUsername* and *IRISPassword* are not present, InterSystems IRIS 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.

For information about editing a web application, see:

- Create an Application

- Edit a Web Application: The General Tab
- Edit an Application: The Application Roles Tab
- Edit an Application: The Matching Roles Tab

6.2.1.1 An Example of Programmatically Escalating Roles in a Web Application

The following example demonstrates how to escalate application roles for a web application. It performs the following actions:

1. It changes control to the %SYS namespace. This is required to invoke the required calls, because the code for role management and escalation is only available there.
2. It adds the application's target roles.
   a. All users receive the **MYAPP** role. This is because there is no matching role listed before the colon in the line that initially sets the value of *matchroles*. The syntax is **matchrole:targetrole**, where an empty value for **matchrole** means that all users receive **targetrole**.
   b. It adds the **MYAPP2** roles for users who already have the **MYAPPSPECIAL** role. The syntax here is **matchrole1:targetrole1,matchrole2:targetrole2**. Hence, if a user has the **MYAPPSPECIAL** role, then the applications adds the **MYAPP2** role. (The leading comma follows the syntax for second and subsequent match and target roles; see the **Security.Applications.Create** method for more details.)
3. It uses the local variable that holds role escalation information to update the application's MatchRoles property. The **Security.Applications.Modify** method only updates the properties for which it has values; it leaves the others unchanged.
4. Finally, if role escalation succeeds, announce this.

The code is:

Class Member

```
Method UpdateRoles() As %Status {
 // ******************** modify application roles *********************
 write !, "All users receive the added MYAPP role."
 write !, "Users who have MYAPP2 receive MYAPPSPECIAL also."

 // Change to the %SYS namespace.
 new $NAMESPACE
 set $NAMESPACE="%SYS"

 // Add roles for the application.
 //
 // Add the MYAPP role for all users.
 set matchroles=":MYAPP"
 // Also add MYAPP2 for users who already have the MYAPPSPECIAL role.
 set matchroles=matchroles_",MYAPPSPECIAL:MYAPP2"

 // Use the matchroles variable to
 // set the applications's MatchRoles property.
 set MyAppProps("MatchRoles")=matchroles
 set status=##class(Security.Applications).Modify("/csp/MyApp",.MyAppProps)

  // Announce success.
  if $$$ISOK(status) {
    write !, "Roles were successfully modified."
 }
}
```

6.2.2 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. To use a privileged routine application:

1. Create an application definition in the Management Portal, as described in Create an Application.
2. Add classes or routines to it, as described in Edit an Application: The Routines/Classes Tab.
3. Edit the application definition's classes or routines in your development environment to escalate roles, as described in Escalate Roles in a Privileged Routine Application: The AddRoles Method.

The Portal provides the following pages to edit a privileged routine application (which includes the first two mentioned above):

- Edit a Privileged Routine Application, a Client Application, or Document Database Application: The General Tab
- Edit an Application: The Application Roles Tab
- Edit an Application: The Matching Roles Tab
- Edit an Application: The Routines/Classes Tab

6.2.2.1 Escalate 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:

ObjectScript

```
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 "Edit an Application: The Application Roles Tab") and any relevant matching roles (as described in "Edit 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 User-Defined Code.

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**:

ObjectScript

```
New $Roles
```

For more information on these topics, see Programmatically Manage Roles.

6.2.2.2 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**:

Class Member

```
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>
```

6.2.3 Client Applications

These are applications that use the client application type to connect to InterSystems IRIS.

Important: Regarding client applications:

- They are only supported on Windows. Therefore, options in the Management Portal for these applications are only available on Windows.
- For setting up their authentication, use the tools described in Authentication.

To edit a client application, the Portal provides the following pages:

- Edit a Privileged Routine Application, a Client Application, or Document Database Application: The General Tab
- Edit an Application: The Application Roles Tab
- Edit an Application: The Matching Roles Tab

6.2.4 Document Database Applications

These are applications that connect to InterSystems IRIS using the document database.

Important: For applications, use the authentication tools described in Authentication.

To edit a document database application, the Portal provides the following pages:

- Edit a Privileged Routine Application, a Client Application, or Document Database Application: The General Tab
- Edit an Application: The Application Roles Tab

- Edit an Application: The Matching Roles Tab

6.3 Create and Edit Applications

This section describes several topics:

- Create an Application
- Edit a Web Application: The General Tab
- Edit a Privileged Routine Application, a Client Application, or Document Database Application: The General Tab
- Edit an Application: The Application Roles Tab
- Edit an Application: The Matching Roles Tab
- Edit an Application: The Routines/Classes Tab
- Set the Web Application for an Interoperability-Enabled Namespace

6.3.1 Create 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**, **Client Applications**, or **Doc DB 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. 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 either:
   - Edit a Web Application: The General Tab
   - Edit a Privileged Routine Application, a Client Application, or Document Database Application: The General Tab

6.3.2 Edit a Web Application: The General Tab

To edit a web application:

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`).
2. Select an application, click **Edit**, and enter or change the information.
3. When finished with edits, restart InterSystems IRIS for the new settings to take effect.

6.3.2.1 General Settings

The initial section of the **General** tab displays various options.

Note: The fields described here are only those that are relevant for InterSystems IRIS REST-based web applications. The other type of web application, called a CSP/ZEN application, is a legacy InterSystems application type. Because new applications based on InterSystems IRIS should be REST applications, this section does not describe fields exclusively related to CSP/ZEN applications. (If you have migrated a CSP/ZEN application to InterSystems IRIS from another InterSystems product, documentation is available for the relevant fields.)

Name

An identifier for the application. The name must include a leading slash (/), such as in the /myorg/myapp application.

Note that the name `/csp/docbook` is reserved.

Description

A text description of the application.

Namespace

The namespace where this application runs. When you select a different namespace, the dialog immediately displays the default application for that namespace to the right of this drop-down menu.

Namespace Default Application

Whether or not the application is the default application for this namespace. The %System.CSP.GetDefaultApp method returns the default application for the namespace. InterSystems IRIS import functions, such as $system.OBJ.Load or $system.OBJ.ImportDir, use the default application when importing a page without an associated application.

Enable Application

Whether or not the application is available for use. When enabled, an application is available, subject to user authentication and authorization; when disabled, it is not available.

Enable REST or CSP/ZEN

Whether this is a REST application or a CSP/ZEN application, which is a type of InterSystems legacy application. For new applications, InterSystems recommends the use of **REST**, which supports modern third-party front-end technologies.

If you have an existing application that uses CSP, InterSystems supports ongoing development using that technology and provides documentation for the CSP settings in Editing a CSP Application: The General Tab.

Dispatch Class

The corresponding custom subclass of %CSP.REST for implementing a REST service. For more information, see Creating a REST Service Manually.

Redirect Empty Path

Whether the application directs empty paths to /. For example, for application `/csp/appname`, a request for `/csp/appname` will be redirected to `/csp/appname/`.

Use JWT Authentication

Whether the application supports JSON web token (JWT) authentication.

JWT Access Token Timeout

The number of seconds until the JWT expires.

JWT Refresh Token Timeout

The number of seconds until the refresh token for the JWT expires.

6.3.2.2 Security Settings

The security settings are:

Resource Required

A resource for which users must have the Use permission so they can run the application. For information on resources and permissions, see About Resources.

Group by ID

Do not use. This field is for migrated legacy applications only and documentation is available for it.

Allowed Authentication Methods

The application's supported authentication mechanisms. The options available here depend on what is selected on the **Authentication Options** page (**System Administration** > **Security** > **System Security** > **Authentication/Web Session Options**). If an application supports multiple authentication mechanisms, authentication occurs as follows:

- If more than one option is enabled *including* **Unauthenticated**, then the user can log in without providing a username and password.
- If multiple options are enabled *and* the user enters a username and password, then InterSystems IRIS attempts cascading authentication.
- If the selected options are Kerberos authentication and instance authentication (password), but **Unauthenticated** is *not* selected, then the user must provide a username and password. InterSystems IRIS attempts to perform authentication first using Kerberos and then instance authentication. If either succeeds, the user is authenticated; if both fail, the user is denied access.

For more information, see Authentication.

Permitted Classes

Do not use. This field is for migrated legacy applications only and documentation is available for it.

6.3.2.3 Session Settings

This settings in this section allow you to manage the session properties for a web application.

Important: To use these settings, you must first set the UseSession parameter of the application's dispatch class to a non-zero value; otherwise, changes in the value of any of these settings have no effect. For more information, see Creating a REST Service Manually.

The session settings are:

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 web applications during its life span, the new application does not uses its default timeout to update the session's timeout value. For example, if a session starts out in Web Application A, with a default timeout of 900 seconds, and then moves into Web Application B, which has a default timeout of 1800 seconds, the session still times out after 900 seconds.

To cause an application change to update the session timeout value, then, in a session event class, override the **OnApplicationChange** callback method to add code to update the AppTimeout property of the *%session* object.

If you disable automatic logouts for **Interoperability** pages in the InterSystems Management Portal, the session timeout does not apply to those pages. That is, the pages will not time out. Disabling automatic logouts is not recommended. For more information, see Automatic Logout Behavior in the Management Portal.

Event Class

The default name of the class (a subclass of %CSP.SessionEvents) whose methods are invoked for web application events, such as a timeout or session termination. To override this value, specify the value of the EventClass property of the %CSP.Session object, using a class name without an extension (such as .cls) as the value.

Use Cookie for Session

Whether or not the application tracks the browser session by using cookies or a URL-rewriting technique that places a value in each URL. Choices are:

- **Always** — *Default*. Always use cookies.
- **Never** — Never use cookies.
- **Autodetect** — Use cookies unless the client browser has disabled them. If the user has disabled cookies, the application uses URL rewriting.

Note that this option does not set *whether* an application uses cookies; rather, it controls *how the application manages sessions*, subject to the user's preferences. Further, even with values of **Always** or **Autodetect**, an application only uses cookies if its code is written to do so.

Session Cookie Path

The portion of the URL that the browser uses to send the session cookie back to InterSystems IRIS for this application. If you do not specify a value for this field, the application uses the value of the **Name** field with leading and following slashes as its default scope. Hence, for an application named `myapp`, specifying no value here means that /myapp/ is the scope.

The application only sends the cookie for pages within the specified scope. If you restrict the scope to pages required by a single web application, this prevents other web applications on this machine from using this session cookie; it also prevents any other web application on this web server from seeing the cookie.

Note that a primary application and its subapplications can have different security settings while simultaneously sharing a session cookie (if they all use the primary application's path).

Session Cookie Scope

Controls the default value of the *SameSite* attribute for session cookies associated with the web application. For more details, see "About the SameSite Attribute," below.

User Cookie Scope

Controls the default value of the *SameSite* attribute for user-defined cookies created with %CSP.Response.SetCookie. For more details, see "About the SameSite Attribute," below.

About the SameSite Attribute

The SameSite attribute determines how an application handles cookies in relation to third-party applications (aka *cross-site requests*).

The **Session Cookie Scope** and **User Cookie Scope** fields allow you to set SameSite for an application's cookies. **Session Cookie Scope** sets the value of the attribute for session, login, CSRF, and Group ID cookies; **User Cookie Scope** sets it for user-defined cookies.

SameSite can have a value of:

- **None** — The application sends cookies with cross-site requests. If *SameSite* has a value of **None**, browsers may require applications to use HTTPS connections.
- **Lax** — The application sends cookies with safe, top-level cross-site navigation.
- **Strict** — The application does not send cookies with cross-site requests. (The default for system web applications and new or upgraded user applications.)

To exercise more granular control over an application's cookies, use the %CSP.Response.SetCookie method, which overrides the default *SameSite* value for a particular cookie. If you use %CSP.Response.SetCookie to specify that a cookie has a SameSite value of **None**, then you must use an HTTPS connection.

The SameSite attribute is part of an initiative from the IETF and is addressed in several of their documents.

6.3.3 Edit a Privileged Routine Application, a Client Application, or Document Database Application: The General Tab

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**, **Client Applications**, or **Doc DB 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:

The name field (varies by application type)

An identifier for the application.

Description

A text 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.

6.3.4 Edit an Application: The Application Roles Tab

For web applications, privileged routine applications, or client applications, 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.

6.3.5 Edit an Application: The Matching Roles Tab

For web applications, privileged routine applications, or client applications, 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 InterSystems IRIS 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).

6.3.6 Edit 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.

6.3.7 Set the Web Application for an Interoperability–Enabled Namespace

InterSystems IRIS enables you to use a different web application for each interoperability-enabled namespace in a given instance. Consequently, you can enable different sets of users to access different interoperability-enabled namespaces within the same InterSystems IRIS instance.

To set the web application for an existing interoperability-enabled namespace, the procedure is:

1. Create a web application that is a copy of the initial web application for the namespace.

   When you create a namespace, the system creates an initial web application named `/csp/`*`namespace`*, where *namespace* is the name of the namespace.

   For instructions, see Create an Application. You can use the **Copy from** field to specify the application to copy.

2. Set the `^%SYS("Ensemble","InstalledNamespace","`*`namespace`*`")` global node to the name of the web application that you created. The *namespace* value is the name of the namespace that will use the new web application.

   For example, if you created a web application named `/csp/ensdemocopy` and you want to use the web application for the `ENSDEMO` namespace, then execute the following command in the Terminal:

   ```
   set ^%SYS("Ensemble","InstalledNamespace","ENSDEMO")="/csp/ensdemocopy"
   ```

   When a user navigates to the interoperability pages for the namespace, the new web application appears.

6.4 Built-In Applications

Each InterSystems IRIS instance comes with a number of built-in applications. This includes a group of *system applications*; there is always access to system applications, even if the **`%Service_WebGateway`** service is disabled.

Table 6–2: InterSystems IRIS Built-In Web Applications

Name	Purpose or Managed Interactions	Associated Resource	System Application
/api/atelier	REST API used by the InterSystems VS Code extensions (%Api.Atelier dispatch class).	**`%Development`**	No
/api/deepsee	REST API used by InterSystems IRIS Business Intelligence (%Api.DeepSee dispatch class).		No
/api/docdb	DocDB REST API (%Api.DocDB dispatch class).		No

Name	Purpose or Managed Interactions	Associated Resource	System Application
/api/iam	InterSystems API Manager (IAM) REST API (%Api.IAM dispatch class); used for obtaining an IAM license from InterSystems IRIS.	**%IAM**	No
/api/iknow	iKnow REST API (%Api.iKnow dispatch class).		No
/api/interop-editors	Rule Editor REST API (%Api.InteropEditors dispatch class).		No
/api/mgmnt	API Management REST API (%Api.Mgmnt dispatch class).		No
/api/monitor	Monitoring REST API (%Api.Monitor dispatch class)		No
/api/uima	UIMA REST API (%Api.UIMA dispatch class).		No
/csp/broker	Common static file store. For InterSystems internal use only.		Yes
/csp/documatic	InterSystems class reference documentation.	**%Development**	Yes
/csp/sys	General Portal access.		Yes
/csp/sys/exp	Data management options in the Portal.	**%Development**	Yes
/csp/sys/mgr	Configuration and licensing options in the Portal.	**%Admin_Manage**	Yes
/csp/sys/op	Operations options in the Portal.	**%Admin_Operate**	Yes
/csp/sys/sec	Security management and encryption options in the Portal.	**%Admin_Secure**	Yes
/csp/user	Default application for the USER namespace.		No
/isc/pki	InterSystems public key infrastructure (PKI).		Yes
/isc/studio/rules	Mapping to the CSP rules files.		Yes
/isc/studio/templates	Mapping to system-defined Studio template files.	**%Development**	Yes
/isc/studio/usertemplates	Mapping to user-defined Studio template files.		No
/oauth2	When InterSystems IRIS is configured as an OAuth 2.0 authorization server, used by that server.	**%Admin_Secure**	No
/ui/interop/rule-editor	User interface for the Rule Editor.		No

7

Using Delegated Authorization

InterSystems IRIS® data platform supports the use of user-defined authorization code. This is known as *delegated authorization*.

- Overview of Delegated Authorization
- Create Delegated (User-Defined) Authorization Code
- Configure an Instance to Use Delegated Authorization
- After Authorization — The State of the System

7.1 Overview of Delegated Authorization

Delegated authorization allows administrators to implement custom mechanisms to replace the role-assignment activities that are part of InterSystems security. For example, user-defined authorization code might look up a user's roles in an external database and provide that information to InterSystems IRIS.

To use delegated authorization, there are the following steps:

1. Create Delegated (User-Defined) Authorization Code in the **ZAUTHORIZE** routine.
2. Configuring an Instance to Use Delegated Authorization for the InterSystems IRIS 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 Using Delegated Authentication) provides support for authorization. When using **ZAUTHENTICATE**, you have the option to segregate authentication and authorization code.

Important: If using authentication with HealthShare®, you must use the **ZAUTHENTICATE** routine provided by InterSystems and cannot write your own.

7.2 Create Delegated (User-Defined) Authorization Code

Topics associated with creating delegated authorization code include:

- Start From the ZAUTHORIZE.mac Template
- ZAUTHORIZE Signature
- Authorization Code with ZAUTHORIZE
- ZAUTHORIZE Return Value and Error Messages

7.2.1 Start From the ZAUTHORIZE.mac Template

InterSystems provides a sample routine, ZAUTHORIZE.mac, that you can copy and modify. This routine is part of the Samples-Security sample on GitHub (https://github.com/intersystems/Samples-Security). You can download the entire sample as described in Downloading Samples for Use with InterSystems IRIS , but it may be more convenient to simply open the file on GitHub and copy its contents.

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 ZAUTHORIZE.mac sample. These provide important guidance about how to implement a custom version of the routine.
3. Edit your routine by adding custom authorization code and any desired code to set user account characteristics.

CAUTION: Because InterSystems IRIS places no constraints on the authorization code in **ZAUTHORIZE**, the application programmer is responsible for ensuring that this code is sufficiently secure.

Upgrade Delegated Authorization Code

Before upgrading to a new version of InterSystems IRIS, check ZAUTHORIZE.mac to determine if your current authorization code needs any changes to support new functionality.

7.2.2 ZAUTHORIZE Signature

When configured for delegated authorization, the system automatically calls **ZAUTHORIZE** after authentication occurs. InterSystems IRIS supplies values for the parameters defined in the **ZAUTHORIZE** signature as necessary. The signature of **ZAUTHORIZE** is:

ObjectScript

```
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 InterSystems IRIS, such as **%Service_Console** or **%Service_WebGateway**.

- *Namespace* — A string specifying the namespace on the InterSystems IRIS 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 InterSystems IRIS.
- *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.

7.2.3 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 InterSystems IRIS 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.

7.2.3.1 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 InterSystems IRIS.
- *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 InterSystems IRIS. (This property is described in User Account Properties.) 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 InterSystems IRIS. (This property is described in User Account Properties.) 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 InterSystems IRIS. (This property is described in User Account Properties.) 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 InterSystems IRIS, 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 InterSystems IRIS. (This property is described in User Account Properties.) 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 in to 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 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 InterSystems IRIS instance, then the user is assigned to the role.
- If a role is listed in *Properties("Roles")* and is not defined by the InterSystems IRIS 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 _PUBLIC account; for information on public resources, see Services and their resources.

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 InterSystems IRIS. (This property is described in User Account Properties.) 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 InterSystems IRIS. (This property is described in User Account Properties.) 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.

7.2.3.2 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 InterSystems IRIS.

If information in the repository changes, this information is only propagated back into the InterSystems IRIS 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**.

7.2.4 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.

7.3 Configure 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 InterSystems IRIS to invoke the **ZAUTHORIZE.mac** routine, if one exists, in the %SYS namespace.

Important: InterSystems IRIS only calls **ZAUTHORIZE** after user authentication.

7.3.1 Delegated Authorization and User Types

When a user first logs in to InterSystems IRIS 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 the Password user type; 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 About User Types.

7.4 After Authorization — The State of the System

If the user is successfully authorized, the InterSystems IRIS 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").