InterSystems API Index

Version 2023.3
2024-05-16

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

InterSystems API Index
InterSystems IRIS Data Platform Version 2023.3 2024-05-16

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

Tools Index

This reference is organized into topics that correspond either to a kind of item you might want to manipulate programmatically (class definitions, DDL files, and so on), a technology of interest (HTTP, XML, and so on), or a task you might be interested in (testing, debugging, and so on).

It lists the APIs for *some* tasks that are commonly performed in the Management Portal, if those are tasks you might need to perform in an installation program.

Application Management (Tools/APIs)

Work with web applications, privileged routine applications, and client applications (create, modify, export, and so on).

Background Information

In InterSystems terminology, there are three kinds of applications: web applications, privileged routine applications, and client applications.

You define applications and specify their security via the Management Portal; see Defining Applications.

Available Tools

Security.Applications class

Persistent class that contains the application definitions. This class provides method like the following:

- **Create()**
- **Delete()**
- **Export()**
- And others

It also provides the following class queries:

- **Detail()**
- **List()**

Availability: %SYS namespace.

Archiving (Tools/APIs)

Archive files to an archive server.

Available Tools

%Archive package

Enable you to archive files to an archive server. %Archive.Session is an API for data archiving; the class reference contains details and examples. %Archive.Content represents the source or target of an archive operation.

Availability: All namespaces.

Auditing (Tools/APIs)

Add entries to the audit log.

Background Information

Auditing is the process of automatically recording selected actions, including actions of the authentication and authorization systems. In InterSystems IRIS® data platform, auditing is considered a security function. The Management Portal provides pages you can use to enable auditing, configure custom audit events, and view the audit log.

See Auditing Guide.

Available Tools

%SYSTEM.Security class

Provides the **Audit()** method, which enables you to add your own entries to the audit log.

Availability: All namespaces.

%SYS.Audit class

Persistent class that contains the audit log. This class provides class queries like the following:

- **ListByEvent()**
- **ListByPid()**
- And others.

Availability: All namespaces.

^%AUDIT routine

Allows the reporting of data from the logs, and the manipulation of entries in the audit logs as well as the logs themselves. See Command-Line Security Management Utilities.

Availability: All namespaces.

^LOGDMN routine and SYS.LogDmn class

Enable you to set up *structured logging*, which will write the same messages seen in the audit database to a machine-readable file that can be ingested by your choice of monitoring tool. See Setting Up Structured Logging.

Availability: %SYS namespace.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

C (Tools/APIs)

Access InterSystems IRIS® data platform from C programs.

Background Information

C is a commonly used programming language.

Available Tools

InterSystems Callin API

Enables you to access InterSystems IRIS from C programs. You can execute ObjectScript commands and evaluate InterSystems IRIS expressions. The Callin API permits a wide variety of applications. For example, you can use it to create an interface between ObjectScript and an application that presents an integrated menu or GUI. If you gather information from an external device, such as an Automatic Teller Machine or piece of laboratory equipment, the Callin API lets you store this data in an InterSystems IRIS database. Any language that uses the C/C++ calling standard for that platform can invoke the Callin functions.

See Using the Callin API.

Availability: All namespaces.

Class Definitions (Tools/APIs)

Work with class definitions and class members programmatically (obtain information about, define, make deployed, and so on).

Background Information

For information on creating class definitions in InterSystems IRIS® data platform, see Defining and Using Classes and the Class Definition Reference. In InterSystems IRIS, a *class member* is a method, property, parameter, or other element of a class definition.

You can examine, compile, and export classes using an integrated development environment (IDE) with support for ObjectScript, such as Visual Studio Code (VS Code) with the InterSystems ObjectScript extension or Studio.

Available Tools

%Dictionary package

Provides persistent classes you can use to examine class definitions, modify class definitions, create new classes, and even write programs that automatically generate documentation. There are two parallel sets of class definition classes: those that represent defined classes and those that represent compiled classes. For example, you can use %Dictionary.ClassDefinition to work with class definitions, and you can use %Dictionary.CompiledClass to work with compiled classes.

There are similar classes in the %Library package, but the latter classes are deprecated.

See Class Definition Classes.

Availability: All namespaces.

%SYSTEM.OBJ class

Provides methods for working with class definitions and other Studio items. These include:

- **Compile()**
- **Delete()**
- **Export()**
- **ExportAllClasses()**
- **GetClassList()**
- **GetDependencies()**
- **IsUpToDate()**
- **MakeClassDeployed()**
- **ShowClasses()**
- **UnCompile()**
- **Upgrade()**
- And others

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

Concurrency Mode (Tools/APIs)

Get and set the concurrency mode for the current process.

Background Information

See the entry for Locks for background information on concurrency. Also see Object Concurrency.

The *concurrency mode* determines what type of locking is performed when you access an object.

Available Tools

%SYSTEM.OBJ class

Includes the following class methods:

- **GetConcurrencyMode()**
- **SetConcurrencyMode()**

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

CPF (Tools/APIs)

Modify the CPF programmatically (change settings such as memory and journal settings, define mappings, configure devices, and so on).

Background Information

The CPF (Configuration Parameter File) contains a set of parameters that affect how InterSystems IRIS® data platform operates. InterSystems IRIS reads this file for configuration settings when it starts up. For more information, see Introduction to the Configuration Parameter File.

The CPF can be modified programmatically through the configuration merge feature and a number of classes. Most individual settings can also be modified using the Management Portal.

Available Tools

configuration merge feature

The ISC_CPF_MERGE_FILE environment variable lets you specify a configuration merge file, which contains one or more configuration settings to be merged into the CPF with which a new instance is installed or deployed before the instance is first started. You can modify the configuration of an existing instance in similar fashion using the **iris merge** command or by setting ISC_CPF_MERGE_FILE when starting or restarting. The configuration merge feature supports the automated deployment of multiple differently-configured instances from the same source (container image or installation kit), as well as automated reconfiguration. For more information, see Automating Configuration of InterSystems IRIS with Configuration Merge.

Config package

Most of the classes in this package enable you to modify the CPF. Classes in this package include:

- Config.Databases
- Config.MapGlobals
- Config.SQL
- Config.Startup
- And others

Many of these classes are persistent and many provide class queries.

Availability: %SYS namespace.

%SYS.System class

Provides the following method:

- **GetCPFFileName()**

Availability: All namespaces.

CPUs (Processors) (Tools/APIs)

Obtain information about CPUs (processors).

Available Tools

%SYSTEM.CPU class

Holds information about available processors.

Availability: All namespaces.

%SYSTEM.Util class

Includes the **NumberOfCPUs()** class method, which you can use to discover the number of CPUs on the system.

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

CSV Files (Tools/APIs)

Work with CSV (comma-separated values) data sources.

Background Information

CSV (comma-separated values) is a simple, common format for data. For example, you can export to a .csv file from Microsoft Excel.

Available Tools

LOAD DATA SQL command

Load from CSV files into SQL tables using only InterSystems SQL. For more details, see **LOAD DATA**.

Record Mapper

Map CSV data into a production. For more details, see Using the Record Mapper.

Interactive CSV Import/Export

Import/export SQL data to or from a CSV file interactively by using the Management Portal. Use this option if you need to import or export data a single time. For more details, see Importing and Exporting SQL Data.

Current Date and Time (Tools/APIs)

Obtain the current date and time.

Background Information

ObjectScript provides a special variable (**$HOROLOG**) that you can use to obtain the current date and time; see Date and Time Values. Note that **$HOROLOG** is available within SQL queries (as are other special variables).

Available Tools

In addition, InterSystems provides the following tools:

%SYSTEM.SYS class

Includes the following class methods that return current date/time values:

- **Horolog()**
- **TimeStamp()**

Availability: All namespaces.

%Library.Utility class

Includes the following class methods that return current date/time values:

- **Date()**
- **DateTime()**
- **Time()**

Availability: All namespaces.

%Library.UTC class

Includes the following class methods that return current date/time values:

- **NowLocal()**
- **NowUTC()**

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

See Also

- Date/Time Values

Databases (Tools/APIs)

Manage database files programmatically (disable and enable journaling, copy, configure, and so on).

Background Information

InterSystems IRIS® data platform stores data and code in database files. For an introduction, see Namespaces and Databases.

Typically you create and configure databases via the Management Portal. See Configuring Databases.

Available Tools

SYS.Database class

Represents InterSystems IRIS database files, as configured in an instance of InterSystems IRIS. Properties of this class specify configuration details for that database as well as current read-only details such as size and last expansion time.

Methods in this class enable you to work with database files. These methods include:

- **Copy()**
- **DisableJournaling()**
- **EnableJournaling()**
- **GetDatabaseFreeSpace()**
- And others

This class also provides queries that provide information about databases. These queries include:

- **FreeSpace()**
- **List()**
- **RemoteDatabaseList()**
- And others

Availability: %SYS namespace.

Config.Databases class

Enables you to modify and obtain information about the [Databases] section of the CPF. (Note that you usually perform this configuration via the Management Portal, as noted above.)

The class also provides the **List()** and **MirrorDatabaseList()** class queries.

The class documentation includes examples and details.

Availability: %SYS namespace.

%Installer.Manifest class and other classes in the %Installer package

Enable you to define and use an installation manifest. Among other tasks, you can configure databases and namespaces.

Availability: All namespaces.

^DATABASE routine

This routine provides ways to manage databases; it is an alternative to using the Management Portal. For details, see ^DATABASE.

Availability: %SYS namespace.

See Also

- Namespaces

Date/Time Values (Tools/APIs)

Work with date/time values.

Available Tools

The tools are grouped into the following categories: data type classes and tools to work with date/time values.

Available Data Type Classes

Class Name	Logical Value	ODBC Type	XSD Type
%Date	A date value in **$HOROLOG** format (see Current Date and Time). For example: `46674`	DATE	date
%Time	A time value, expressed as the number of seconds past midnight. For example: `67080`	TIME	time
%TimeStamp	A date value and a time value, in ODBC format (in `YYYY-MM-DD HH:MM:SS.nnnnnnnnn`). For example: `1968-10-15 18:38:47`	TIMESTAMP	dateTime
%UTC	A UTC date and time value, in ODBC format.	TIMESTAMP	

Availability: All namespaces.

Available Tools for Working With Date/Time Values

%SYSTEM.SYS class

Includes the **TimeZone()** class method.

Availability: All namespaces.

%SYSTEM.SQL.Functions class

Includes class methods that you can use to work with date/time values. These include:

- A large set of methods that implement SQL functions to extract date parts (**DAYOFWEEK()**, **MONTHNAME()**, **YEAR()**, and so on). You can use these methods in the same way that you use other class methods; you are not restricted to an SQL context.
- Methods that implement SQL date conversion functions (**CONVERT()**, **TODATE()**, and **TOTIMESTAMP()**)
- Methods that add and subtract dates (**DATEADD()** and **DATEDIFF()**)

Availability: All namespaces.

%SYSTEM.Util class

Includes the following class methods that you can use to convert or obtain information about date/time values:

- **IsDST()**
- **LocalWithZTIMEZONEtoUTC()**
- **UTCtoLocalWithZTIMEZONE()**

Availability: All namespaces.

%Library.UTC class

Provides a set of class methods for working with %TimeStamp values. These are as follows:

- **Compare()**
- **ConvertHorologToTimeStamp()**
- **ConvertUTCtoLocal()**
- **LogicalToOdbc()**
- And others

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

See Also

- Current Date and Time

DDL (Tools/APIs)

Work with DDL statements and with DDL files.

Background Information

Generically, *DDL* means data definition language, which refers to syntax for defining data structures. SQL provides a set of statements for this purpose, and they are known collectively as SQL DDL. A *DDL file* is a text file that contains a series of these statements.

Available Tools

InterSystems SQL

InterSystems SQL includes support for SQL DDL. For information, see Defining Tables and Defining Views, which discuss DDL as one of the techniques for defining tables and views.

Availability: All namespaces.

%SYSTEM.SQL.Schema class

Includes class methods for working with DDL files, including the following:

- **ImportImport()**
- **ImportDDLDir()**
- And others

Some of these methods require specific permissions.

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

See Also

- SQL

Devices (Tools/APIs)

Work programmatically with devices such as printers; configure; query for list of devices.

Background Information

ObjectScript provides commands for working with devices. For details, see the I/O Device Guide and ObjectScript Reference.

Available Tools

In addition, InterSystems provides the following tools:

%Device class

Provides a set of class methods for working with devices. These include:

- **Broadcast()**
- **ChangePrincipal()**
- **Get()**
- **GetReadTerminators()**
- **GetReadType()**
- And others

Availability: All namespaces.

%SYS.NLS.Device class

Exposes some NLS properties of the current device and provides methods for changing those properties.

Availability: All namespaces.

Config.Devices and Config.DeviceSubTypes classes

Enable you to modify and obtain information about the [Devices] and [DeviceSubTypes] sections of the CPF. (Note that you usually modify this file via the Management Portal. See Devices and DeviceSubTypes.)

Each class also provides a class query called **List()**.

The class documentation includes examples and details.

Availability: %SYS namespace.

See Also

- Files

Directories and Drives (Tools/APIs)

Work with directories and drives programmatically.

Available Tools

%File class

Includes utility methods for working with directories and drives. These methods include:

- **CopyDir()**
- **CreateDirectory()**
- **CreateDirectoryChain()**
- **DirectoryExists()**
- **DriveListFetch()**
- **GetDirectoryPiece()**
- **GetDirectorySpace()**
- **ParentDirectoryName()**
- **SetReadOnly()**
- And others

This class also provides the following class queries:

- **DriveList()**
- **FileSet()**
- **ParseDirectory()**

Availability: All namespaces.

See Also

- Operating System Commands

Email (Tools/APIs)

Send and receive email programmatically.

Available Tools

%Net.MailMessage class and other classes in the %Net package

Support email as follows:

- InterSystems IRIS® data platform provides an object representation of MIME email messages. It supports text and non-text attachments, single-part or multipart message bodies, and headers in ASCII and non-ASCII character sets.
- You can send email via an SMTP server. SMTP (Simple Mail Transport Protocol) is the Internet standard for sending email.
- You can also retrieve email from an email server via POP3, the most common standard for retrieving email from remote servers.

For details and examples, see Using Internet Utilities.

Availability: All namespaces.

email adapters

Enable a production to send and receive email. These adapters are built on the basic support provided in the %Net package. See Using Email Adapters in Productions.

Availability: All interoperability-enabled namespaces.

Note: InterSystems IRIS does not provide an email server. Instead, it provides the ability to connect to and interact with email servers. The Management Portal uses the same API that is provided for you. Multiple parts of the Management Portal can send email automatically in the case of various events; an administrator specifies an SMTP server to use and other details as needed.

See Also

- MIME

Encryption (Tools/APIs)

Protect information against unauthorized viewing.

Background Information

Encryption is the process of using a mathematical algorithm to transform information so that it becomes unreadable. The information is then available only to those who possess the key that can be used for decryption.

Available Tools

Managed key encryption

InterSystems IRIS® data platform includes support for managed key encryption, a suite of technologies that protect data at rest.

Availability: All namespaces.

TLS

InterSystems IRIS TLS support includes the ability to encrypt communications.

Availability: All namespaces.

SOAP

InterSystems IRIS SOAP support includes the ability to encrypt and decrypt SOAP messages.

Availability: All namespaces.

XML

InterSystems IRIS XML support includes the ability to encrypt and decrypt XML documents.

Availability: All namespaces.

%SYSTEM.Encryption class

Provides methods to perform data encryption, base–64 encoding, hashing, and generation of message authentication codes. The preceding encryption tools use these methods. Methods in this class include:

- **AESCBCDecrypt()**
- **AESCBCManagedKeyDecrypt()**
- **AESGCMEncrypt()**
- **ActivateEncryptionKey()**
- **GenCryptRand()**
- **HMACSHA()**
- **RSAGetLastError()**
- And others

Availability: Some methods can be used in all namespaces. Some are available only in %SYS.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

Environment Variables (Tools/APIs)

Access the value of an environment variable.

Available Tools

%SYSTEM.Util class

Provides the **GetEnviron()** method.

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

Exporting Data (Tools/APIs)

Export data programmatically.

Background Information

The Management Portal provides two generic options for exporting data:

- The **Data Export Wizard**, which exports data from SQL tables. See Exporting Data to a Text File.
- The **Export Globals** page, which exports globals to .gof files. See Exporting Globals.

Available Tools

%SQL.Export.Mgr class

Enables you to export SQL tables to text files. For information, see %SQL.ExImData, the utility class from which %SQL.Export.Mgr inherits.

Availability: All namespaces.

%Global class

Includes the **Export()** class method.

Availability: All namespaces.

Numerous tools within InterSystems IRIS® data platform provide more specific export options, documented elsewhere.

Extents (Tools/APIs)

Work with extent definitions programmatically.

Background Information

Each persistent class has an *extent*, which consists of all saved instances of that class. See Extents.

Available Tools

%ExtentMgr.Util class

Maintains extent definitions and globals registered for use by those extents. Extent definitions most commonly originate from compiling a persistent class but can also be defined outside of any class. This class provides a public interface for deleting extent definitions and registering the extents of all managed extent classes or a single class.

In addition to the public interface implemented here, the %ExtentMgr tables are visible to SQL and can be queried directly. There are two examples implemented in this class: **GlobalUses()** and **GlobalsUsed()**. Both are public class methods that return a single result set and both are projected as stored procedures and can be invoked by dynamic SQL, embedded SQL or through a database driver. These methods are more important as examples of how the %ExtentMgr tables can be queried.

Availability: All namespaces.

Files (Tools/APIs)

Work with files programmatically (read, write, copy, rename, and so on).

Background

ObjectScript provides commands for working with files and other devices.

Available Tools

In addition, InterSystems IRIS® data platform provides the following tools:

%File class

Represents disk files *and* provides utility methods for working with files, directories, and drives. This class includes instance methods like the following:

- **Clear()**
- **Open()**
- **Rewind()**
- And others

And it also contains class methods like the following:

- **ComplexDelete()**
- **CopyFile()**
- **ManagerDirectory()**
- **NormalizeFilenameWithSpaces()**
- **Rename()**
- **SetReadOnly()**
- **TempFilename()**
- And others

Note that this class is a fairly simple wrapper around the ObjectScript file commands. For simply reading or writing to a file, it is suggested that you use the file stream classes. These open the file using the correct mode automatically in order to read or write to the file and thus are simpler to use.

Availability: All namespaces.

file adapters

Enable a production to read and write files. See Using File Adapters in Productions.

These adapters are included automatically in many specialized business host classes.

Availability: All interoperability-enabled namespaces.

See Also

- Devices

FTP (Tools/APIs)

Use FTP from within InterSystems IRIS® data platform.

Background Information

File Transfer Protocol (FTP) is a standard network protocol used to transfer files from one host to another host over the Internet or other TCP-based networks.

Available Tools

%Net.FtpSession class

Enables you to you to establish a session with an FTP server from within InterSystems IRIS. For details and examples, see Using Internet Utilities.

Availability: All namespaces.

FTP adapters

Enable a production to receive and send files between local and remote systems via the File Transfer Protocol (FTP). See Using FTP Adapters in Productions.

These adapters are included automatically in many specialized business host classes.

Availability: All interoperability-enabled namespaces.

Globals (Tools/APIs)

Manage globals programmatically (import, export, get size, set collation, configure mappings, and so on).

Background Information

InterSystems IRIS® data platform stores all data in its databases in globals. A single set of rules governs the names of globals and the names of their subscripts (that is, different languages do not have different rules). See Introduction to Globals and Using Globals.

You can define *global mappings* so that you can access data in a non-default location; see Configuring Namespaces. Typically you do this within the Management Portal.

The Management Portal also provides options for examining and managing globals. See Managing Globals.

Available Tools

The fundamental tool for working with globals is the ObjectScript language. In addition, InterSystems provides the following tools:

^$GLOBAL

This structured system variable returns information about globals.

Availability: All namespaces.

%Global class

Provides the following class methods:

- **Export()**
- **Import()**

Availability: All namespaces.

%GlobalEdit class

Enables you to see and modify properties of globals. It provides the methods like the following:

- **CheckIntegrity()**
- **CollationSet()**
- **GetGlobalSize()**
- **GetGlobalSizeBySubscript()**
- **KillRange()**
- And others

Availability: All namespaces.

%ExtentMgr.Util class

Maintains extent definitions and globals registered for use by those extents. It includes the following methods:

- **GlobalUses()**
- **GlobalsUsed()**

The %ExtentMgr tables are visible to SQL and can be queried directly. For details, see the reference for %ExtentMgr.Util.

Availability: All namespaces.

%SYS.GlobalQuery class

Provides the following queries:

- **DirectoryList()**
- **NameSpaceList()**
- **NameSpaceListChui()**
- **Size()**

Availability: All namespaces.

%Studio.Global class

Provides an interface to globals. It includes methods like the following:

- **GlobalListClose()**
- **Kill()**
- **Set()**
- And others

It also provides a couple of queries.

Availability: All namespaces.

%SYSTEM.OBJ class

Provides the following methods that you can use with globals:

- **Export()**
- **ExportToStream()**
- **Load()**

Availability: All namespaces.

Config.MapGlobals class

Enables you to modify and obtain information about the [Map.xxx] section of the CPF, which defines global mappings. (Note that you usually perform this configuration via the Management Portal, as noted above.)

The class also provides the **List()** class query.

The class documentation includes examples and details.

Availability: %SYS namespace.

%Installer.Manifest class and other classes in the %Installer package

Enable you to define and use an installation manifest. Among other tasks, you can configure global mappings.

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

GUIDs (Globally Unique Identifiers) (Tools/APIs)

Work with GUIDs (Globally Unique Identifiers).

Background Information

A GUID (globally unique identifier) is a unique reference number used as an identifier.

Available Tools

GUIDENABLED class parameter

Enables you to generate a GUID for each instance of the class. See Object Synchronization.

%ExtentMgr.GUID class

This class is a persistent class that gives you access to GUID,OID value pairs. This class can be queried using SQL. This class presents examples.

Availability: All namespaces.

%GUID class

Provides utility methods for GUIDs. These include:

- **%FindGUID()**
- **AssignGUID()**
- And others

Availability: All namespaces.

%SYSTEM.Util class

Includes the following method:

- **CreateGUID()**

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

HTTP (Tools/APIs)

Send and receive HTTP requests and responses.

Background Information

HTTP (Hypertext Transfer Protocol) is an application protocol used widely on the Internet.

Available Tools

%Net.HttpRequest and %Net.HttpResponse classes

Enable you to send HTTP requests and receive HTTP responses.

For details and examples, see Using Internet Utilities. The class reference for %Net.HttpRequest is also quite detailed.

Availability: All namespaces.

HTTP adapters

Provide an HTTP listener for custom port listening, XML listening, or raw HTML handling. The adapters support the standard HTTP operations Post, Get, and Put, and they allow the use of proxy servers. See Using HTTP Adapters in Productions.

These adapters are included automatically in many specialized business host classes.

Availability: All interoperability-enabled namespaces.

Importing Data (Tools/APIs)

Import data programmatically.

Background Information

The Management Portal provides two generic options for importing data:

- The **Data Import Wizard**, which imports data into SQL tables. See Importing Data from a Text File.
- The **Import Globals** page, which imports globals from .gof files. See Importing Globals.

You can also import data from .txt files or .csv files into SQL tables by using the **LOAD DATA** SQL command. This command also enables you to import data from JDBC sources by using a SQL Gateway Connection.

Available Tools

%SQL.Import.Mgr class

Enables you to import text files into SQL tables. For information, see %SQL.ExImData, the utility class from which %SQL.Import.Mgr inherits.

Availability: All namespaces.

%SQL.Migration.Import class

Enables you to import objects from relational databases. It includes support for data scrubbing.

Availability: All namespaces.

%SQL.Util.Procedures class

Provides the following methods, which you can use to import from CSV files:

- **CSV()**
- **CSVTOCLASS()**

Availability: All namespaces.

%Global class

Includes the **Import()** class method

Availability: All namespaces.

Numerous tools within InterSystems IRIS® data platform provide more specific import options, documented elsewhere.

See Also

- SQL

Include Files (Tools/APIs)

Export include files programmatically.

Background Information

An include file contains definitions for ObjectScript macros. See Using Macros and Include Files.

Available Tools

%SYSTEM.OBJ class

Includes the following class methods that you can use with include files:

- **Export()**
- **ExportToStream()**

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

Installation (Tools/APIs)

Create custom installers.

Available Tools

%Installer.Manifest class and other classes in the %Installer package

Enable you to define and use an installation manifest. Among other tasks, you can configure databases and namespaces.

Availability: All namespaces.

ObjectScript utility for unattended installation (Windows)

Enables you to perform unattended custom installation, upgrade, reinstallation (repair), and removal (uninstall) of instances of InterSystems IRIS® data platform on your computer.

Availability: All namespaces.

Support for extending the InterSystems IRIS distribution (on UNIX®)

Enables you to add a UNIX® install package to an existing InterSystems IRIS distribution.

Availability: All namespaces.

For information on all these tools, see the Installation Guide.

Inventory Facility (Tools/APIs)

Create a catalog of your code.

Background Information

The Inventory facility is provided to enumerate and catalog the file and routine components of an InterSystems IRIS® data platform system. Inventories are run during installation and upgrade, and can be used to identify changes in an InterSystems IRIS system over time.

InterSystems uses this facility to identify changes systematically between releases, and it is available for your use as well.

Available Tools

Inventory package

Enables you to create a catalog of your code. Inventory.Scan is a persistent class that represents the results of scanning the installation and examining its components. Other persistent classes contain additional details.

Inventory.Scanner is a utility class for initializing and manipulating inventory scans.

In advanced cases, you can customize this system to scan your own new kinds of "components" or your application code.

See the class reference for these classes and other classes in the Inventory package.

Availability: %SYS namespace.

IP Addresses (Tools/APIs)

Work with IP addresses (validate, get IP addresses, and so on).

Background Information

InterSystems IRIS® data platform always accepts IPv4 addresses and DNS forms of addressing (host names, with or without domain qualifiers). You can configure InterSystems IRIS to also accept IPv6 addresses; see IPv6 Support.

Available Tools

%NetworkAddress class

Datatype class that validates IP addresses and ports in the format IP|Port. The IP address can either be an IPV4, IPV6, or DNS name.

Availability: All namespaces.

%Function class

Provides the **IPAddresses()** method, which returns the IP addresses for a given host.

Availability: All namespaces.

%SYSTEM.INetInfo class

Provides an interface for Internet address manipulation. These interfaces support both IPV6 and IPV4 Internet addresses. Includes the following class methods:

- **BinaryAddrToText()**
- **CheckAddressExist()**
- **CheckSubnetMatch()**
- **GetInterfacesInfo()**
- **OSsupportsIPV6()**
- And others

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

JSON (Tools/APIs)

Create, use, and modify JSON-format objects and arrays; serialize objects as JSON; create objects from JSON.

Background Information

JSON (JavaScript Object Notation) is a lightweight, human-readable data interchange format, described by RFC 7159 and ECMA-404. It is commonly used in client-server communications.

Available Tools

dynamic object and dynamic array classes

A *dynamic object* is a special kind of InterSystems IRIS® data platform object that has no schema. Instead, such an object is empty, and you can create properties simply by using assignment statements. A dynamic array is similar.

Dynamic objects and arrays are defined by three classes: %DynamicObject, %DynamicArray, and %DynamicAbstractObject (the common superclass).

These classes provide methods to serialize themselves to and from JSON. For details, see Using JSON.

Availability: All namespaces.

dynamic object and dynamic array expressions

ObjectScript provides support for JSON-format object and array expressions, which return instances of %DynamicObject and %DynamicArray, respectively. For details, see Using JSON.

Availability: All namespaces.

LDAP (Tools/APIs)

Interact with an LDAP database programmatically.

Background Information

LDAP (Lightweight Directory Access Protocol) is an application protocol for accessing and maintaining distributed directory information services, including user information, which can be used for authentication.

Available Tools

You can configure InterSystems IRIS® data platform to use LDAP authentication; see LDAP Guide.

For more complex authentication requirements, InterSystems provides the following tools:

%SYS.LDAP class

Enables you to interface with an LDAP database. It provides methods you can use for authentication and for working with entries in the LDAP database.

See %SYS.LDAP.

Availability: All namespaces.

LDAP Outbound adapter

Sends requests to an LDAP server and receives responses.

See EnsLib.LDAP.Adapter.Outbound.

Availability: All interoperability-enabled namespaces.

Licenses (Tools/APIs)

Access information about license usage programmatically; configure license servers.

Background Information

See Managing InterSystems IRIS Licensing.

Available Tools

%SYSTEM.License class

Provides an interface to the InterSystems IRIS® data platform license API. This class provides methods like the following:

- **ConnectionCount()**
- **GetFeature()**
- **GetKeyStatus()**
- **GetUserLimit()**
- And others

It also provides extensive class documentation.

Availability: All namespaces.

Config.LicenseServers class

Enables you to modify and obtain information about the [LicenseServers] section of the CPF. (Note that you usually modify this file via the Management Portal. See Managing InterSystems Licensing.)

The class also provides the **List()** class query.

The class documentation includes examples and details.

Availability: %SYS namespace.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

Locks (Tools/APIs)

Read lock table information programmatically; remove locks; query and adjust lock table parameters.

Background Information

An important feature of any multi-process system is concurrency control, the ability to prevent different processes from changing a specific element of data at the same time, resulting in corruption. Thus ObjectScript and InterSystems SQL each provide commands for working with *locks*, which you use for concurrency control.

The %Persistent class provides a way to control concurrent access to objects, namely, the *concurrency* argument to **%OpenId()** and other methods of this class. These methods ultimately use the ObjectScript LOCK command. All persistent objects inherit these methods.

Internally, the in-memory *lock table* contains the current locks, along with information about the processes that hold those locks. You can use the Management Portal to view the lock table and (if necessary) remove locks; see Monitoring Locks.

For more information on locks, see Locking and Concurrency Control.

Available Tools

In addition, InterSystems provides the following tools:

^$LOCK

This structured system variable returns information about locks.

Availability: All namespaces.

^LOCKTAB routine

Enables you to view and remove locks. See Managing the Lock Table.

Availability: %SYS namespace.

%SYS.LockQuery class

Enables you to read lock table information. This class provides details and examples.

Availability: All namespaces.

SYS.Lock class

Enables you to remove locks. Also enables you to query and adjust lock table parameters. This class provides methods like the following:

- **DeleteOneLock()**
- **GetLockSpaceInfo()**
- **SetMaxLockTableSize()**
- And others

Availability: %SYS namespace.

Important: Rather than removing a lock, the best practice is to identify and then terminate the process that created the lock. Removing a lock can have a severe impact on the system, depending on the purpose of the lock.

See Also

- Concurrency Mode

Macros (Tools/APIs)

Export macros programmatically; print information about available macros.

Background Information

A macro defines a substitution in a line of ObjectScript code. For details, see Using Macros and Include Files.

Available Tools

%SYSTEM.OBJ class

Includes the following class methods that you can use with macros:

- **Export()**
- **ExportToStream()**
- **ShowMacros()**

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

Memory (Tools/APIs)

Modify the memory settings programmatically.

Background Information

You typically modify memory settings via the Management Portal. See Configuring System Information and Advanced Memory Settings.

Available Tools

%SYSTEM.Config class

Includes the following methods:

- **ModifyZFSize()**
- **ModifyZFString()**
- **Modifybbsiz()**
- **Modifynetjob()**

Availability: All namespaces.

%SYSTEM.Config.SharedMemoryHeap class

Provides an interface to return the amount of shared memory heap (gmheap), used by the system. It also provides an API to get available shared memory heap and recommended shared memory heap parameters for configuration. It provides methods like the following:

- **FreeCount()**
- **GetUsageSummary()**
- **RecommendedSize()**
- And others

It also provides a couple of class queries.

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

messages.log (Tools/APIs)

Write to the messages.log file, the operator messages log.

Background Information

InterSystems IRIS® data platform reports general messages, system errors, certain operating system errors, and network errors through an operator console facility. On Windows, there is no operator console as such and all console messages are sent to the messages.log file in the InterSystems IRIS system manager directory (*install-dir*`/mgr`). For InterSystems IRIS systems on UNIX® platforms, you can send operator console messages to the messages.log file or the console terminal.

For more information, see Monitoring Log Files.

The directory location of this file is configurable in the CPF file; see ConsoleFile.

Available Tools

%SYS.System class

Provides the **WriteToConsoleLog()** method, which you can use to write to the messages.log file.

Availability: All namespaces.

%SYSTEM.Config class

Provides the **ModifyConsoleFile()** method.

Availability: All namespaces.

^LOGDMN routine and SYS.LogDmn class

Enable you to set up *structured logging*, which will write the same messages seen in messages.log to a machine-readable file that can be ingested by your choice of monitoring tool. See Setting Up Structured Logging.

Availability: %SYS namespace.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

MIME (Tools/APIs)

Send and receive MIME messages.

Background Information

MIME (Multipurpose Internet Mail Extensions) is a standard for email and for other content exchanged over the Internet.

Available Tools

%Net.MIMEPart class and other classes in the %Net package

Enable you to create and send MIME messages from within InterSystems IRIS® data platform. For details and examples, see Using Internet Utilities.

Availability: All namespaces.

See Also

- Email

MQ (IBM WebSphere MQ) (Tools/APIs)

Exchange messages between InterSystems IRIS® data platform and IBM WebSphere MQ.

Background Information

IBM WebSphere MQ is a third-party product for transmitting messages.

Available Tools

%Net.MQSend class and other classes in the %Net package

Define an interface to IBM WebSphere MQ, which you can use to exchange messages between InterSystems IRIS and the message queues of IBM WebSphere MQ. For details and examples, see Using Internet Utilities.

Availability: All namespaces.

MQSeries adapters

Enable a production to receive and send messages in IBM WebSphere MQ (MQ Series) format. Message content can be a specific data type or a binary data stream. The adapters can simply send the message, or send it and then pull the corresponding response from the message queue.

See Using IBM WebSphere MQ Adapters in Productions.

Availability: All interoperability-enabled namespaces.

Namespaces (Tools/APIs)

Get information about namespaces programmatically; query for list of namespaces.

Background Information

In InterSystems IRIS® data platform, any code runs within a namespace. A namespace provides access to data and to code, which is stored (typically) in multiple database files. For an introduction, see Namespaces and Databases.

Typically you create and configure namespaces via the Management Portal. See Configuring Namespaces.

Available Tools

%SYS.Namespace class

Provides the following class methods:

- **Exists()**
- **GetGlobalDest()**
- **GetRoutineDest()**

This class also provides the following query:

- **List()**

Availability: All namespaces.

%SYSTEM.SYS class

Includes the following class method:

- NameSpace()

Availability: All namespaces.

Config.Namespaces class

Enables you to modify and obtain information about the [Namespaces] section of the CPF. (Note that you usually perform this configuration via the Management Portal, as noted above.)

The class also provides the **List()** class query. The class documentation includes examples and details.

Availability: %SYS namespace.

%Installer.Manifest class and other classes in the %Installer package

Enable you to define and use an installation manifest. Among other tasks, you can configure namespaces.

Availability: All namespaces.

%Library.EnsembleMgr class

Provides the **EnableNamespace()** method, which you can use to enable a namespace that supports interoperability productions. This is useful if you create namespaces programmatically.

Do not use this method to repair a damaged namespace. In the event of a damaged namespace, contact the InterSystems Worldwide Response Center (WRC) for assistance.

Ignore all other methods in this class.

Availability: %SYS namespace.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

See Also

- Databases

Operating System (Tools/APIs)

Obtain information about the operating system.

Available Tools

%SYSTEM.Version class

Includes the following class methods:

- **GetOS()**
- **GetPlatform()**
- **Is64Bits()**
- **IsBigEndian()**
- **IsUnicode()**
- And others

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

Operating System Commands (Tools/APIs)

Invoke operating system commands from within InterSystems IRIS® data platform.

Available Tools

InterSystems Callout interface

Enables you to invoke executables, operating system commands, and custom written dynamic link libraries from within InterSystems IRIS. See Using the Callout Gateway.

Availability: All namespaces.

Pipe adapters

Enable a production to execute a shell command and communicate with it via pipes. Capable of handling character data or a binary data stream.

See EnsLib.Pipe.InboundAdapter and EnsLib.Pipe.OutboundAdapter.

Availability: All interoperability-enabled namespaces.

Packages (Tools/APIs)

Work with packages programmatically (compile, export, delete, and so on); configure mappings.

Background Information

A package is the initial part of a full class name. Packages group related classes so that you can more easily avoid name conflicts; there are other benefits as well. For more information, see Packages.

You can define *package mappings* so that you can access code in a non-default location; see Configuring Namespaces. Typically you do this within the Management Portal.

Available Tools

%SYSTEM.OBJ class

Provides the following methods that you can use with packages:

- **CompilePackage()**
- **DeletePackage()**
- **Export()**
- **ExportJavaPackage()**
- **ExportPackage()**
- **ExportPackageToStream()**
- **ExportToStream()**
- **GetPackageList()**

Availability: All namespaces.

%Studio.Package class

Represents the package information used by the class compiler. This class provides the following methods:

- **Exists()**
- **LockItem()**

Availability: All namespaces.

Config.MapPackages class

Enables you to modify and obtain information about the [Map.xxx] section of the CPF, which defines package mappings. (Note that you usually perform this configuration via the Management Portal, as noted above.)

The class also provides the **List()** and **ListPackages()** class queries.

The class documentation includes examples and details.

Availability: %SYS namespace.

%Installer.Manifest class and other classes in the %Installer package

Enable you to define and use an installation manifest. Among other tasks, you can configure package mappings.

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

Processes (Jobs) (Tools/APIs)

Get information about and manipulate CPU processes (also known as jobs).

Background Information

A CPU process is an instance of an InterSystems IRIS® data platform virtual machine running on an InterSystems IRIS server. Every active process has a unique job number. You typically use the Management Portal to view the process list and, if necessary, suspend, resume, or terminate them; see Controlling InterSystems IRIS Processes.

Note: Within productions, a CPU process is called a *job*, to avoid confusion with the term *business processes*, which are frequently referred to simply as *processes*.

Available Tools

^$JOB

This structured system variable returns information about processes.

Availability: All namespaces.

%SYSTEM.Process class

Allows manipulation and display of the current process. This class provides class methods like the following:

- **BatchFlag()**
- **CallingRoutine()**
- **ExceptionLog()**
- **GetCPUTime()**
- **NodeNameInPid()**
- **PrivateGlobalLocation()**
- **TruncateOverflow()**
- And others

Some of the class methods have restrictions on where they may be called.

Availability: All namespaces.

%SYSTEM.SYS class

Provides the following class methods:

- **ProcessID()**
- **MaxLocalLength()**

Availability: All namespaces.

%SYSTEM.Util class

Provides the following class methods:

- **JobPrio()**
- **GetPrio()**

- **SetPrio()**

Availability: All namespaces.

%SYS.ProcessQuery class

Enables you to display and manipulate InterSystems IRIS processes. This class provides properties that you can set to modify a process, as well as read-only properties that provide information about its current state. Properties of this class include:

- ClientExecutableName
- CurrentDevice
- JobType
- LastGlobalReference
- Priority
- Routine
- UserName
- And others

It also provides the following class methods:

- **GetCPUTime()**
- **KillAllPrivateGlobals()**
- **NextProcess()**
- And others

It also provides queries, which include:

- **AllFields()**
- **CONTROLPANEL()**
- **JOBEXAM()**
- And others

Availability: All namespaces.

SYS.Process class

Provides instance methods which operate on a process instance as well as class methods for use by managers. This class provides the following methods:

- **ProcessTableSize()**
- **ReleaseAllLocks()**
- **Resume()**
- **Suspend()**
- **Terminate()**

This class extends %SYS.ProcessQuery and thus also includes the properties, methods, and queries of that class.

Availability: %SYS namespace.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

Productions (Tools/APIs)

(Production-enabled namespaces) Work with productions programmatically (start, check status, enable configuration items, stop, and so on).

Background Information

A production is a specialized package of software and documentation that solves a specific integration problem for an enterprise customer. See Developing Productions.

You create and compile productions in the Management Portal (or in your IDE). Typically you also start, configure, and stop productions in the Management Portal.

Available Tools

Ens.Director class

Provides a large set of methods that you can use to start, stop, and otherwise control productions programmatically. These methods include:

- **EnableConfigItem()**
- **GetHostSettings()**
- **GetProductionStatus()**
- **ProductionNeedsUpdate()**
- **StartProduction()**
- And others

Availability: All interoperability-enabled namespaces.

%SYS.Ensemble class

Provides the following methods for working with productions:

- **CreateDocumentation()**
- **GetEnsMetrics()**
- **StartProduction()**
- **StopProduction()**

Availability: All namespaces.

Python (Tools/APIs)

InterSystems IRIS® data platform gives you options to use Python in different ways, depending on your needs.

Embedded Python

Embedded Python allows you to use Python as a fully supported alternative to InterSystems ObjectScript for creating InterSystems IRIS applications. When you write code in Embedded Python, it is compiled into object code that is interchangeable with compiled ObjectScript code. This allows for tighter integration than is possible with the InterSystems Python External Server or the Native SDK for Python.

The Python External Server

The Python External Server allows your ObjectScript code to generate a proxy object that controls a corresponding Python target object. You can then access methods and properties just as if you were using the Python object directly.

The Native SDK for Python

The Native SDK for Python is a lightweight interface that allows you to access many InterSystems IRIS features directly from your Python application. You can:

- Call ObjectScript class methods or functions from your Python application as easily as you can call native Python methods.
- Generate a proxy object in your Python application that creates and controls an ObjectScript object on the server. The Python proxy can call instance methods and get or set properties just as if you were using the server object directly.
- Work with multidimensional global arrays (known in InterSystems IRIS simply as globals). You can create, read, change, and delete globals just as if you were using ObjectScript.

The Production EXtension (PEX)

The Production EXtension (PEX) framework allows you to use Python to develop custom components of an interoperability production, including adapters and business hosts. For productions, PEX is the standard way to interoperate with Python.

The Python ODBC Bridge (pyodbc)

The Python ODBC Bridge (pyodbc) allows you to use ODBC to connect to InterSystems IRIS by implementing the DB API 2.0 specification (PEP 249—Python Database API Specification v2.0), leveraging ODBC to access the underlying database.

Regular Expressions (Tools/APIs)

Perform pattern matching using regular expressions.

Background Information

Regular expressions are a short and flexible way to match strings, so that you can locate particular characters, words, or patterns of characters.

Available Tools

$LOCATE ObjectScript function

Returns the position of the first match of a regular expression in a string. See $LOCATE and Regular Expressions.

$MATCH ObjectScript function

Returns a boolean value depending on whether a string matches a regular expression. See $MATCH and Regular Expressions.

%Regex.Matcher class

Creates an object that does pattern matching using regular expressions. The regular expressions come from the International Components for Unicode (ICU). The ICU maintains web pages at http://www.icu-project.org.

For details, see Regular Expressions.

Availability: All namespaces.

Notes

The ObjectScript pattern matching operator is also quite flexible, but does not provide the full range of syntax given by regular expressions. See the Pattern Matching reference page.

Routines (Tools/APIs)

Work with routines programmatically (create, compile, get time stamp, export, and so on); configure mappings.

Background Information

You can create routines in ObjectScript. For information, see:

- Orientation Guide for Server-Side Programming
- Using ObjectScript

You can define *routine mappings* so that you can access code in a non-default location; see Configuring Namespaces. Typically you do this within the Management Portal.

Available Tools

^$ROUTINE

This structured system variable returns information about routines.

Availability: All namespaces.

%Routine class

Enables you to read, create, manipulate, save, and compile routines. This class provides methods such as the following:

- **CheckProtect()**
- **CheckSyntax()**
- **Compile()**
- **GetCurrentTimeStamp()**
- **Lock()**
- **Rewind()**
- **RoutineExists()**
- And others

It also provides the following queries:

- **Compare()**
- **Find()**
- **RoutineList()**
- **RoutineSortByField()**

Availability: All namespaces.

%RoutineIndex class

Index for all the routines in this namespace.

Availability: All namespaces.

%SYSTEM.OBJ class

Includes the following class methods that you can use with routines:

- **CompileList()**
- **Export()**
- **ExportToStream()**
- **Load()**

Availability: All namespaces.

Config.MapRoutines class

Enables you to modify and obtain information about the [Map.xxx] section of the CPF, which defines routine mappings. (Note that you usually perform this configuration via the Management Portal, as noted above.)

The class also provides the **List()** class query.

The class documentation includes examples and details.

Availability: %SYS namespace.

%Installer.Manifest class and other classes in the %Installer package

Enable you to define and use an installation manifest. Among other tasks, you can configure routine mappings.

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

SASL (Tools/APIs)

Implement SASL to include authentication in connection-based protocols.

Background Information

SASL is the Simple Authentication and Security Layer as defined by RFC 2222. It is a method for adding authentication support to connection-based protocols.

Available Tools

%Net.Authenticator class

Implements SASL. This class will pick a security mechanism (e.g. CRAM-MD5) from a list defined by the user of this class based on server options. The selected security mechanism will use its challenge-response mechanism to authenticate this client with the selected server.

Availability: All namespaces.

%Net.SASL package

Implements security mechanisms for SASL for use with the preceding class. For example, %Net.SASL.CRAMMD5 implements the CRAM-MD5 SASL mechanism.

Availability: All namespaces.

Security Items (Tools/APIs)

Work with roles, resources, applications, TLS configurations, and other security items programmatically (create, manipulate, export, and so on).

Background Information

For an introduction to security in InterSystems IRIS® data platform, see About InterSystems Security.

Typically you create and modify security items via the Management Portal.

Available Tools

%SYSTEM.Security class

Provides security-related utility methods. These are as follows:

- **AddRoles()**
- **Audit()**
- **ChangePassword()**
- **Check()**
- **GetGlobalPermission()**
- **Login()**
- **ValidatePassword()**

Availability: All namespaces.

Security package

Provides classes you can use to define and manipulate security items programmatically. Typically, you would use these to define resources, roles, and possibly starter user IDs as part of installation. Classes in this package include:

- Security.Applications
- Security.Events
- Security.Resources
- Security.SQLPrivileges
- Security.SSLConfigs
- And others

Availability: %SYS namespace.

Security routines

InterSystems provides several routines that you can use as an alternative to the Management Portal. See Command-Line Security Management Utilities.

Availability: Most are available only in the %SYS namespace.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of ##class(%SYSTEM.class).method(), you can use $SYSTEM.class.method().

See Also

- Auditing

Server (Tools/APIs)

Obtain information about the InterSystems IRIS® data platform server and its environment.

Background Information

InterSystems IRIS is an application server, which works in conjunction with a web server and with the Web Gateway. For information on the Web Gateway and on supported web servers, see Introduction to the Web Gateway.

This topic discusses the application server, as opposed to the third-party web server.

Available Tools

%SYS.System class

Provides methods like the following:

- **GetInstanceName()**
- **GetUniqueInstanceName()**
- **GetNodeName()**
- **TempDirectory()**
- And others

Availability: All namespaces.

%SYSTEM.Util class

Provides methods like the following:

- **BinaryDirectory()**
- **InstallDirectory()**
- **ManagerDirectory()**
- And others

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

SQL (Tools/APIs)

Use SQL within InterSystems IRIS® data platform; access third-party ODBC- or JDBC-compliant databases; access InterSystems IRIS as an ODBC- or JDBC-compliant database.

Background

SQL (Structured Query Language) is a programming language designed for managing data in relational database management systems (RDBMS).

Available Tools

InterSystems SQL

InterSystems IRIS provides an implementation of SQL, known as InterSystems SQL, which you can use within various programmatic contexts.

Also, you can execute InterSystems SQL directly within the SQL Shell (in the Terminal) and in the Management Portal. Each of these includes an option to view the query plan, which can help you identify ways to make a query more efficient.

For an introduction, see Using InterSystems SQL.

For reference information, see the InterSystems SQL Reference.

Availability: All namespaces.

%SYSTEM.SQL class and classes in the %SYSTEM.SQL package

Includes methods related to InterSystems SQL. These include methods that do the following tasks:

- Implement SQL functions
- Check SQL privileges
- Purge cached queries
- Access the *%ROWID* and *SQLCODE* variables
- Import DDL files
- Launch SQL shells
- Modify SQL configuration settings
- Modify SQL Gateway connections
- And others

Availability: All namespaces.

%SQL.Migration.Util class

Provides utilities for SQL migration. This class provides methods like the following;

- **CopyOneView()**
- **DSNFetch()**
- **DropTable()**
- **ExecSql()**

- And others

It also provides several class queries.

Availability: All namespaces.

InterSystems SQL Gateway

Enables your applications to access third-party relational databases (via ODBC or JDBC). Using the SQL Gateway, applications can:

- Access data stored in third-party relational databases within InterSystems IRIS applications using objects and/or SQL queries.
- Store persistent InterSystems IRIS objects in external relational databases.

For details, see Using the InterSystems SQL Gateway.

Availability: All namespaces.

InterSystems ODBC driver

Enables you to access InterSystems IRIS as a ODBC-compliant database. See Using the InterSystems ODBC Driver.

Availability: All namespaces.

InterSystems JDBC driver

Enables you to access InterSystems IRIS as a JDBC-compliant database. See Using Java with the InterSystems JDBC Driver.

Availability: All namespaces.

Config.SQL, Config.SqlSysDatatypes, and Config.SqlUserDatatypes classes

Enable you to modify and obtain information about the [SQL], [SqlSysDatatypes], and [SqlUserDatatypes] sections of the CPF. (Note that you usually modify this file via the Management Portal. See the [SQL] section of Configuration Parameter Reference File.)

Config.SqlSysDatatypes and Config.SqlUserDatatypes each provide the **List()** class query.

The class documentation includes examples and details.

Availability: %SYS namespace.

SQL Adapters

Enable productions to execute SQL statements against a remote database via an ODBC-defined or JDBC-defined Data Source Name (DSN). See Using SQL Adapters in Productions.

Availability: All interoperability-enabled namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

See Also

- DDL Files

- SQL Gateway Connections

SQL Gateway Connections (Tools/APIs)

Manage and access SQL Gateway connections programmatically (check connections, query by name, and so on).

Background Information

The SQL Gateway allows InterSystems IRIS® data platform to access external databases via both JDBC and ODBC. For a detailed description of the SQL Gateway, see Using the InterSystems SQL Gateway.

An SQL Gateway connection contains information about accessing a specific external database via JDBC or via ODBC. You generally define these connections in the Management Portal.

Available Tools

%SYSTEM.SQLGateway class

Provides an interface for managing SQL Gateway connections. This class provides methods like the following:

- **DropConnection()**
- **GetJDBCConnection()**
- **GetODBCConnection()**
- **SetAutoCommit()**
- **Test()**
- And others

Availability: All namespaces.

%SQLConnection class

Stores SQL Gateway connections. This class provides the following methods:

- **ConnExists()**
- **setEncode()**

It also provides the following class queries:

- ByConnection()
- ByName()

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

SSH (Tools/APIs)

Use SSH to communicate securely.

Background Information

SSH (Secure Shell) is a network protocol for secure communication over a network.

It is more common to use TLS, however.

Available Tools

%Net.SSH.Session and %Net.SSH.SFTP classes

Enable you to communicate securely via SSH. You can perform SCP (Secure Copy) operations of single files to and from the remote system. You can also execute remote commands, tunnel TCP traffic, and perform SFTP operations.

See Using SSH

Availability: All namespaces.

Startup and Shutdown Behavior (Tools/APIs)

Customize startup and shutdown behavior.

Available Tools

Startup and shutdown behavior is a broad topic. The following list indicates tools you can use to programmatically customize startup and shutdown behavior of the system as a whole:

Config.Startup class

Enables you to modify and access information about the [Startup] section of the CPF. (Note that you usually modify this file via the Management Portal. See Memory and Startup Settings.)

The class documentation includes examples and details.

Availability: %SYS namespace.

^ZWELCOME routine

InterSystems reserves this routine name for your use; the routine is not predefined. The **^ZWELCOME** routine is intended to contain custom code to execute when the Terminal starts. See Using the ObjectScript Shell.

Availability: Affects only the namespace in which it is defined (by default).

^%ZSTART routine

InterSystems reserves this routine name for your use; the routine is not predefined. The **^%ZSTART** routine is intended to contain custom code to execute when certain events happen, such as when a user logs in. If you define this routine, the system calls it when these events happen. See Customizing Start and Stop Behavior with ^%ZSTART and ^%ZSTOP Routines.

Availability: Affects all namespaces.

^%ZSTOP routine

InterSystems reserves this routine name for your use; the routine is not predefined. The **^%ZSTOP** routine is intended to contain custom code to execute when certain events happen. See the comments for **^%ZSTART**.

Availability: Affects all namespaces.

In addition, many classes that you use provide callback methods that enable you to customize startup and shutdown behavior of the code. For example, you can customize callback methods to control startup and shutdown behavior of productions.

Tasks (Tools/APIs)

Work with tasks (Task Manager) programmatically (schedule, export definitions, query, and so on).

Background Information

A *task* is a unit of work that you schedule via the Task Manager; see Using the Task Manager.

Available Tools

%SYS.Task class and classes in the %SYS.Task package

These classes define an API to schedule tasks to run in the background. They include methods like the following:

- **AssignSettings()**
- **ExportTasks()**
- **Resume()**
- **RunNow()**
- And others

They also inherits class queries from the %SYS.TaskSuper class, such as:

- **QuickTaskList()**
- **SuspendedTasks()**
- **TaskList()**
- And others

Availability: All namespaces.

TCP/IP (Tools/APIs)

Communicate via TCP/IP; work with TCP devices.

Background Information

TCP/IP is the common name for a suite of protocols for the Internet and other networks. TCP (Transmission Control Protocol) and IP (Internet Protocol) are the two original components of this suite.

Available Tools

InterSystems IRIS TCP binding

Enables you to set up communication between InterSystems IRIS® data platform processes using TCP/IP. See the I/O Device Guide.

Availability: All namespaces.

%SYSTEM.INetInfo class

Includes the following methods:

- **TCPName()**
- **TCPStats()**

Availability: All namespaces.

%SYSTEM.Socket class

Provides an interface for multiplexing TCP devices. The reference information for this class includes examples.

Availability: All namespaces.

%SYSTEM.TCPDevice class

Provides an interface for retrieving the IP address and port of current InterSystems IRIS TCP device.

Availability: All namespaces.

TCP adapters

Enable a production to manage an incoming or outgoing TCP connection. The adapters allow simultaneous handling of multiple connections. They support character and binary data streams, and counted data blocks.

See Using TCP Adapters in Productions.

The TCP adapters are also built into many specialized production classes.

Availability: All interoperability-enabled namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

Telnet (Tools/APIs)

Use Telnet from within InterSystems IRIS® data platform.

Background Information

Telnet is a network protocol used on the Internet or in some networks. The term *Telnet* also refers to client software that uses this protocol.

Available Tools

%Net.TelnetStream class

Emulates the handshaking behavior of Windows Telnet.exe.

See %Net.TelnetStream.

Availability: All namespaces.

Config.Telnet class

Enables you to modify and access information about the [Telnet] section of the CPF. (Note that you usually modify this file via the Management Portal. See the [Telnet] section in Configuration Parameter Reference File.)

For this class, the class reference provides extensive details and an example.

Availability: %SYS namespace.

Telnet adapter

Enable a production to initiate and manage a Telnet connection.

See EnsLib.Telnet.OutboundAdapter.

Availability: All interoperability-enabled namespaces.

TLS (Tools/APIs)

Use TLS to communicate securely; obtain information about TLS connection in use.

Background Information

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide communication security over the Internet. (InterSystems sometimes uses the term *SSL/TLS* to refer to TLS.)

Available Tools

SSL/TLS configurations

InterSystems IRIS® data platform supports the ability to store a TLS configuration and specify an associated name. When you need a TLS connection (for HTTP communications, for example), you provide the applicable configuration name, and InterSystems IRIS automatically handles the TLS connection.

See InterSystems TLS Guide.

Configurations are stored in the Security.SSLConfigs class, which provides an object-based API; this class cannot be accessed via SQL.

Availability: All namespaces.

%SYSTEM.Security.Users class

Provides methods that you can use to get information about the TLS connection in use on the principal device, if any. These methods include:

- **SSLGetCipher()**
- **SSLGetCipherList()**
- **SSLGetLastError()**
- **SSLGetPeerCertificate()**
- **SSLGetPeerName()**
- **SSLGetProtocol()**
- **SSLPeekClientHello()**

Availability: All namespaces.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

UDDI (Tools/APIs)

(Production-enabled namespaces) Use UDDI to work with web services.

Background Information

UDDI (Universal Description, Discovery, and Integration) is a XML-based registry for web services. It is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS).

Available Tools

EnsLib.UDDI package

Provides classes that represent an interface to a UDDI server. For example, EnsLib.UDDI.Connection represents a connection to a UDDI server.

Availability: All interoperability-enabled namespaces.

URLs (Tools/APIs)

Parse a URL into its component parts.

Available Tools

%Net.URLParser class

Provides the **Parse()** method.

Availability: All namespaces.

Version (Tools/APIs)

Obtain information about the system version.

Background Information

ObjectScript provides a special variable (**$ZVERSION**) that you can use to obtain version information for InterSystems IRIS® data platform. Note that **$ZVERSION** is available within SQL queries (as are other special variables).

Available Tools

%SYSTEM.Version class

Provides methods like the following:

- **GetComponents()**
- **GetBuildNumber()**
- **GetPlatform()**
- **GetProduct()**
- **GetVersion()**
- And others

Availability: All namespaces.

%ZHSLIB.HealthShareMgr class

Provides the **VersionInfo()** method, which displays information on HealthShare product versions.

Availability: All namespaces in a HealthShare instance.

Note

The special variable *$SYSTEM* is bound to the %SYSTEM package. This means that instead of `##class(%SYSTEM.class).method()`, you can use `$SYSTEM.class.method()`.

Web Gateway (Tools/APIs)

Manage the Web Gateway programmatically.

Background Information

The Web Gateway serves requests to REST services defined in InterSystems IRIS® data platform. It is a DLL or shared library installed on and loaded by the web server.

Usually you configure and manage the Web Gateway via the Web Gateway Management page; see the Web Gateway Guide.

Available Tools

%CSP.Mgr.GatewayMgr class

Defines an API used to control a Web Gateway from InterSystems IRIS code. Its methods provide the infrastructure for accessing (and modifying) the Web Gateway's internal tables, configuration, and log files from participating servers. Methods in this class include:

- **ClearCache()**
- **CloseConnections()**
- **GetCSPIni()**
- **GetInfo()**
- **GetSystemStatus()**
- **SetServerParams()**
- And others

You must have specific permissions to use these methods.

Availability: All namespaces.

%CSP.Mgr.GatewayRegistry class

Is a registry of Web Gateways managed by InterSystems IRIS. This class provides the following methods:

- **GetGatewayMgrs()**
- **RemoveFilesFromCaches()**

You must have specific permissions to use these methods.

Availability: All namespaces.

X12 (Tools/APIs)

(Production-enabled namespaces) Receive, work with, and send X12 documents.

Background Information

X12 is the ANSI standard for Electronic Data Interchange (EDI). There are more than 300 document types defined within this standard.

Available Tools

X12 support in productions

InterSystems IRIS® data platform provides classes that enable productions to receive, work with, and send X12 documents as virtual documents. See Routing X12 Documents in Productions.

Availability: All interoperability-enabled namespaces.

X.509 Certificates (Tools/APIs)

Use X.509 certificates.

Background Information

X.509 is a standard that defines elements that can be used for encryption, digital signatures, decryption, and verifying digital signatures. These elements include public keys and X.509 certificates.

Available Tools

X.509 certificate storage

InterSystems IRIS® data platform supports the ability to load an X.509 certificate and private key and specify an associated configuration name. When you need an X.509 certificate (to digitally sign a SOAP message, for example), you provide the applicable configuration name, and InterSystems IRIS automatically extracts and uses the certificate information.

You can optionally enter the password for the associated private key file, or you can specify this at runtime.

Configurations are stored in the %SYS.X509Credentials class, which provides an object-based API; this class cannot be accessed via SQL.

Availability: All namespaces.

Access to a certificate authority (CA)

If you place a CA certificate of the appropriate format in the prescribed location, InterSystems IRIS uses it to validate digital signatures and so on.

Availability: All namespaces.

Both items are discussed in Securing Web Services and Using XML Tools.