Authentication Guide

Version 2023.3
2024-02-19

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

Authentication Guide
InterSystems IRIS Data Platform Version 2023.3 2024-02-19
Copyright © 2024 InterSystems Corporation
All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble® InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

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

Table of Contents

List of Figures

List of Tables

1

Overview of Authentication Mechanisms

1.1 About Authentication

Authentication verifies the identity of any user or other entity attempting to connect to InterSystems IRIS®. As it's often said, authentication is how you prove that you are who you say you are.

Once authenticated, a user has established communications with InterSystems IRIS, so that its data and tools are available. Without trustworthy authentication, authorization is moot — one user can impersonate another and then take advantage of the fraudulently obtained privileges.

1.2 Authentication Mechanisms

There are a number of different ways that a user can be authenticated; each is known as an *authentication mechanism*. InterSystems IRIS supports a number of authentication mechanisms:

- Kerberos — The Kerberos protocol was designed to provide secure authentication to services over an unsecured network. Kerberos uses tickets to authenticate a user and avoids the exchange of passwords across the network.
- Operating System–Based — OS-based authentication uses the operating system's identity for each user to identify that user to InterSystems IRIS.
- Instance Authentication — With Instance authentication, InterSystems IRIS prompts the user for a password and compares a hash of the provided password against a value it has stored.
- Lightweight Directory Access Protocol (LDAP) — With the Lightweight Directory Access Protocol, InterSystems IRIS authenticates the user based on information in a central repository, known as the LDAP server.
- Delegated Authentication — Delegated authentication provides a means for creating customized authentication mechanisms. The application developer entirely controls the content of delegated authentication code.

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

Generally, if you configure InterSystems to allow unauthenticated access, it is recommended there be unauthenticated access exclusively. If there is support for an authentication mechanism and then unauthenticated access if authentication fails, this is what is called *cascading authentication*, which is described in Cascading authentication. The circumstances

for using more than one authentication mechanism are described in Use Multiple Authentication Mechanisms. InterSystems IRIS is typically configured to use only one of them.

1.3 How Authentication Works

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

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

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

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

1.3.1 About the Different Access Modes

1.3.1.1 Local Access Mode

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

Local access is available for:

- The terminal — **`%Service_Console`** on Windows and **`%Service_Terminal`** on other operating systems
- Callin — **`%Service_CallIn`**

1.3.1.2 Client-Server Access Mode

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

- ComPort — **`%Service_ComPort`**
- Java — **`%Service_Bindings`**
- JDBC — **`%Service_Bindings`**
- ODBC — **`%Service_Bindings`**

- Telnet — **%Service_Telnet**

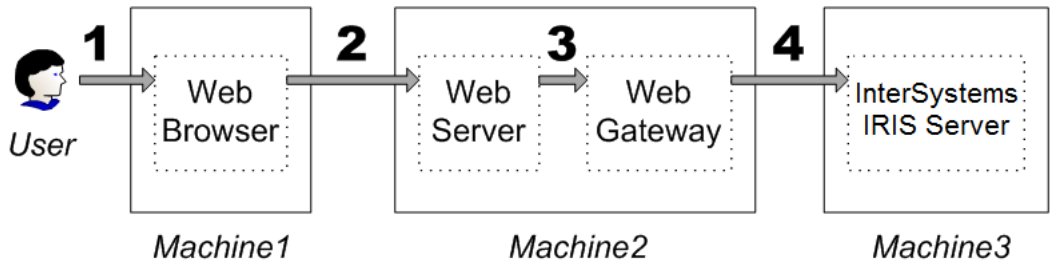
All connection tools support authentication through Kerberos or instance authentication except **%Service_ComPort**, which only supports authentication through instance authentication.

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

1.3.1.3 Web Access Mode

The web access mode supports connections of the following form:

Figure 1–1: Architecture of a Web Connection



1. A user requests content or an action in a web browser.
2. The web browser passes along the request to the web server.
3. The web server is co-located with the web gateway and passes the request to the gateway.
4. The gateway passes the request to the InterSystems IRIS server.

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

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

1.4 Overview of Setting Up Authentication

1. Choose an authentication mechanism. Your choice may be based on your authorization needs and access modes.
2. Configure authentication according to the instructions in
   - Kerberos Authentication
   - Operating System–Based Authentication
   - Instance Authentication
   - LDAP Authentication
   - Delegated Authentication
3. Optionally implement two-factor authentication.

4. Optionally implement JSON web token (JWT) authentication.

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

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

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

2

Kerberos Authentication

2.1 About Kerberos Authentication

2.1.1 Kerberos Background

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

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

Kerberos is also:

- Time-tested — Kerberos was originally developed in the late nineteen-eighties. Its principal architecture and design have been used for many years at many sites; subsequent revisions have addressed issues that have been discovered over the years.
- Available on all supported InterSystems IRIS platforms — Originally developed for UNIX®, Kerberos is available on all InterSystems IRIS-supported variants of UNIX®; Microsoft has integrated Kerberos into Windows 2000 and subsequent versions of Windows. (Note that because the Microsoft .NET framework does not include direct Kerberos support, InterSystems IRIS does not support Kerberos for the InterSystems IRIS Managed Provider for .NET.)
- Flexibly configurable — It accommodates heterogeneous networks.
- Scalable — The Kerberos protocol minimizes the number of interactions with its Key Distribution Center (KDC); this prevents such interactions from becoming a bottleneck on larger systems.
- Fast — As an open-source product, the Kerberos code has been scrutinized and optimized extensively over the years.

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

For background on Kerberos, see the MIT Kerberos website and its list of available documentation.

2.1.2 How Kerberos Works

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

Specifically, Kerberos authentication involves three separate transactions:

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

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

2.1.3 How InterSystems IRIS Uses Kerberos

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

2.2 Overview of Configuring Kerberos

To configure an InterSystems IRIS instance for Kerberos authentication:

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

   The procedure varies, depending on the operating system of the InterSystems IRIS server and the type of environment; see Preparing the Security Environment for Kerberos for more information.

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

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

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

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

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

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

6. Specify how the authentication process obtains user credentials. This is either by checking the user's Kerberos credentials cache or by providing a Kerberos password prompt for the user. See Obtain User Credentials for more information.
7. To maximally secure web connections, set up secure channels for the following connections:
   - Web browser to web server
   - Web Gateway to InterSystems IRIS server

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

2.3 About Kerberos and the Access Modes

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

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

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

Connection Tool	Access Mode	Service
InterSystems IRIS Telnet	Client-Server	**%Service_Telnet**
CallIn	Local	**%Service_CallIn**
Console	Local	**%Service_Console**
Java	Client-Server	**%Service_Bindings**
JDBC	Client-Server	**%Service_Bindings**
ODBC	Client-Server	**%Service_Bindings**
Terminal	Local	**%Service_Terminal**
Web technologies	Web	**%Service_WebGateway**

2.3.1 Local

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

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

2.3.2 Client-Server

Client-server applications include connections from Java, JDBC, ODBC, and through Telnet. For a client-server application using Kerberos authentication, the user needs credentials to interact with InterSystems IRIS via the application.

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

With client-server connections, Kerberos supports various connection security levels, which are configured on the InterSystems IRIS server machine:

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

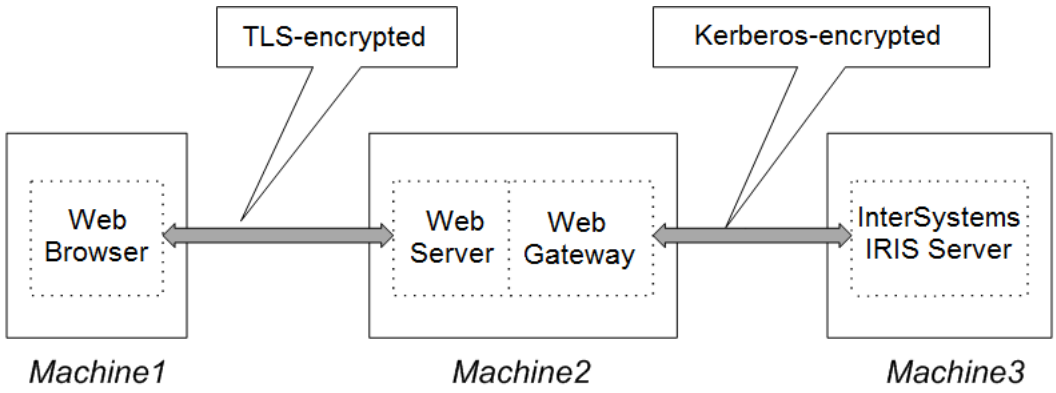
2.3.3 Web

When running a web application, the user does not interact directly with the InterSystems IRIS server. To protect all information from monitoring, you need to encrypt the connections between the user and InterSystems IRIS as follows:

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

The architecture is:

Figure 2–1: Architecture of a Kerberos-Protected Web Connection



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

Because the InterSystems IRIS server cannot prompt the end-user for a password, it invokes an API that sends HTML content to the browser to prompt. The user completes this form that has been sent; it travels back to the web server, which

hands it to the Web Gateway, which then hands it to the web server (which is part of InterSystems IRIS itself). The web server acts as a proxy on behalf of the user at the browser; this is why this kind of a connection is known as a *proxy* connection. At the same time, all information related to the user resides on the server machine (as with the local access mode); hence a web connection is also a form of in-process authentication.

2.4 Specify Connection Security Levels

Client-server connections to InterSystems IRIS use one of the following services:

- **`%Service_Bindings`** — Java, JDBC, ODBC
- **`%Service_Telnet`** — Telnet

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

1. On the **Authentication/Web Session Options** page (**System Administration** > **Security** > **System Security** > **Authentication/Web Session Options**), specify which connection security levels to enable for the entire InterSystems IRIS instance, where these can be:
   - **Kerberos** — Initial authentication only
   - **Kerberos with Packet Integrity** — Initial authentication and packet integrity
   - **Kerberos with Encryption** — Initial authentication, packet integrity, and encrypting all messages

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

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

2.5 Set Up a Client

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

2.5.1 Telnet: Set Up the Preferred Server for Use with Kerberos

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

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

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

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

   If you are adding a new server, the fields to complete are described in Define a Remote Server Connection.
6. In the dialog's Kerberos-related fields, specify values for the following fields:
   - The connection security level, where the choices are Kerberos authentication only; Kerberos authentication with packet integrity; or Kerberos authentication, packet integrity, and encryption
   - The service principal name. For information on setting up service principal names, see Names and Naming Conventions.
   - If you are configuring a telnet connection to a Windows machine, check the box specifying that the connection use the Windows InterSystems IRIS Telnet server.
7. Click **OK** to save the specified values and dismiss the dialog.

2.5.2 Set Up an ODBC DSN with Kerberos

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

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

The names of these variables must have spaces between the words. They are not case-sensitive.

- On a Windows client, you can specify connection information through a GUI: the ODBC DSN configuration dialog. InterSystems IRIS provides options on the **System DSN** tab. This screen has associated help that describes its fields. The path on the Windows Start menu to display this screen varies by version of Windows; it may be listed under **Administrative Tools**.

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

2.5.3 Set Up a Java or JDBC Client with Kerberos

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

1. To configure the client for use with Kerberos, issue the Java **Configure** command such as:

   ```
   java –classpath '$IRIS_INSTALL_DIRECTORY/dev/java/lib/JDK18/*' com.intersystems.jgss.Configure
   ```

   This allows **Configure** to run from any location on the machine, not just from within the JDK directory. Note that the specifics of this command may vary, depending on your site, such as to accommodate Windows path styles or using JDK11.

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

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

2. The program then prompts you to create and configure the krb5.conf file. If the file exists, the command prompts if you wish to use the existing krb5.conf or replace it; if you choose to replace it, it prompts for the following information:

   a. Kerberos realm — It offers the local domain in lowercase as a default value for the domain.

   b. Primary KDC — You only need include the local machine name, as the program appends the Kerberos realm name to the machine name for you.

   c. Secondary KDC(s) — You can specify the names of zero or more KDCs to replicate the content of the primary KDC.

3. After receiving this information, run the command a second time. (It instructs you to do this.)
4. When prompted to replace krb5.conf, choose to leave the existing file. The command then tests the connection by prompting for the username and password of a principal in the specified Kerberos realm.

If this succeeds, then client configuration is complete.

2.6 Obtain User Credentials

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

2.6.1 Obtain Credentials for Local Access Mode

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

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

To specify how to get credentials, the procedure is:

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

   Click **Save** to use the settings.

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

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

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

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

2.6.2 Obtain Credentials for Client-Server Access Mode

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

- ODBC and Telnet
- Java and JDBC

2.6.2.1 ODBC and Telnet

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

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

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

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

2.6.2.2 Java and JDBC

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

Note: IBM implementations of Java are available through version 8 only; for later versions, IBM supports open source versions.

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

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

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

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

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

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

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

Specifying Behavior on a Client Using the IBM Implementation

The options are:

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

Specifying Behavior on a Client Using the Oracle Implementation

The options are:

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

2.6.3 Obtain Credentials for Web Access Mode

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

2.7 Set Up a Secure Channel for a Web Connection

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

- Between the web browser and web server
- Between the Web Gateway and InterSystems IRIS

2.7.1 Set Up a Kerberized Connection from the Web Gateway to InterSystems IRIS

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

Note: For information on setting up a connection between the Web Gateway and the InterSystems IRIS server that is protected by TLS, see Configuring the Web Gateway to Connect to InterSystems IRIS Using TLS.

The procedure is:

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

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

2. Create a user in InterSystems IRIS with the same name as the Gateway's Kerberos principal. To do this, follow the instructions in Create a New User.
3. Give that user permissions to use, read, or write any required resources (these are also known as privileges). This is done by associating those privileges with a role and then associating the user with the role.
4. Configure the **`%Service_WebGateway`** service. To do this, complete the fields described in Service Properties.
5. Configure the Gateway so that it can contact the server. The procedure is:
   a. From the Management Portal home page, go to the **Web Gateway Management** page (**System Administration** > **Configuration** > **Web Gateway Management**).
   b. On the Web Gateway management page, there are a set of choices on the left. Under **Configuration**, click **Server Access**. This displays the **Server Access** page.
   c. On the **Server Access** page, you can add a new configuration or edit an existing one. To add a new configuration, click the **Add Server** button; to edit an existing one, select it from the list on the left, select the **Edit Server** radio button, and click **Submit**. This displays the page for editing or configuring server access parameters. In addition to the general parameters on this page (described on its help screen), this page allows you to specify security-related parameters for the Gateway. For Kerberos connections, these are:
      - **Connection Security Level** — Choose the kind of protection that you would like Kerberos to attempt to provide this connection. (Note that this must match or exceed the type of security specified for the web service in the previous step.)
      - **User Name** — The name of the Kerberos principal that represents the Gateway. (This must be the same principal as was used in the first step of this process.)
      - **Password** — Do not specify a value for this. (This field is used when configuring the Gateway for use with instance authentication.)
      - **Product** — InterSystems IRIS.
      - **Service Principal Name** — The name of the principal that represents the InterSystems IRIS server. This is typically a standard Kerberos principal name, of the form "iris/machine.domain", where `iris` is a fixed string indicating that the service is for InterSystems IRIS, *machine* is the machine name, and *domain* is the domain name, such as "intersystems.com".
      - **Key Table** — When connecting to an instance of InterSystems IRIS on Windows, leave this field blank; for other operating systems, provide the name of the keytab file containing the permanent key belonging to the Web Gateway, including the full path.

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

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

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

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

3

Operating System–Based Authentication

3.1 About OS-Based Authentication

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

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

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

OS-based authentication is only available for local processes, namely:

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

3.2 Configuring OS-Based Authentication

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

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

   Click **Save** to use the settings.

This type of authentication requires no other configuration actions.

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

3.3 A Note on %Service_Console

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

3.4 A Note on %Service_Callin

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

4

Instance Authentication

4.1 About Instance Authentication

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

InterSystems IRIS stores only irreversible cryptographic hashes of passwords. The hashes are calculated using the PBKDF2 algorithm with the HMAC-SHA-512 pseudorandom function, as defined in Public Key Cryptography Standard #5 v2.1: "Password-Based Cryptography Standard." The current implementation uses 10,000 iterations, 64 bits of salt, and generates 64-byte hash values; to specify a different algorithm or increase the number of iterations, use the **Security.System.PasswordHashAlgorithm** and **Security.System.PasswordHashWorkFactor** methods, respectively. There are no known techniques for recovering original passwords from these hash values.

The services available for authentication with instance authentication are:

- **%Service_Binding**
- **%Service_CallIn**
- **%Service_ComPort**
- **%Service_Console**
- **%Service_Telnet**
- **%Service_Terminal**
- **%Service_WebGateway**

4.2 Overview of Configuring Instance Authentication

For a service to use instance authentication, you must configure it as follows:

1. On the **Authentication/Web Sessions Options** page (**System Administration** > **Security** > **System Security** > **Authentication/Web Session Options**), enable authentication with instance authentication by selecting **Allow Password authentication**).
2. For the particular service, go to the **Services** page (**System Administration** > **Security** > **Services**) and select that service, such as **%Service_Bindings**, in the **Name** column; this displays the **Edit Service** page for the service.
3. On this page, choose instance authentication, listed simply as **Password** from the list of authentication types.
4. Click **Save** to save this setting.
5. In addition to this basic procedure, certain services require further configuration. This is described in the following sections:
   - Web
   - ODBC
   - Telnet

4.3 Web

For web access, you can optionally require that the Web Gateway authenticate itself to the InterSystems IRIS server through instance authentication. To perform this configuration, the procedure is:

1. From the Management Portal home page, go to the **Web Gateway Management** page (**System Administration** > **Configuration** > **Web Gateway Management**).
2. On the Web Gateway management page, there are a set of choices on the left. Under **Configuration**, click **Server Access**. This displays the **Server Access** page.
3. On the **Server Access** page, you can add a new configuration or edit an existing one. To add a new configuration, click the **Add Server** button; to edit an existing one, select it from the list on the left, select the **Edit Server** radio button, and click **Submit**. This displays the page for editing or configuring server access parameters. In addition to the general parameters on this page (described on its help screen), this page allows you to specify security-related parameters for the Gateway. For instance authentication connections, these are:
   - **Connection Security Level** — Choose **Password** from the drop-down list to use instance authentication.
   - **User Name** — The user name under which the Gateway service runs (the installation process creates the CSPSystem user for this purpose). This user (CSPSystem or any other) should have no expiration date; that is, its Expiration Date property should have a value of 0.
   - **Password** — The password associated with the user account just entered.

     Note: For the **Password** field, the Web Gateway interprets a string enclosed in braces ({}) as an operating system command, and attempts to execute it. This allows the Web Gateway to retrieve the password from a vault application or mounted secret file. See Retrieve Passwords Programmatically in Protecting Web Gateway Connections.

     This command syntax means that the password for the user which the Web Gateway uses to authenticate access to the instance cannot begin with the { character and end with the } character.
   - **Product** — InterSystems IRIS.
   - **Service Principal Name** — Do not specify a value for this. (This field is used when configuring the Gateway for use with Kerberos.)

- **Key Table** — Do not specify a value for this. (This field is used when configuring the Gateway for use with Kerberos.)

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

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

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

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

4.4 ODBC

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

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

4.5 Telnet

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

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

   If you are adding a new server, the fields to complete are described in Define a Remote Server Connection.
6. Click **OK** to save the specified values and dismiss the dialog.

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

5

Delegated Authentication

5.1 About Delegated Authentication

5.1.1 Delegated Authentication Background

InterSystems IRIS supports *delegated authentication*, which allows you to implement custom mechanisms to replace the authentication and role-management activities that are part of InterSystems security, for example, an enterprise's existing authentication system. As the application developer, you fully control the content of delegated authentication code. Delegated authentication occurs if an instance of InterSystems IRIS finds a **ZAUTHENTICATE** routine in its %SYS namespace. If such a routine exists, InterSystems IRIS uses it to authenticate users, either with calls to new or existing code. InterSystems IRIS includes a routine, ZAUTHENTICATE.mac, that serves as a template for creating the **ZAUTHENTICATE** routine.

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

5.1.2 How Delegated Authentication Works

When a user attempts to log in and InterSystems IRIS invokes delegated authentication, the sequence of events is:

1. When a service or application uses delegated authentication, a login attempt automatically results in a call to the **ZAUTHENTICATE** routine. The authentication code in this routine can be any user-defined ObjectScript, class methods, or **$ZF** callout code.
2. The next step depends on whether or not authentication succeeds and whether or not this is the first login using **ZAUTHENTICATE**:
   - If **ZAUTHENTICATE** succeeds and this is the first time that the user has been authenticated through this mechanism, the user is added to the list of InterSystems IRIS users with a type of "Delegated user". If **ZAUTHENTICATE** sets roles or other characteristics, these become part of the user's properties.
   - If **ZAUTHENTICATE** succeeds and this is not the first login, **ZAUTHENTICATE** updates the user's properties.
   - If **ZAUTHENTICATE** fails, then the user receives an access denied error and is unable to access the system. To determine why this has occurred:
     a. Check the **Reason for Failing to Login** field in the User Profile.

b. For more detailed information, check the audit log for the relevant %System/%Login/LoginFailure event. If auditing or the LoginFailure event are not enabled, you may need to enable both of these and then re-create the circumstances of the login failure.

3. If two-factor authentication is enabled for the instance and the relevant services, then there is a check that the user's PhoneNumber and PhoneProvider properties have been set. If these properties are set, then two-factor authentication proceeds; if they are not set, two-factor authentication cannot proceed and the user is not authenticated.
4. A delegated user is listed as such in the Type column of the list of users on the **Users** page (**System Administration** > **Security** > **Users**). The user's properties are displayed read-only in the Management Portal and are not editable from within InterSystems IRIS (since all the information comes from outside InterSystems IRIS).

   Note: A delegated user cannot also be an InterSystems IRIS password user.

5.2 Overview of Configuring Delegated Authentication

To use delegated authentication, the steps are:

1. Create the user-defined authentication code in the **ZAUTHENTICATE** routine. This can include the use of two-factor authentication. This routine can also perform basic setup for a user account, such as specifying roles and other user properties.

   If you are using HealthShare Health Connect, create a custom **ZAUTHENTICATE** routine as described in this guide.

   If you are using HealthShare Unified Care Record, you cannot create a custom version of **ZAUTHENTICATE** to implement delegated authentication because Unified Care Record comes with its own version of the routine. Instead, you must customize methods in the class HS.Local.ZAUTHENTICATE. For more information, see "Customizing Authentication via Local ZAUTHENTICATE" in the book *Authenticating HealthShare Users*.
2. Enable delegated authentication for the InterSystems IRIS instance on the Authentication Options page.
3. Enable delegated authentication for the relevant services, applications, or both, as required.
4. Optionally enable two-factor authentication for the InterSystems IRIS instance and, if required, for web applications and client-server applications.

For example, to use delegated authentication for an instance's Management Portal, the steps are:

1. Create the user-defined authentication code in **ZAUTHENTICATE**.
2. Enable delegated authentication for the InterSystems IRIS instance as a whole.
3. Enable delegated authentication for the set of /csp/sys* applications.

5.3 Create Delegated (User-Defined) Authentication Code

This section describes various aspects of creating your own **ZAUTHENTICATE** routine:

- Authentication Code Fundamentals
- Signature
- Authentication Code

- Set Values for Roles and Other User Characteristics
- Return Value and Error Messages

5.3.1 Authentication Code Fundamentals

InterSystems provides a sample routine, ZAUTHENTICATE.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 routine on GitHub and copy its contents.

To create your own ZAUTHENTICATE.mac:

1. To use ZAUTHENTICATE.mac as a template, copy its contents and save them into a ZAUTHENTICATE.mac routine in the %SYS namespace.
2. Review the comments in the ZAUTHENTICATE.mac sample. These provide important guidance about how to implement a custom version of the routine.
3. Edit your routine by adding custom authentication code and any desired code to set user account characteristics.

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

5.3.2 Signature

The signature of **ZAUTHENTICATE** is:

ObjectScript

```
ZAUTHENTICATE(ServiceName, Namespace, Username, Password, Credentials,
        Properties) PUBLIC {

    // authentication code
    // optional code to specify user account properties and roles
}
```

where:

- *ServiceName* — A string representing the name of the service through which the user is connecting to InterSystems IRIS, such as **`%Service_Console`** or **`%Service_WebGateway`**.
- *Namespace* — A string representing the namespace on the InterSystems IRIS server to which a connection is being established. This is for use with the **`%Service_Bindings`** service, such as with Studio or ODBC.
- *Username* — A string representing the name of the account entered by the user that is to be validated by the routine's code.
- *Password* — A string representing the password entered by the user that is to be validated.
- *Credentials* — *Passed by reference.* Not implemented in this version of InterSystems IRIS.
- *Properties* — *Passed by reference.* An array of returned values that defines characteristics of the account named by *Username*.

When InterSystems IRIS calls **ZAUTHENTICATE**, it has values for these arguments and supplies them to the routine.

Note: In older versions of InterSystems products, **ZAUTHENTICATE** took four arguments. For backwards compatibility, you can still use the four-argument version. If you are updating your code from the old to new version, note that the new arguments are second and fifth ones: *Namespace* and *Credentials*.

5.3.3 Authentication Code

The content of authentication code is application specific. If authentication succeeds, the routine should return the $$$OK macro; otherwise, it should return an error code. See Return Value and Error Messages for more information on return values.

CAUTION: Because InterSystems IRIS does not and cannot place any constraints on the authentication code in **ZAUTHENTICATE**, the application programmer is responsible for ensuring that this code is sufficiently secure.

5.3.3.1 The GetCredentials Entry Point

ZAUTHENTICATE includes an **GetCredentials** entry point. This entry point is called whenever delegated authentication is enabled for a service, and is called before the user is prompted for a username and password. Instead of getting a username and password from the user, code in the function (created by the application developer) specifies the username and password. The username and password returned are then authenticated in the normal manner as if the user entered them. A possible use of this mechanism is to provide a username and password within the entry point and then, within authentication code, to $roles for the process.

The username and password returned from this entry point can be obtained by any mechanism that the application developer chooses. They can come from a global, come from an external DLL or LDAP call, or simply be set within the routine. The application developer could even provide code to prompt for the username and password, such as in a terminal connection or with a login page.

When there is a call to the **GetCredentials** entry point, the return value and other factors determine what happens next:

- If the code sets the values of *Username* and *Password* and also returns a success status ($$$OK), then:
  - There is no additional username/password prompting.
  - The authentication process proceeds.

  Important: If the access point returns $$$OK, then its code must set the values of *Username* and *Password*. Otherwise, the user is denied access to the system and an error is written to the audit log.
- If the entry point returns the error status $SYSTEM.Status.Error($$$GetCredentialsFailed), then normal username/password prompting proceeds.
- If the entry point returns any other error status, then:
  - The user is denied access to the system.
  - The error is logged in the audit log.

In the following example of a **GetCredentials** entry point, the code performs different actions for different services:

- For **%Service_Console**, it does not prompt the user for any information and sets the process's username and password to _SYSTEM and SYS, respectively.
- For **%Service_Bindings**, it forces the user to provide a username and password.
- For web applications, it checks if the application in use is the /csp/ samples application; if it is that application, it sets the username and password to AdminUser and Test. For all other web applications, it denies access.
- For any other service, it denies access.

Finally, the **Error** entry point performs clean-up as necessary.

The code is:

ObjectScript

```
GetCredentials(ServiceName,Namespace,Username,Password,Credentials) Public {

  // For console sessions, authenticate as _SYSTEM.
  If ServiceName="%Service_Console" {
    Set Username="_SYSTEM"
    Set Password="SYS"
    Quit $SYSTEM.Status.OK()
  }

  // For a web application, authenticate as AdminUser.
  If $isobject($get(%request)) {
    If %request.Application="/csp/samples/" {
      Set Username="AdminUser"
      Set Password="Test"
      Quit $System.Status.OK()
    }
  }

  // For bindings connections, use regular prompting.
  If ServiceName="%Service_Bindings" {
    Quit $SYSTEM.Status.Error($$$GetCredentialsFailed)
  }

  // For all other connections, deny access.
  Quit $SYSTEM.Status.Error($$$AccessDenied)
}
```

For more details, see the comments for this entry point in **ZAUTHENTICATE.mac**.

5.3.3.2 The SendTwoFactorToken Entry Point

ZAUTHENTICATE includes an **SendTwoFactorToken** entry point. This entry point is for use with two-factor authentication. If it is defined and the InterSystems IRIS instance has two-factor authentication enabled, then you can override the default system setting for the format of the message and token that the instance sends to the user's mobile phone. This allows for messages that can vary by application even on the same InterSystems IRIS instance.

For more details and an example of how to use this entry point, see this entry point in the sample **ZAUTHENTICATE.mac**.

5.3.4 Set Values for Roles and Other User Characteristics

If initial authentication succeeds, **ZAUTHENTICATE** can establish the roles and other characteristics for the authenticated user. For subsequent logins, **ZAUTHENTICATE** can update these elements of the user record.

For this to happen, code in **ZAUTHENTICATE** sets the values of the *Properties* array. (*Properties* is passed by reference to **ZAUTHENTICATE**.) Typically, the source for the values being set is a repository of user information that is available to **ZAUTHENTICATE**.

5.3.4.1 User Properties

The elements in the *Properties* array are:

- *Properties("Comment")* — Any text
- *Properties("FullName")* — The first and last name of the user
- *Properties("NameSpace")* — The default namespace for a Terminal login
- *Properties("Roles")* — The comma-separated list of roles that the user holds in InterSystems IRIS
- *Properties("Routine")* — The routine that is executed for a Terminal login
- *Properties("Password")* — The user's password
- *Properties("Username")* — The user's username

- *Properties("PhoneNumber")* — The user's mobile phone number, for use with two-factor authentication
- *Properties("PhoneProvider")* — The user's mobile phone's service provider, for use with two-factor authentication

Each of these elements is described in more detail in one of the following sections.

Note: The value of each element in the properties array determines the value of its associated property for the user being authenticated. It is not possible to use only a subset of the properties or to manipulate their values after authentication.

Comment

If **ZAUTHENTICATE** sets the value of *Properties("Comment")*, then that string becomes the value of the user account's *Comment* property in 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 then holds no content.

FullName

If **ZAUTHENTICATE** sets the value of *Properties("FullName")*, then that string becomes the value of the user account's *Full name* property in 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 then holds no content.

NameSpace

If **ZAUTHENTICATE** sets the value of *Properties("Namespace")*, then that string becomes the value of the user account's *Startup Namespace* property in 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 then holds no content.

Once connected to InterSystems IRIS, the value of *Startup Namespace* (hence, that of *Properties("Namespace")*) determines the initial namespace for any user authenticated for local access (such as for Console, Terminal, or Telnet). If *Startup Namespace* has no value (since *Properties("Namespace")* has no value), then the initial namespace for any user authenticated for local access is determined as follows:

1. If the USER namespace exists, that is the initial namespace.
2. If the USER namespace does not exist, the initial namespace is the %SYS namespace.

Note: If the user does not have the appropriate privileges for the initial namespace, access is denied.

Password

If **ZAUTHENTICATE** sets the value of *Properties("Password")*, then that string becomes the value of the user account's *Password* property in 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.

Roles

If **ZAUTHENTICATE** sets the value of *Properties("Roles")*, then that string specifies the *Roles* to which a user is assigned; this value is a string containing a comma-delimited list of roles. If no value is passed back to the calling routine, then the value of *Roles* for the user account is a null string and the relevant field in the Management Portal then holds no content. Information about a user's roles is available on the **Roles** tab of a user's **Edit User** page.

If any roles returned in *Properties("Roles")* are not defined, then the user is not assigned to the role.

Hence, the logged-in user is assigned to roles as follows:

- If a role is listed in *Properties("Roles")* and is defined by the 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 **ZAUTHENTICATE** 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 for Console, Terminal, or Telnet). If *Properties("Routine")* has no value, then login starts the Terminal session in programmer mode.

Username

If **ZAUTHENTICATE** returns the *Username* property, then the value of *Username* is written to the security database after any processing in the function; this provides chance to modify the value that the user entered at the prompt. If **ZAUTHENTICATE** does not return the *Username* property, then the value of the property is written to the security database as entered.

If **ZAUTHENTICATE** 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.

If there is no explicit call that passes the value of *Properties("Username")* back to the calling routine, then there is no normalization and the value entered by the end-user at the prompt serves as the value of the user account's *Name* property without any modification.

PhoneNumber and PhoneProvider

These are properties associated with two-factor authentication.

If **ZAUTHENTICATE** sets the value of *Properties("PhoneNumber")* and *Properties("PhoneProvider")*, then these then these are written to the InterSystems IRIS database for the user as the user's mobile phone number and mobile phone service provider. If these are not passed back to the calling routine, then the phone number and service provider written to the InterSystems IRIS database are a null string. Hence, to use two-factor authentication with delegated authentication, you must supply both of these.

5.3.4.2 The User Information Repository

ZAUTHENTICATE can refer to any kind of repository of user information, such as a global or an external file. It is up to the code in the routine to set any external properties in the *Properties* array so that the authenticated user can be created or updated with this information. For example, while a repository can include information such as roles and namespaces, **ZAUTHENTICATE** code must make that information available to 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 **ZAUTHENTICATE** to perform this action. Also, if there is such code, changes to users' roles must occur in the repository; if you change a user's roles during a session, the change does not become effective until the next login, at which point the user's roles are re-set by **ZAUTHENTICATE**.

5.3.5 Return Value and Error Messages

The routine returns one of the following values:

- Success — $$$OK. This indicates that username/password combination was successfully authenticated
- Failure — $SYSTEM.Status.Error($$$ERRORMESSAGE). This indicates that authentication failed.

ZAUTHENTICATE can return system-defined or application-specific error messages. All these messages use the **Error** method of the %SYSTEM.Status class. This method is invoked as **$SYSTEM.Status.Error** and takes one or two arguments, depending on the error condition.

The available system-defined error messages are:

- **$SYSTEM.Status.Error($$$AccessDenied)** — Error message of "Access Denied"
- **$SYSTEM.Status.Error($$$InvalidUsernameOrPassword)** — Error message of "Invalid Username or Password"
- **$SYSTEM.Status.Error($$$UserNotAuthorizedOnSystem,Username)** — Error message of "User *Username* is not authorized"
- **$SYSTEM.Status.Error($$$UserAccountIsDisabled,Username)** — Error message of "User *Username* account is disabled"
- **$SYSTEM.Status.Error($$$UserInvalidUsernameOrPassword,Username)** — Error message of "User *Username* invalid name or password"
- **$SYSTEM.Status.Error($$$UserLoginTimeout)** — Error message of "Login timeout"
- **$SYSTEM.Status.Error($$$UserCTRLC)** — Error message of "Login aborted"
- **$SYSTEM.Status.Error($$$UserDoesNotExist,Username)** — Error message of "User *Username* does not exist"
- **$SYSTEM.Status.Error($$$UserInvalid,Username)** — Error message of "Username *Username* is invalid"
- **$SYSTEM.Status.Error($$$PasswordChangeRequired)** — Error message of "Password change required"
- **$SYSTEM.Status.Error($$$UserAccountIsExpired,Username)** — Error message of "User *Username* account has expired"
- **$SYSTEM.Status.Error($$$UserAccountIsInactive,Username)** — Error message of "User *Username* account is inactive"
- **$SYSTEM.Status.Error($$$UserInvalidPassword)** — Error message of "Invalid password"
- **$SYSTEM.Status.Error($$$ServiceDisabled,ServiceName)** — Error message of "Logins for Service *Servicename* are disabled"
- **$SYSTEM.Status.Error($$$ServiceLoginsDisabled)** — Error message of "Logins are disabled"
- **$SYSTEM.Status.Error($$$ServiceNotAuthorized,ServiceName)** — Error message of "User not authorized for service"

To generate a custom message, use the **$SYSTEM.Status.Error()** method, passing it the `$$$GeneralError` macro and specifying any custom text as the second argument. For example:

```
$SYSTEM.Status.Error($$$GeneralError,"Any text here")
```

Note that when an error message is returned to the caller, it is logged in the audit database (if LoginFailure event auditing is turned on). However, the only error message the user sees is $SYSTEM.Status.Error($$$AccessDenied). However, the user also sees the message for the `$$$PasswordChangeRequired` error. Return this error if you want the user to change from the current to a new password.

5.4 Set Up Delegated Authentication

Once you have created a **ZAUTHENTICATE** routine to perform authentication (and, optionally, authorization tasks), the next step is to enable it for the instance's relevant services or applications. This procedure is:

1. Enable delegated authentication for entire instance. On the **Authentication/Web Session Options** page (**System Administration** > **Security** > **System Security** > **Authentication/Web Session Options**), select **Allow Delegated authentication** and click **Save**.

   With delegated authentication enabled for the instance, a **Delegated** check box appears on the **Edit Service** page for relevant services and the **Edit Web Application** page for those applications.
2. Enable delegated authenticated for services and applications, as appropriate.

The following services support delegated authentication:

- **%Service_Bindings**
- **%Service_CallIn**
- **%Service_ComPort**
- **%Service_Console**
- **%Service_Login**
- **%Service_Terminal**
- **%Service_Telnet**
- **%Service_WebGateway**

These fall into several categories of access modes:

- Local Access —

  %Service_CallIn, **%Service_ComPort**, **%Service_Console**, **%Service_Login**, **%Service_Terminal**, **%Service_Telnet**

  To use delegated authentication with local connections, enable it for the service.
- Client-Server Access —

  %Service_Bindings

  To use delegated authentication with client-server connections, enable it for the service.
- Web Access —

  %Service_WebGateway

  To use delegated authentication with web-based connections, enable it for the web application. You may also enable it for the Web Gateway by enabling the service **%Service_WebGateway**

5.5 After Delegated Authentication Succeeds

Once the user has authenticated, two important topics are:

- The State of the System

- Change Passwords

5.5.1 The State of the System

Any user who is initially authenticated using delegated authentication is listed in the table of users on the **Users** page (**System Administration** > **Security** > **Users**) as having a type of "Delegated user". If a system administrator has explicitly created a user through the Management Portal (or using any other native InterSystems IRIS facility), that user has a type of "InterSystems IRIS password user". If a user attempts to log in using delegated authentication and is successfully authenticated, InterSystems IRIS determines that this user already exists as an InterSystems IRIS user — not a Delegated user — and so login fails.

Note: To perform sharded operations on a sharded cluster, a delegated user must have been previously authenticated on each node of the sharded cluster by some means other than an internal sharding connection. For more information on sharding, see Horizontally Scaling InterSystems IRIS with Sharding.

5.5.2 Change Passwords

The **ZAUTHENTICATE** routine also includes an entry point, **ChangePassword**, to include code to change a user's password. The signature of this entry point is:

ObjectScript

```
ChangePassword(Username,NewPassword,OldPassword,Status) Public {}
```

where

- *Username* is a string specifying the user whose password is being changed.
- *NewPassword* is a string specifying the new value of the user's password.
- *OldPassword* is a string specifying the old value of the user's password.
- *Status* (passed by reference) receives an InterSystems IRIS status value indicating either that the password change has been successful or specifying the error that caused the routine to fail.

5.6 Use LDAP with Delegated Authentication or Other Mechanisms

You can also use LDAP as part of a custom authentication system (that is, with the InterSystems IRIS delegated authentication feature). To do this, use calls to the %SYS.LDAP class as part of the custom authentication code in the **ZAUTHENTICATE** routine.

InterSystems provides a sample routine, LDAP.mac, that demonstrates these calls. This routine is part of the Samples-Security sample on GitHub (https://github.com/intersystems/Samples-Security).

Also, if you need to authenticate to LDAP or use instance authentication after collecting credentials through another mechanism, call **$SYSTEM.Security.Login** with those credentials to authenticate the user.

6

Two-Factor Authentication

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

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

This section covers the following topics:

6.1 Overview of Setting Up Two-Factor Authentication

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

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

- Changing configuration information as necessary for any existing providers (default or added).

3. Configure the service, as appropriate:
   - **%Service_Bindings** — Enable two-factor authentication for the service and continue to the next step.
   - **%Service_Console** and **%Service_Terminal** — Simply enable two-factor authentication for the service. This is all that is required.
   - **%Service_WebGateway** — There is no central means of enabling two-factor authentication for **%Service_WebGateway**. Continue to the next step.

   You can enable either or both types of authentication for each service. For more information about services, see Services.
4. Configure client-server applications and web applications, as appropriate:
   a. For client-server applications (those that use **%Service_Bindings**), add the appropriate calls into the client application to support it; this is a programming task that varies according to the client-side component in use (for example, Java, JDBC, or .NET, among others).

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

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

   Note: For the InterSystems IRIS Terminal, which uses the **%Service_Console** service on Windows and the **%Service_Terminal** service on other operating systems, there is no configuration required other than server-side setup; since InterSystems IRIS controls the prompting in these, it simply follows the standard prompt (regardless of the authentication mechanism) with the two-factor authentication prompt and processes end-user input accordingly.
5. If you are using delegated authentication, modify the **ZAUTHENTICATE.mac** routine as required. See Delegated Authentication for more information.
6. Configure each end-user to enable SMS text authentication or TOTP authentication. An end-user can be configured to use both mechanisms, but cannot have both mechanisms enabled simultaneously.

6.1.1 Two-Factor TOTP Overview

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

1. Select either an authentication device or an application that generates a TOTP, and then provide it or ensure that your users have it.
2. When you configure an end-user for two-factor TOTP authentication, the system generates a secret key, which is displayed as a base-32 encoded randomized bit string. InterSystems IRIS and the end-user share this secret key (which is why it is known as a *shared secret*). Both InterSystems IRIS and the end-user's authentication device or application use it to generate the TOTP itself, which serves as a verification code. The TOTP, which the end-user enters into a **Verification code** field or prompt, is a string of six digits, and a new one is generated at a regular interval (thirty seconds, by default).
3. At login time, after the end-user provides InterSystems IRIS with a password, InterSystems IRIS then additionally prompts for the TOTP. The end-user provides the TOTP, and then completes the login process.

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

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

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

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

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

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

Figure 6–1: A TOTP Issuer, Account, Key, and QR Code

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

6.2 Configure Two-Factor Authentication for the Server

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

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

6.2.1 Enable and Configure Two-Factor Authentication Settings for an Instance

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

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

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

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

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

6.2.2 Configure Mobile Phone Service Providers

The topics related to configuring mobile phone service providers are:

- Create or Edit a Mobile Phone Service Provider
- Delete a Mobile Phone Service Provider
- Predefined Mobile Phone Service Providers

6.2.2.1 Create or Edit a Mobile Phone Service Provider

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

1. From the Management Portal home page, go to the **Mobile Phone Service Providers** page (**System Administration** > **Security** > **Mobile Phone**):

   - To create a new provider, click **Create New Provider**.
   - To edit an existing provider, click **Edit** on the provider's row in the table of providers.

   This displays the **Edit Phone Provider** page for the selected mobile phone service provider.
2. On the **Edit Phone Provider** page, enter or change the value for each of the following fields:

   - **Service Provider** — The name of the mobile phone service provider (typically, its company name).
   - **SMS Gateway** — The address of the server that the mobile phone service provider uses to dispatch SMS (short message service) messages.

6.2.2.2 Delete a Mobile Phone Service Provider

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

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

6.2.2.3 Predefined Mobile Phone Service Providers

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

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

- Verizon — vtext.com

6.3 Enable or Disable Two-Factor Authentication for a Service

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

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

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

6.4 Configure Web Applications for Two-Factor Authentication

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

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

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

6.5 Configure an End-User for Two-Factor Authentication

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

1. From the Management Portal home page, go to the **Users** page (**System Administration** > **Security** > **Users**):
2. For an existing user, click the name of the user to edit; for a new user, begin creating the user by clicking **Create New User** (for details about creating a new user, see Create a New User). Either of these actions displays the **Edit** page for the end-user.
3. On the **Edit User** page, select **SMS text enabled** or **Time-based One-time Password enabled**, as appropriate.
4. If you select **SMS Text**, you must complete the following fields:
   - **Mobile phone service provider** — The company that provides mobile phone service for the user. Either select a provider from those listed or, if the provider does not appear in the list, click **Create new provider** to add a new provider for the InterSystems IRIS instance. (Clicking **Create a new provider** displays the **Create a New Mobile Phone Provider** window, which has fields for the **Service Provider** and the **SMS Gateway**, the purpose of which are identical to those described in Create or Edit a Mobile Phone Service Provider.)
   - **Mobile phone number** — The user's mobile phone number. This is the second factor, and is where the user receives the text message containing the one-time security token.
5. If you select **Time-based One-time Password enabled**, the page displays the following fields and information:
   - **Display Time-Based One-time Password QR Code on next login** — Whether or not to display a QR code when the user next logs in. If selected, InterSystems IRIS displays the code at the next login and prompts the user to scan it into the authentication device or application, and then to provide the displayed token to complete the authentication process. By default, this option is not selected. InterSystems recommends that you do *not* use this option.
   - **Generate a new Time-based One-time Password Key** — Creates and displays both a new shared secret for the end-user and a new QR code.

     Important: If you generate a new time-based one-time password key for a user, the current key in the user's authenticator application will no longer work. Before logging in, the user must enter the new key into the authenticator, either by scanning the QR code or by manually entering it. (This does not affect existing sessions.)
   - **Issuer** — The identifier for the InterSystems IRIS instance, which you established when configuring two-factor TOTP authentication for the instance.
   - **Account** — The identifier for the InterSystems IRIS account, which is the account's username.
   - **Base-32 Time-Based One-Time Password (OTP) Key** — The secret key that the end-user enters into the authentication device or application.
   - **QR Code** — A scannable code that contains the values of the issuer, account, and secret key.
6. Click **Save** to save these values for the user.

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

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

Otherwise, the end-user cannot authenticate:

- For SMS text authentication, the end-user must have a mobile phone and be able to receive text messages on that phone. This is the phone number at which the user receives a text message containing the one-time security token as an SMS text.

- For TOTP authentication, the user must have an authentication device or application that can either scan a QR code or that can accept the secret key and other information required to generate each TOTP (which serves as a verification code).

6.6 Configure Bindings Clients for Two-Factor Authentication

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

- Java and JDBC
- .NET
- ODBC

Client-side code performs three operations:

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

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

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

6.6.1 Java and JDBC

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

- `public boolean isTwoFactorEnabled() throws Exception`

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

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

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

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

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

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

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

6.6.2 .NET

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

- `bool IRISConnection.isTwoFactorEnabledOpen()`

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

- `void IRISConnection.sendTwoFactorToken(token)`

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

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

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

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

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

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

6.6.3 ODBC

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

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

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

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

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

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

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

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

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

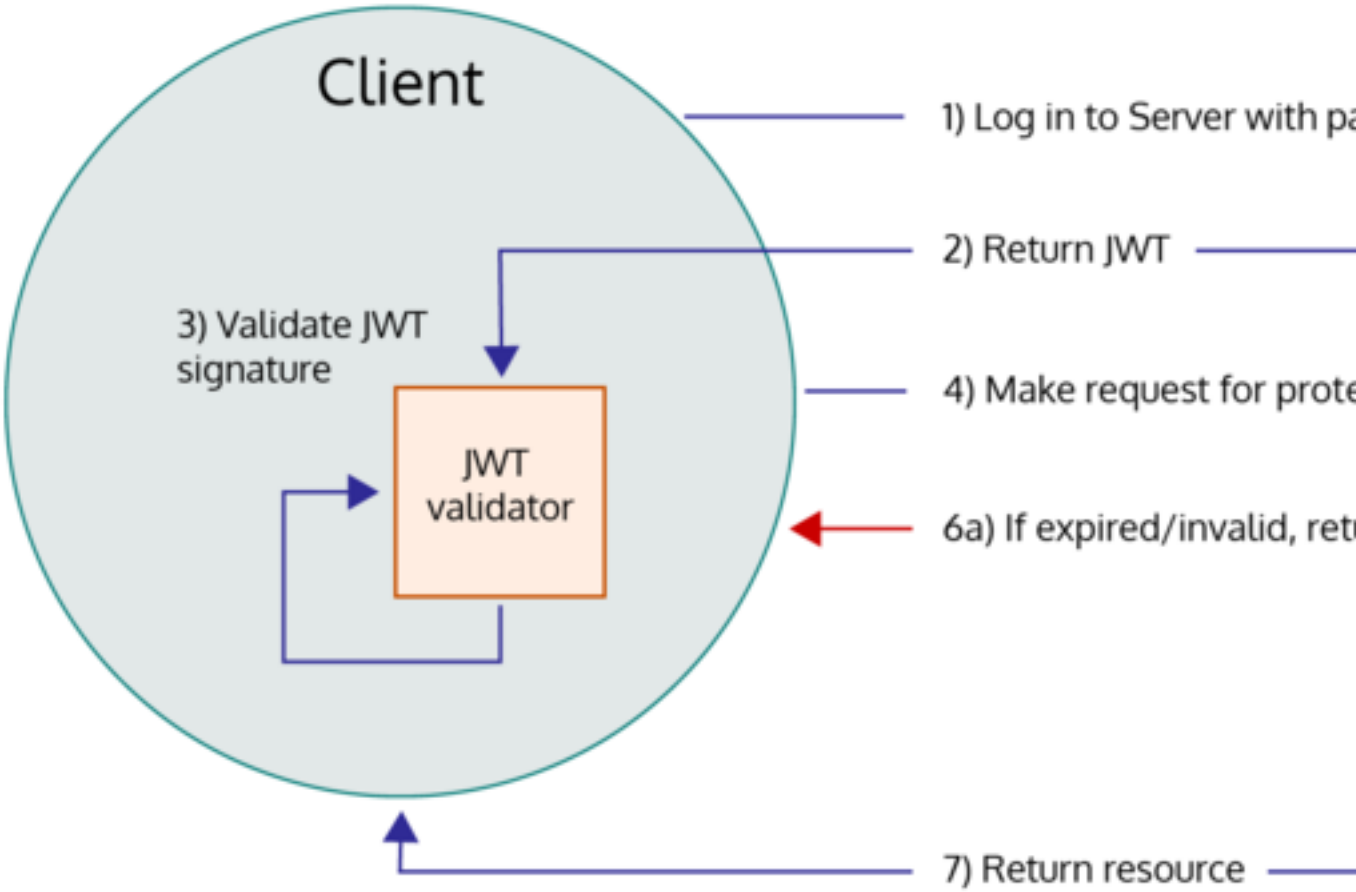
```
// Given a connection called "conn"
SQLINTEGER stringLengthPtr;
SQLINTEGER attr;
SQLRETURN rc = SQLGetConnectAttr(conn, 1002, &attr, sizeof(attr), &stringLengthPtr);
if attr {
  // Prompt the user for the two-factor authentication token.
  wstring token;
  SQLRETURN rc = SQLSetConnectAttr(conn, 1002,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

7

JSON Web Token (JWT) Authentication

7.1 Overview of JWT Authentication

A JWT is a compact, URL-safe means of authentication, authorization, or information exchange. In the case of authentication, a server provides a JWT to an already-authenticated client so that the client does not need to reprovide a password to access protected resources on the server until the JWT expires. The authentication flow can look like the following diagram:



7.2 Configuring JWT Authentication

To use JWT authentication, InterSystems IRIS must have a configuration for issuing JWTs so that the client can validate their integrity.

- The **System Administration** > **Security** > **System Security** > **Authentication/Web Session Options** page includes the following settings:
  - **JWT Issuer field** — Defines the `iss` claim in the JWT payload which identifies the issuer of the JWT to the client.
  - **JWT Signature Algorithm** — Determines the encryption algorithm that InterSystems IRIS uses for signing and validating the JWT.
  - **Reset Key Store** — Rotates the encryption keys. This invalidates all previously issued, unexpired JWTs.

Issuing JWTs is not the only configuration required. You must also configure the client receiving the JWT to accept and use them. REST web applications are among the clients that can use JWTs for authentication. The Create and Edit Applications page describes these settings.

7.3 Support for JWT Usage

InterSystems IRIS supports JWT usage for authentication to a REST API and in the OAuth 2.0 framework.

7.3.1 With a REST API

Once configured for JWT authentication, a REST API gains four endpoints that should not be included in the UrlMap in the dispatch class:

- `/login` — A call to this endpoint using basic HTTP authentication or with valid credentials in the body of the request returns an access token and a refresh token that can be used in subsequent requests.
- `/logout` — A call to this endpoint, if not using Group-By-ID, invalidates the supplied access token and the associated refresh token. If using Group-By-ID, then all sessions with the current By-ID group are invalidated.
- `/refresh` — A call to this endpoint issues a new access and refresh token pair when invoked with a valid refresh token. This invalidates the previous access and refresh token pair.
- `/revoke` — If not using Group-By-ID, this is functionally the same as `/logout`. If using Group-By-ID, this revokes only the current access and refresh token pair.

You can customize the endpoint names in the dispatch class using the following:

```
Parameter TokenLoginEndpoint = "mylogin";
Parameter TokenLogoutEndpoint = "mylogout";
Parameter TokenRevokeEndpoint = "myrevoke";
Parameter TokenRefreshEndpoint = "myrefresh";
```

7.3.1.1 Accessing REST Endpoints with a JWT

You supply the access token in HTTP requests to the REST API in a header using the format of `Authorization: Bearer ACCESS_TOKEN_HERE`. Other than supplying this access token instead of your credentials in the request, you access your web application endpoints as normal except for the `/login` and `/refresh` endpoints. To retrieve the access token, you first access the `/login` endpoint.

The `/login` Endpoint

To access the `/login` endpoint and retrieve the access and refresh tokens, make an HTTP POST request without an authentication header and with your credentials in the body in JSON format as below:

```
{"user": "YOUR USER", "password": "YOUR PASSWORD"}
```

If the credentials are valid, you receive a response similar to the following:

```
{
"access_token":
"[illegible]",
"refresh_token":
"[illegible]",
"sub": "YOUR USER",
"iat": 1682707417.749942,
"exp": 1682707477
}
```

Using the `/login` access token as an example, the `Authorization` header for requests to your other REST API endpoints has the value of:

```
Bearer
[illegible]
```

The `/refresh` Endpoint

You access the `/refresh` endpoint with an HTTP POST request without an access token. Instead, you send the following JSON-formatted data in the body of the request:

```
{
"refresh_token": "YOUR REFRESH TOKEN",
"grant_type": "refresh_token"
}
```

This returns a new access token and refresh token pair, similar to accessing the `/login` endpoint but without losing your session from a logout.

7.3.2 With OAuth2.0 and OpenID Connect

To learn more about how InterSystems IRIS supports JWTs with the OAuth 2.0 framework, see Using OAuth 2.0 and OpenID Connect.

8

Services

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

8.1 Available Services

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

There are two groups of services:

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

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

- **`%Service_Bindings`** — SQL or objects; use of Studio [resource-based]
- **`%Service_CacheDirect`** — A proprietary mechanism for connecting to other InterSystems products [resource-based]
- **`%Service_CallIn`** — The CallIn interface [resource-based]
- **`%Service_ComPort`** — COMM ports attached to a Windows system [resource-based]
- **`%Service_Console`** — The local Terminal from a Windows console (analogous to **`%Service_Terminal`** for macOS, UNIX®, and Linux) [resource-based]
- **`%Service_DataCheck`** — The DataCheck utility [basic]
- **`%Service_DocDB`** — Document database applications [resource-based]
- **`%Service_ECP`** — Enterprise Cache Protocol (ECP) [basic]
- **`%Service_Login`** — Use of **$SYSTEM.Security.Login** [resource-based]
- **`%Service_Mirror`** — InterSystems IRIS database mirroring [basic]

- **%Service_Monitor** — SNMP and remote monitor commands [basic]
- **%Service_Shadow** — Access to this instance from shadow destinations (for use only with existing configurations) [basic]
- **%Service_Sharding** — Access to this instance as a shard server [basic]
- **%Service_Telnet** — Telnet sessions on a Windows server and remote Windows Terminal sessions [resource-based]
- **%Service_Terminal** — The Terminal from a macOS, UNIX®, and Linux console (analogous to **%Service_Console** for Windows) [resource-based]
- **%Service_WebGateway** — Web application pages [resource-based]
- **%Service_Weblink** — WebLink, which is available as a legacy service [basic]

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

8.1.1 Notes on Individual Services

8.1.1.1 %Service_Bindings

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

This service also controls access to Studio. For more information about Studio and security, see Integration with InterSystems Security.

8.1.1.2 %Service_Console and %Service_Terminal

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

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

8.1.1.3 %Service_DataCheck

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

8.1.1.4 %Service_ECP

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

For details on how privileges work within an ECP-based configuration, see Distributed Cache Cluster Security.

8.1.1.5 %Service_Login

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

ObjectScript

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

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

8.1.1.6 %Service_Mirror

This service regulates the use of InterSystems IRIS database mirroring. For more details about mirroring generally, see Mirroring; for more details about security for mirroring (though the use of TLS), see Configuring InterSystems IRIS to Use TLS with Mirroring.

8.1.1.7 %Service_Sharding

This service regulates the use of an InterSystems IRIS instance as a shard data server. For more details, see Horizontally Scaling for Data Volume with Sharding.

8.1.1.8 %Service_WebGateway

This service manages connections that serve up web pages. Specifically, it manages connections between the Web Gateway processes running on the Web Server and the InterSystems IRIS server. You will not interact with this service directly within a Web Application; instead, the authentication mechanisms are configured within the relevant Web Application definition.

Under the following circumstances, there is no access to the server via the Web Gateway:

1. There are authentication mechanisms enabled for the service
2. The Web Gateway has no valid authentication information for any of the enabled authentication mechanisms
3. Unauthenticated access is disabled for the service

Hence, if you disable unauthenticated access through this service (that is, the Unauthenticated authentication mechanism is disabled), you must ensure that the Web Gateway has the information it needs to authenticate to the InterSystems IRIS server. For example, for Instance Authentication (password) access, this is a valid username-password pair; for Kerberos access, this is a valid service principal name and key table location. To specify authentication information for the Web Gateway, use its management interface; for a standard installation, the URL has the following form, using the <baseURL> for your instance:

https://<baseURL>/csp/bin/systems/module.cxw.

%Service_WebGateway controls only the background authentication between the Web Gateway processes running on the Web Server and the InterSystems IRIS instance. As a result, authentication mechanisms for Web Applications are configured and managed within the relevant Web Application definition, not by editing the allowed authentication methods of %Service_WebGateway itself.

Because **%Service_WebGateway** regulates the use of the Portal and its subapplications, disabling **%Service_WebGateway** does not disable any system applications, so that there can always be access to the Portal. For more information on system applications, see Built-In Applications.

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

8.2 Service Properties

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

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

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

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

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

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

  For more details, see Two-Factor Authentication.

  Note: If two-factor authentication is enabled for an instance, this check box appears on the **Edit Service** page for all its services. However, two-factor authentication is only available for **`%Service_Bindings`**, **`%Service_Console`**, and **`%Service_Terminal`** (and only when it is enabled for the instance).
- **Allowed Incoming Connections** — Specifies a list of IP addresses or machine names from which the service accepts connections; if a service has no associated addresses or machine names, then it accepts connections from any machine. This capability can be very useful with multi-tier configurations; for example, with the Web Gateway service, it can be used to limit the set of Web servers that can connect to InterSystems IRIS. The Allowed Incoming Connections facility for distributed cache cluster data servers has additional features, as described in Distributed Cache Cluster Security.

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

- N/A — The service has no associated resource; this means that service can simply be turned on or off.

- NO — Access is available to any user holding a role that has the Use permission on the service's resource. This is checked after authentication.
- YES — Access is available to any user.

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

8.3 Services and Authentication

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

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

Table 8–1: Services with Authentication Mechanisms

Service Name	KRB Cache	KRB Login	Del	LDAP	OS	IA	Un
%Service_Bindings	N	Y	Y	Y	N	Y	Y
%Service_CallIn	N	N	Y	Y	Y	N	Y
%Service_ComPort	N	N	Y	Y	N	Y	Y
%Service_Console	Y	Y	Y	Y	Y	Y	Y
%Service_Login	N	N	Y	Y	Y	Y	Y
%Service_Telnet	N	Y	Y	Y	N	Y	Y
%Service_Terminal	Y	Y	Y	Y	Y	Y	Y
%Service_WebGateway	N	Y	Y	Y	N	Y	Y

Key:

- KRB Cache — Kerberos Cache
- KRB Login — Kerberos Login
- Del — Delegated authentication
- LDAP — LDAP authentication
- OS — Operating system–based authentication
- IA — Instance Authentication
- Un — Unauthenticated access

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

8.4 Services and Their Resources

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

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

Independent of privileges for other resources, service privileges provide little to the user.

9

Advanced Topics in Authentication

This topic covers advanced authentication topics.

9.1 System Variables and Authentication

After authentication, two variables have values:

- *$USERNAME* contains the username
- *$ROLES* contains a comma-delimited list of the roles that the user holds

You can use the *$ROLES* variable to manage roles programmatically.

9.2 Use Multiple Authentication Mechanisms

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

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

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

9.3 Cascading Authentication

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

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

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

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

For example, if a service supports authentication through:

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

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

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

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

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

9.4 Establish Connections with the UnknownUser Account

If instance authentication and unauthenticated mode are both enabled, then a user can simply press **Enter** at the **Username** and **Password** prompts to connect to the service in unauthenticated mode, using the UnknownUser account. If only instance

authentication is enabled, then pressing **Enter** at the **Username** and **Password** prompts denies access to the service; InterSystems IRIS treats this as a user attempting to log in as the UnknownUser account and providing the wrong password.

9.5 Programmatic Logins

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

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

ObjectScript

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

where

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

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

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

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

ObjectScript

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

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

Note: The single-argument form of the **Login** method is a restricted system capability.

9.6 The JOB Command and Establishing a New User Identity

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

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

The following pseudocode illustrates how this can be done:

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

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

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

9.7 Authentication and the Management Portal

The Management Portal consists of several separate web applications. The main page of the Portal is associated with the /csp/sys application and other pages are associated with various /csp/sys/* applications (such as the security-related content, which is associated with the /csp/sys/sec application). If the applications do not all have a common set of authentication mechanism(s) in use, users going from one Portal page to another may encounter login prompts or sudden shifts in their level of privilege.

For example, if the /csp/sys application is using instance authentication exclusively, while other related Portal applications are using unauthenticated access exclusively, then, as users move from one Portal page to another, they go from unauthenticated access to requiring authentication. Another possible case is this: the /csp/sys application supports only instance authentication, the other applications support only unauthenticated access, and UnknownUser has no special privileges; in this case, when users go from the Portal's main page to its other pages, they may not have sufficient privileges to perform any action.

To check and configure the authentication mechanism for a web application, select the application from the **Web Applications** page in the Portal (**System Administration** > **Security** > **Applications** > **Web Applications**) and, for the displayed application, make selections under **Allowed Authentication Methods** as appropriate (typically, so that /csp/sys and /csp/sys/* share a common set of authentication mechanisms).