



InterSystems Programming Tools Index

Version 2019.4
2020-01-28

InterSystems Programming Tools Index

InterSystems IRIS Data Platform Version 2019.4 2020-01-28

Copyright © 2020 InterSystems Corporation

All rights reserved.

InterSystems, InterSystems IRIS, InterSystems Caché, InterSystems Ensemble, and InterSystems HealthShare are registered trademarks of InterSystems Corporation.

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

About This Book	1
Tools Index	3
.NET	4
Applications	5
Archiving	6
Arrays	7
Auditing	8
Authentication	9
Authorization	10
Bitstrings	11
Business Intelligence	12
Business Logic	13
C	14
Class Definitions	15
Concurrency Mode	17
CPF	18
CPUs (Processors)	19
CSV Files	20
Current Date and Time	21
Custom Language Elements	22
Custom Metrics	23
Dashboards	24
Data Synchronization	25
Data Transformation	26
Data Type Classes	27
Databases	28
Date/Time Values	30
DDL	32
Debugging	33
Devices	34
Directories and Drives	35
DLLs and Executables (Non-InterSystems)	36
DOMs	37
Dynamic SQL	38
EDIFACT	39
Email	40
Encryption	41
Environment Variables	43
Exporting Data	44
Extents	45
Files	46
FTP	47
Globals	48
GUIDs (Globally Unique Identifiers)	51
HTTP	52
Importing Data	53
Include Files	54

Installation	55
Inventory Facility	56
IP Addresses	57
ISQL	58
Java	59
JSON	60
LDAP	61
Licenses	62
Lists	63
Locks	64
Macros	65
MDX	66
Memory	67
messages.log	68
MIME	69
MQ (IBM WebSphere MQ)	70
Namespaces	71
Node.js	73
Objects	74
ObjectScript	75
ODBC	76
Operating System	77
Operating System Commands	78
Packages	79
Performance Diagnosis	81
PMML (Predictive Modelling Markup Language)	82
Processes (Jobs)	83
Productions	86
Publish and Subscribe	87
Push Notifications	88
Regular Expressions	89
REST (Web Services and Clients)	90
Routines	91
SAML Tokens	93
SASL	94
Security Items	95
Server	97
Siebel	98
SOAP (Web Services and Clients)	99
SQL	100
SQL Gateway Connections	103
SSH	104
SSL/TLS	105
Startup and Shutdown Behavior	106
Status Values	107
Streams	109
Tasks	110
TCP/IP	111
Telnet	112
Testing	113
Transact-SQL	114

UDDI	115
Unstructured Data	116
URLs	117
User Portal	118
Version	119
Web Gateway	120
Workflow	121
X12	122
X.509 Certificates	123
XML	124
XPATH	125
XSLT	126

About This Book

This book is an index of the InterSystems tools for programmers. This reference is intended to help you find the tool that meets your needs, with links to the appropriate sources of information. It contains one section:

- [Tools Index](#)

For the main topic list, see the [table of contents](#).

Other Topics

If there is no reference page for the item you seek, consult the following list:

- Concurrency — See [Locks](#)
- Informix — See [ISQL](#)
- JDBC — See [Java](#)
- Internet addresses — See [IP Addresses](#)
- Login — See [Authentication](#)
- Mappings — See [CPF](#)
- MSSQL — See [TSQL](#)
- MTOM — See [SOAP \(Web Services and Clients\)](#)
- POP3 — See [Email](#)
- Privileges — See [Security Items](#)
- Resources — See [Security Items](#)
- Result sets — See [Dynamic SQL](#)
- Roles — See [Security Items](#)
- SMTP — See [Email](#)
- SQLServer — See [Transact-SQL](#)
- Sybase — See [Transact-SQL](#)
- Users — See [Security Items](#)
- WS-Security, WS-Policy, and other SOAP-related specifications — See [SOAP \(Web Services and Clients\)](#)

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.

If there is no reference page for the item you seek, see the additional list in the [preface](#) to this book.

.NET

Interoperate with .NET clients and objects.

Background Information

The .NET framework is a software framework that runs primarily on Microsoft Windows.

Available Tools

InterSystems Managed Provider for .NET

Provides an easy and efficient way to use InterSystems IRIS from .NET applications. The InterSystems Managed Provider is unique in that it offers simultaneous relational (SQL) and object access to data via a common API and without requiring any object-to-relational mapping. See *Using the InterSystems Managed Provider for .NET*.

Availability: All namespaces.

Gateway for .NET

Enables you to instantiate an external .NET object and manipulate it as if it were a native object within InterSystems IRIS. See *Using the Object Gateway for .NET*. Note that you can also use the .NET Gateway in a production.

Availability: All namespaces.

InterSystems IRIS XEP

Provides extremely rapid storage and retrieval of structured data. XEP provides ways to control schema generation for optimal mapping of complex data structures, but schemas for simpler data structures can often be generated and used without modification.

See *Persisting .NET Objects with InterSystems XEP*.

Applications

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 “Applications” in the *Security Administration Guide*.

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

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.

Arrays

Work with arrays.

Background Information

In InterSystems IRIS, the word *array* can refer to either of the following items:

- An InterSystems IRIS multidimensional array. ObjectScript provides commands for working directly with InterSystems IRIS multidimensional arrays. *Using Globals* describes these arrays, which have two general forms: persistent and in-memory. Persistent multidimensional arrays are known as *globals*.

See *ObjectScript Reference*.

- An instance of any of the InterSystems IRIS array classes.

The InterSystems IRIS class library provides classes that represent a simple form of array — key and value. (In contrast, InterSystems IRIS multidimensional arrays do not have a fixed structure.)

This reference topic lists these classes.

Available Tools

%Collection.ArrayOfDT class

Defines the interface for an array property whose elements are literal values. That is, each array item has a key and a literal value.

For an introduction, see “Working with Collections” in *Defining and Using Classes*.

Availability: All namespaces.

%Collection.ArrayOfObj class

Defines the interface for an array property whose elements are objects. That is, each array item has a key and an object value.

For an introduction, see “Working with Collections” in *Defining and Using Classes*.

Availability: All namespaces.

%ArrayOfDataTypes class

Defines a stand-alone object that represents an array of literal values. Each array item has a key and a literal value.

Availability: All namespaces.

%ArrayOfObjects class

Defines a stand-alone object that represents an array of objects. Each array item has a key and an object value.

Availability: All namespaces.

%ArrayOfObjectsWithClassName class

A version of the %ArrayOfObjects collection class that stores class names in OIDs. These classes can be used stand alone to store a collection.

For details, see the class reference for %ArrayOfObjectsWithClassName.

Availability: All namespaces.

Auditing

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, 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” in the *Security Administration 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 “[Using Character-based Security Management Routines](#)” in the *Security Administration Guide*.

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 the article “[Setting Up Structured Logging](#).”

Availability: %SYS namespace.

Reminder

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()`.

Authentication

Authenticate users.

Background Information

Authentication is the process by which a user proves who he or she is when accessing a system.

Available Tools

Server authentication options

InterSystems provides the following options that enable you to authenticate users:

- Kerberos authentication — The most secure means of authentication. The Kerberos Authentication System provides mathematically proven strong authentication over a network.
- Operating-system–based authentication — OS-based authentication uses the operating system’s identity for each user to identify that user to InterSystems IRIS.
- LDAP authentication — With the Lightweight Directory Access Protocol (LDAP), InterSystems IRIS authenticates the user based on information in a central repository, known as the LDAP server.
- 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.
- 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 the server without performing any authentication. This option is appropriate for organizations with strongly protected perimeters or in which neither the application nor its data are an attractive target for attackers.

For information, see the *Security Administration Guide*. Also see the *Security Tutorial*.

Availability: All namespaces.

Support for security-related SOAP specifications

InterSystems IRIS web service services and web clients can validate the WS-Security header element for inbound SOAP messages, as well as automatically decrypt the inbound messages. Generally speaking, this security header element can carry information that authenticates the sender. See *Securing Web Services*.

Availability: All namespaces.

The bindings and gateways also provide support for authentication, as discussed in the books for those bindings and gateways.

See Also

- [LDAP](#)

Authorization

Control what each user is permitted to use, view, or alter.

Background Information

Authorization is the process of controlling what each user can use, view, or alter in the application.

Available Tools

Role-based authorization model

InterSystems uses a role-based authorization model; roles carry permissions, and users are assigned to roles. A user then has the cumulative permissions provided by all the roles to which the user belongs.

As a programmer, you are responsible for including the appropriate security checks within your code to make sure that a given user has permission to perform a given task. You can also temporarily give a user permissions for a specific task, using an option called *role escalation*.

For information, see the *Security Administration Guide*. Also see the *Security Tutorial*.

Availability: All namespaces.

Bitstrings

Convert strings to bitstrings.

Background Information

A *bitstring* is a kind of string, made up of a compressed series of bits. ObjectScript provides the **\$BIT** function and other functions for working with bitstrings. For an overview, see “Bitstrings” in the *ObjectScript Tutorial*.

Available Tools

In addition to the language functions, InterSystems provides the following tools:

%SYSTEM.Bit class

Provides the following methods to convert to bitstrings:

- **StringToBit()**
- **ZBitToBit()**

Availability: All namespaces.

Reminder

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()`.

Business Intelligence

Analyze data; develop a measurement-based approach to making strategic and tactical decisions.

Background Information

Business Intelligence (BI) is a set of tools and techniques that transform raw data into insights that can improve the operation of a business or other organization. BI is intended to support a measurement-based approach to making strategic and tactical decisions.

Available Tools

InterSystems IRIS Business Intelligence

InterSystems IRIS Business Intelligence enables you to embed business intelligence queries and dashboards into your applications so that your users can ask and answer sophisticated questions of their data by performing drag and drop actions.

In contrast to traditional BI systems that use static data warehouses, InterSystems IRIS BI cubes are kept closely in synchronization with the live transactional data.

See *Introduction to InterSystems Business Intelligence*.

Availability: All analytics-enabled namespaces.

See Also

- [Dashboards](#)
- [Unstructured Data](#)

Business Logic

(Production-enabled namespaces) Model business rules and processes within a production.

Background Information

Business logic is logic that describes the business rules and processes of an organization. It is generally desirable to capture business logic in a format that is modifiable by non-technical users.

Available Tools

production business rules

Enable non-technical users to change the behavior of business processes at specific decision points. The logic of the rule can be changed instantly, using a simple forms-based interface in the Management Portal. There is no need for programming or diagramming skills to change the rule, and there is no need to modify or compile production code for changes to take effect.

See *Developing Business Rules*.

Availability: All interoperability-enabled namespaces.

See Also

- [Workflow](#)

C

Access InterSystems IRIS 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

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, 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 typically use Atelier to create, examine, compile, and export classes. See the Atelier home page on the InterSystems website: <http://www.intersystems.com/our-products/embedded-technologies/atelier/>.

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” in *Defining and Using 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.

Reminder

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

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” in *Defining and Using Classes*.

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.

Reminder

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

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 operates. InterSystems IRIS reads this file for configuration settings when it starts up. The default CPF can be customized as desired as part of InterSystems IRIS deployment. See “Introduction to the Configuration Parameter File” in the *Configuration Parameter File Reference* for more information.

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

Available Tools

CPF merge file

The `ISC_CPF_MERGE_FILE` environment variable lets you specify a separate file containing 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. This feature supports the deployment of multiple instances from the same source (container image or installation kit) with different configuration settings.

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)

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.

Reminder

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

- [Performance Diagnosis](#)

CSV Files

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

%SQL.Util.Procedures class

Provides the following methods:

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

Availability: All namespaces.

Current Date and Time

Obtain the current date and time.

Background Information

Each language provides system variables or functions that provide the current system time. See *ObjectScript Reference* and *InterSystems SQL Reference*.

In many InterSystems IRIS applications, dates and times are stored and transmitted in a format called *\$H format* or *\$HOROLOG format*. For an introduction, see “Date and Time Values” in the *Orientation Guide for Server-Side Programming*.

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.

Reminder

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](#)

Custom Language Elements

Add custom commands, functions, and special variables to ObjectScript.

Available Tools

^%ZLANG* extension feature

You can use the **^%ZLANG** extension feature to add custom commands, functions, and special variables to the ObjectScript and other languages. Then you can invoke your extensions in the same way as standard features.

To add custom language elements, define routines as described in “Extending Languages with %ZLANG Routines” in *Specialized System Tools and Utilities*.

Availability: You define these routines in the %SYS namespace and they affect all namespaces.

Custom Metrics

Define custom metrics.

Background Information

A metric is a quantitative indicator of activities or performance.

InterSystems provides monitoring tools that include system metrics of various kinds; these are not discussed here. This topic focuses on custom metrics that you can add to your applications.

Available Tools

InterSystems IRIS Business Intelligence KPIs

A KPI is a key performance indicator. In InterSystems IRIS Business Intelligence, a KPI is a query whose results can be displayed on a dashboard.

For details, see “Defining Basic KPIs” and following chapters in *Advanced Modeling for InterSystems Business Intelligence*.

Availability: All namespaces.

Production business metrics

An business metric is a specialized business service class that defines and calculates a set of one or more measured values.

For details, see “Defining Business Metrics” in *Developing Productions*.

Availability: All interoperability-enabled namespaces.

See Also

- [Dashboards](#)

Dashboards

Create dashboards for your end users.

Background Information

In web applications, a dashboard is a page that displays key information, often in a condensed format.

Available Tools

Analytics dashboard technology

Enables you to create web-based dashboards for your end users. Despite the specific name, these dashboards are not reserved solely for Analytics users.

Dashboards are typically intended to address specific business needs. You can create dashboards to display any mix of Analytics data items, business metrics, and custom portlets.

You can provide direct links to dashboards or provide a user portal to access them.

See *Creating Dashboards*.

Availability: All namespaces except for %SYS.

See Also

- [Custom Metrics](#)
- [User Portal](#)

Data Synchronization

Keep data synchronized on multiple systems.

Background Information

Data synchronization is the process of keeping data consistent in multiple systems.

Available Tools

InterSystems IRIS object synchronization tools

See “Object Synchronization” in *Defining and Using Classes*.

Availability: All namespaces.

Data Transformation

(Production-enabled namespaces) Modify data being sent within a production.

Background Information

Data transformation is the process of converting data from one format into another format.

Available Tools

InterSystems provides a rich set of tools that you can use to transform data. In addition, it offers the following specific item:

data transformations

InterSystems IRIS provides a kind of class called a *data transformation*; it defines a set of rules for creating a new production message or modifying an existing production message. There are two general kinds of data transformations:

- Ones based on `Ens.DataTransformDTL`

These use the Data Transformation Language (DTL). You can create these data transformations graphically in the Management Portal. Later you can modify them there or in Studio. For details, see *Developing DTL Transformations*.

- Ones based on `Ens.DataTransform`

These do not use DTL and can be edited only in Studio.

Availability: All interoperability-enabled namespaces.

See Also

- [Workflow](#)

Data Type Classes

Define properties to contain literal values.

Background Information

Data type classes enable you to enforce sets of rules about the values of literal-valued properties. InterSystems provides an extensible set of data type classes. Each data type class has the following features:

- It specifies values for compiler keywords. Among other things, these control how the property is projected to SQL and to clients such as Java.
- It specifies values for property parameters.
- It provides a set of methods to translate data among the stored (on disk), logical (in memory), and display formats.

Available Tools

%Library package

Provides many data type classes. For a table that lists them, see “Data Types” in *Defining and Using Classes*.

Availability: All namespaces.

See Also

- [Date/time Values](#)

Databases

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

Background Information

InterSystems IRIS stores data and code in database files. For an introduction, see “Namespaces and Databases” in the *Orientation Guide for Server-Side Programming*.

Typically you create and configure databases via the Management Portal. See “Configuring Databases” in the chapter “Configuring InterSystems IRIS” in the *System Administration Guide*.

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 the *Security Administration Guide*.

Availability: %SYS namespace.

See Also

- [Namespaces](#)

Date/Time Values

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.nnnnnnnnnn). 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 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()**)
- **SetDefaultTimePrecision()**
- **SetToDateDefaultFormat()**

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.

Reminder

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

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 the following chapters of *Using InterSystems SQL*:

- “Defining Tables”
- “Defining Views”

These chapters discuss DDL as one of the techniques for defining tables and views.

Availability: All namespaces.

%SYSTEM.SQL class

Includes the following class methods for working with DDL files:

- **DDLImport()**
- **DDLImportDir()**
- **SetDDLDropTabDelData()**
- **SetDDLIdentifierTranslations()**
- **SetDDLNo201()**
- **SetDDLNo30()**
- And others

Some of these methods require specific permissions.

Availability: All namespaces.

Reminder

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](#)

Debugging

Debug programs.

Available Tools

Atelier debugger

See the Atelier documentation, available on the home page on the InterSystems website: <http://www.intersystems.com/our-products/embedded-technologies/atelier/>.

Availability: All namespaces.

Debugging routines

Enable you to obtain information about the state of the process and examine errors. InterSystems provides the `^%STACK` routine and other routines. These are described in *Using ObjectScript*.

Availability: All namespaces.

See Also

- [Performance Diagnosis](#)

Devices

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 the *System Administration Guide*.)

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

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](#)

DLLs and Executables (Non-InterSystems)

Invoke dynamic link libraries and executables provided by other vendors.

Background Information

Dynamic link libraries (DLLs) and executables (EXEs) are files that contain instructions. In general, you use DLLs and EXEs to perform specialized tasks such as accessing specific systems, rendering graphics, or performing conversions or computations.

Available Tools

InterSystems Callout Gateway

Enables you to invoke executables, operating system commands, and user-written dynamic link libraries from within InterSystems IRIS. With the Callout interface, you can:

- Issue operating system commands on several platforms.
- Call programs written in other languages.
- Invoke functions from your own custom Callout library, written in C, C++, or any language that supports the C/C++ calling convention on your platform. A Callout library is a shared library (a DLL file on Windows, an SO file on UNIX® and related operating systems) that includes hooks to the InterSystems Callout Gateway.

See [Using the Callout Gateway](#).

Availability: All namespaces.

DOMs

Work with XML DOMs (document object model).

Background Information

Document Object Model (DOM) is an object model for representing XML and related formats.

Available Tools

%XML.Document and %XML.Node classes

Enables you to work with XML DOMs.

See “Representing an XML Document as a DOM” in *Using XML Tools*.

Availability: All namespaces.

See Also

- [XML](#)

Dynamic SQL

Create SQL statements that are prepared and executed at runtime.

Background Information

In InterSystems IRIS, the phrase *dynamic SQL* refers to SQL statements that are prepared and executed at runtime. Dynamic SQL lets you program within InterSystems IRIS in a manner similar to an ODBC or JDBC application (except that you are executing the SQL statement within the same process context as the database engine).

Available Tools

%SQL.Statement and %SQL.StatementResult classes

The preferred form of dynamic SQL. See “Using Dynamic SQL” in *Using InterSystems SQL*, which discusses all these classes.

Also see %SQL.CustomResultSet and other classes in this package.

Availability: All namespaces.

See Also

- [SQL](#)

EDIFACT

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

Background Information

EDIFACT (Electronic Data Interchange For Administration, Commerce, and Transport) is an international standard document format.

Available Tools

EDIFACT support in productions

InterSystems IRIS provides classes that enable productions to receive, work with, and send EDIFACT documents as virtual documents. See the *Routing EDIFACT Documents in Productions*.

Availability: All interoperability-enabled namespaces.

Email

Send and receive email programmatically.

Available Tools

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

Support email as follows:

- InterSystems IRIS 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

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

Support for managed key encryption

InterSystems IRIS includes support for managed key encryption, a suite of technologies that protects data at rest. These are:

- Block-level database encryption, also known simply as database encryption — A set of tools to allow creation and management of databases in which all the data is encrypted. Such databases are managed through the Management Portal.
- Data-element encryption for applications, also known simply as data-element encryption — A programmatic interface so that applications can include code to encrypt and decrypt individual data elements (such as particular class properties) as they are stored to and retrieved from disk.
- Encryption key management — A set of tools in the Management Portal for creating and managing data-encryption keys and for managing key files. Both database encryption and data-element encryption use key files to support their functionality.

For details, see “Managed Key Encryption” in the *Security Administration Guide*.

Availability: All namespaces.

SOAP support

InterSystems IRIS SOAP support includes the ability to encrypt and decrypt SOAP messages. See *Securing Web Services*.

Availability: All namespaces.

XML support

InterSystems IRIS XML support includes the ability to encrypt and decrypt XML documents. See “Encrypting XML Documents” in *Using XML Tools*.

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()**
- **ActivateEncryptionKey()**
- **GenCryptRand()**
- **HMACSHA()**

- **RSAGetLastError()**
- **RSASHA1Sign()**
- And others

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

Reminder

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

Access the value of an environment variable.

Available Tools

%SYSTEM.Util class

Provides the **GetEnviron()** method.

Availability: All namespaces.

Reminder

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

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” in *Using InterSystems SQL*.
- The **Export Globals** page, which exports globals to .gof files. See “Exporting Globals” in *Using Globals*.

Available Tools

%SQL.Export.Mgr class

Enables you to export SQL tables to text files. For information, see %SQL.ExlMData, 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 provide more specific export options, documented elsewhere.

Extents

Work with extent definitions programmatically.

Background Information

Each persistent class has an *extent*, which consists of all saved instances of that class. See “Introduction to Persistent Objects” in *Defining and Using Classes*.

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 an xDBC client. These methods are more important as examples of how the %ExtentMgr tables can be queried.

Availability: All namespaces.

Files

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 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](#)
- [Streams](#)

FTP

Use FTP from within InterSystems IRIS.

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

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

Background Information

InterSystems IRIS 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” in the *Orientation Guide for Server-Side Programming* and see the book *Using Globals*.

You can define *global mappings* so that you can access data in a non-default location; see “Configuring Namespaces” in the chapter “Configuring InterSystems IRIS” in the *System Administration Guide*. Typically you do this within the Management Portal.

The Management Portal also provides options for examining and managing globals. See “Managing Globals” in *Using 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.

Reminder

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)

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” in *Defining and Using Classes*.

%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.

Reminder

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

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

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” in *Using InterSystems SQL*.
- The **Import Globals** page, which imports globals from .gof files. See “Importing Globals” in *Using Globals*.

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 provide more specific import options, documented elsewhere.

See Also

- [SQL](#)

Include Files

Export include files programmatically.

Background Information

An include file contains definitions for ObjectScript macros. See “ObjectScript Macros and the Macro Preprocessor” in *Using ObjectScript*.

Available Tools

%SYSTEM.OBJ class

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

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

Availability: All namespaces.

Reminder

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

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 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

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 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

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

Background Information

InterSystems IRIS 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” in the chapter “Configuring InterSystems IRIS” in the *System Administration Guide*.

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.

Reminder

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()`.

ISQL

Migrate from ISQL.

Background Information

ISQL (INFORMIX-SQL) is a third-party implementation of SQL.

Available Tools

InterSystems ISQL

InterSystems ISQL is an implementation of Informix ISQL which supports many of its features. See the *ISQL Migration Guide*.

Availability: All namespaces.

Java

Connect from Java to InterSystems IRIS or from InterSystems IRIS to Java.

Connecting from Java to InterSystems IRIS

To connect to and leverage an InterSystems IRIS database from Java, the available tools are:

JDBC

Enables you to access InterSystems IRIS as a JDBC-compliant database. InterSystems IRIS includes a high-performance, pure Java, Type 4 JDBC database driver that is fully compliant with the JDBC 4.2 specification. This is a single standalone JAR file with no dependencies.

If you are already familiar with JDBC, and have a JDK 1.7 or 1.8 installed, just add the JDBC driver to your local *CLASSPATH*. The connection string is:

```
jdbc:IRIS://ipAddress:superserverPort/namespace
```

See *First Look: JDBC and InterSystems Databases*, *Using Java with the InterSystems JDBC Driver*, and the [InterSystems IRIS JDBC Javadoc](#).

XEP

Provides high-performance Java persistence for simple to medium-complexity object schemas. XEP projects a Java object as a *persistent event*, which is a persistent database object that stores the state of a Java object. XEP is optimized for transaction processing applications that require extremely fast and simple data persistence; you can use XEP for object persistence to avoid the need an object/relational mapping (ORM). See *First Look: XEP Object Persistence with InterSystems IRIS*, *Persisting Java Objects with InterSystems XEP*, and the [InterSystems IRIS XEP Javadoc](#).

Native API

Provides direct access to *globals*, which are tree-based sparse arrays at the core of InterSystems IRIS. The Native API for Java exposes these arrays and provides very fast and flexible access to them. The Native API also includes the ability to call InterSystems IRIS class methods and routines. See *First Look: InterSystems IRIS Native API* and *Using the InterSystems Native API for Java*.

Hibernate

Provides a framework for mapping Java classes to relational tables, also known as an object/relational mapping (ORM). InterSystems IRIS includes a Hibernate *dialect*. See “Hibernate Support” in the *Implementation Reference for Java Third Party APIs*.

Connecting from InterSystems IRIS to Java

To connect to and leverage Java from an InterSystems IRIS database, the available tools are:

Java Gateway

Enables InterSystems IRIS to interoperate with Java components. The Java Gateway can instantiate an external Java object and manipulate it as if it were a native object within InterSystems IRIS. See *Using the Java Gateway in Productions*.

See Also

- [Designing a Java Connection Strategy](#), which provides guidance about Java options

JSON

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 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

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 to use LDAP authentication; see “Using LDAP Authentication” in the *Security Administration Guide*. Also see the *Security Tutorial*.

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 the %SYS.LDAP entry in the *InterSystems Class Reference*.

Availability: All namespaces.

LDAP Outbound adapter

Sends requests to an LDAP server and receives responses.

See EnsLib.LDAP.OutboundAdapter in the *InterSystems Class Reference*.

Availability: All interoperability-enabled namespaces.

See Also

- [Authentication](#)

Licenses

Access information about InterSystems IRIS license usage programmatically; configure license servers.

Background Information

See “Managing InterSystems IRIS Licensing” in the *System Administration Guide*.

Available Tools

%SYSTEM.License class

Provides an interface to the InterSystems IRIS 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 the *System Administration Guide*.)

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

The class documentation includes examples and details.

Availability: %SYS namespace.

Reminder

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()`.

Lists

Work with lists.

Background Information

In InterSystems IRIS, the word *list* can refer either of the following:

- A native list as defined by functions in ObjectScript.
See ObjectScript Reference.
- An instance of any of the InterSystems IRIS list classes.

This reference topic lists these classes.

List Classes

%Collection.ListOfDT class

Defines the interface for a list property whose elements are literal values.

For an introduction, see “Working with Collections” in *Defining and Using Classes*.

Availability: All namespaces.

%Collection.ListOfObj class

Defines the interface for a list property whose elements are objects.

For an introduction, see “Working with Collections” in *Defining and Using Classes*.

Availability: All namespaces.

%ListOfDataTypes class

Defines a stand-alone object that represents a list of literal values.

Availability: All namespaces.

%ListOfObjects class

Defines a stand-alone object that represents a list of objects.

Availability: All namespaces.

%List and %ListOfBinary classes

These datatype classes represents the ObjectScript native list format.

Availability: All namespaces.

%ListOfObjectsWithClassName class

A version of the %ListOfObjects collection class that stores class names in OIDs. These classes can be used stand alone to store a collection.

Availability: All namespaces.

Locks

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” in the *Monitoring Guide*.

For more information on locks, see the article *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.

%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.

See Also

- [Concurrency Mode](#)

Macros

Export macros programmatically; print information about available macros.

Background Information

A macro defines a substitution in a line of ObjectScript code. For details, see “ObjectScript Macros and the Macro Preprocessor” in *Using ObjectScript*.

Available Tools

%SYSTEM.OBJ class

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

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

Availability: All namespaces.

Reminder

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()`.

MDX

Create and execute MDX queries.

Background Information

MDX (MultiDimensional Expressions) is a standard query language for OLAP (online analytical processing) databases.

Available Tools

InterSystems IRIS Business Intelligence

The InterSystems IRIS Business Intelligence Analyzer enables you to create and execute MDX queries via drag and drop. You can also execute MDX queries programmatically. For information, see *Using InterSystems MDX* and *InterSystems MDX Reference*.

Availability: All analytics-enabled namespaces.

Memory

Modify the memory settings programmatically.

Background Information

You typically modify memory settings via the Management Portal. See “Configuring System Information” in “Configuring InterSystems IRIS” in the *System Administration Guide*. Also see “Advanced Memory” in the *Configuration Parameter File Reference*.

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 generic 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.

Reminder

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

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

Background Information

InterSystems IRIS 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” in the chapter “Monitoring InterSystems IRIS Using the Management Portal” in the *Monitoring Guide*.

The directory location of this file is configurable; see the ConsoleFile entry in *Configuration Parameter File Reference*.

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 the article “Setting Up Structured Logging.”

Availability: %SYS namespace.

Reminder

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

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. For details and examples, see *Using Internet Utilities*.

Availability: All namespaces.

See Also

- [Email](#)

MQ (IBM WebSphere MQ)

Exchange messages between InterSystems IRIS 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

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

Background Information

In InterSystems IRIS, 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” in the *Orientation Guide for Server-Side Programming*.

Typically you create and configure namespaces via the Management Portal. See “Configuring Namespaces” in the chapter “Configuring InterSystems IRIS” in the *System Administration Guide*.

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 to work with InterSystems IRIS. 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.

Reminder

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](#)

Node.js

Use Node.js to implement for high-performance access to data held in InterSystems IRIS databases.

Background Information

Node.js is a JavaScript-based software system designed for creating Internet applications.

Available Tools

Native API

Enables a Node.js client application to access multidimensional storage (globals) and call user-defined ObjectScript methods and functions. See *Using the Native API for Node.js*.

Availability: All namespaces.

node-odbc open-source Node.js module

Enables a Node.js client application to access an IRIS database via ODBC. See “Support for Node.js Relational Access” in *Using the InterSystems ODBC Driver*,

Objects

Define objects.

Background Information

An object is a container for a set of values that are stored together or passed together as a set. An object often corresponds to a real-life entity, such as a patient, a patient diagnosis, a transaction, and so on.

To define a class that can represent objects, you create a class that inherit from the core InterSystems object classes. An instance of that class is an object. For details, see *Defining and Using Classes*.

Available Tools

%RegisteredObject class

Defines registered objects. You can create these objects but you cannot save them. Registered objects are useful for carrying sets of related values that do not need to be stored.

Availability: All namespaces.

%Persistent class

Defines persistent objects. You can create and save these objects.

A persistent class is automatically projected to a table that you can access via InterSystems SQL.

Availability: All namespaces.

%SerialObject class

Defines serial objects. A serial class is meant for use as a property of another object. You can create and delete these objects, but you cannot save them or open them independently of the object that contains them.

When contained in persistent objects, these objects have an automatic projection to SQL.

Availability: All namespaces.

These classes are explained extensively in *Defining and Using Classes*.

ObjectScript

Write routines and methods in ObjectScript.

Background Information

ObjectScript is the native server-side language provided by InterSystems IRIS.

Available Tools

ObjectScript

For information on ObjectScript, see the following books:

- *Orientation Guide for Server-Side Programming*
- *Using ObjectScript*
- *Defining and Using Classes*
- *ObjectScript Language Reference*
- *Class Definition Reference*

Availability: All namespaces.

ODBC

Access third-party ODBC-compliant databases from within InterSystems IRIS; access InterSystems IRIS as an ODBC-compliant database.

Background Information

ODBC (Open Database Connectivity) is a C-based standard for accessing data.

Available Tools

InterSystems ODBC driver

Enables you to access InterSystems IRIS as a ODBC-compliant database. The InterSystems ODBC driver is a native driver — it is not built on top of any other proprietary interface. The InterSystems ODBC driver offers the following features:

- High performance
- Portability
- Native Unicode support
- Thread safety

You can use InterSystems ODBC with any tool, application, or development environment that supports ODBC.

See *Using the InterSystems ODBC Driver*.

Availability: All namespaces.

InterSystems SQL Gateway

Enables your InterSystems IRIS applications to access third-party relational databases. 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.

See Also

- [SQL](#)
- [SQL Gateway](#)
- [Java](#)

Operating System

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.

Reminder

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

Invoke operating system commands from within InterSystems IRIS.

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` in the *InterSystems Class Reference*.

Availability: All interoperability-enabled namespaces.

See Also

- [DLLs and Executables \(non-InterSystems\)](#)

Packages

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” in *Defining and Using Classes*.

You can define *package mappings* so that you can access code in a non-default location; see “Configuring Namespaces” in the chapter “Configuring InterSystems IRIS” in the *System Administration Guide*. 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.

Reminder

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()`.

Performance Diagnosis

Diagnose performance.

Available Tools

Diagnostic routines

Enable you to gather data you can use for performance diagnosis. InterSystems provides a set of routines that include `^GLOSTAT`, `^%SYS.MONLBL`, and others. See the *Monitoring Guide*.

Availability: Most of these are available in all namespaces. See the *Monitoring Guide* for details.

`%SYS.PTools.StatsSQL` class

Enables you to collect performance statistics on SQL queries. See “SQL Runtime Statistics” in *Using InterSystems SQL*.

Availability: All namespaces.

See Also

- [Debugging](#)

PMML (Predictive Modelling Markup Language)

Execute PMML models.

Background Information

PMML (Predictive Modelling Markup Language) is an XML-based standard that expresses analytics models. It provides a way for applications to define statistical and data mining models so that they can be easily reused and shared.

In a typical scenario, data scientists use an analytical tool to produce a data mining model based on large amounts of historical data, which is then exported to PMML. The model can then be deployed in a runtime environment and executed on incoming observations, predicting values for the model's target metrics.

Available Tools

%DeepSee.PMML classes

Enable you to execute existing PMML models. You can place an existing PMML definition into a subclass of %DeepSee.PMML.Definition. InterSystems IRIS generates code to support the models contained in the definition and provides an API to execute the models.

See *Using PMML Models in InterSystems IRIS*.

Availability: All Analytics-enabled namespaces.

Processes (Jobs)

Get information about and manipulate CPU processes (known as jobs in InterSystems IRIS).

Background Information

A CPU process is an instance of an InterSystems IRIS 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” in the *System Administration Guide*.

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.

Reminder

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

(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 Atelier). 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.

Publish and Subscribe

(Production-enabled namespaces) Route messages to subscribers based on their expressed interests.

Background Information

Publish and subscribe refers to a technique where a message is sent to one or more subscribers based on their interest in a specific topic. Typically a topic is inferred from the contents of the message.

Available Tools

EnsLib.PubSub package

Provides a mechanism to quickly find the set of interested subscribers for a given topic. Your application-specific business process must both invoke the PubSub routing engine and also dispatch messages to subscribers as part of a business process (this is in order to make the actual processing of messages as flexible as possible).

See “Defining Publish and Subscribe Message Routing” in *Managing Productions*.

Availability: All interoperability-enabled namespaces.

Push Notifications

Send push notification messages to iOS and Google devices.

Background Information

Push notifications are messages sent to an application by a central server, according to preferences defined by the user.

Available Tools

Production Push Notifications feature

Enables a production to send push notification messages to iOS and Google devices. This feature makes it possible to easily send notifications to a user with an application running on multiple devices. Your code can send notifications without having to be aware of the number of devices or whether the devices are iOS or Google devices. See *Configuring and Using Push Notifications*.

Availability: All interoperability-enabled namespaces.

Regular Expressions

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 “\$MATCH” in the reference “ObjectScript Functions” in *ObjectScript Reference* and see “Regular Expressions” in *Using ObjectScript*.

\$MATCH ObjectScript function

Returns a boolean value depending on whether a string matches a regular expression. See “\$LOCATE” in the reference “ObjectScript Functions” in *ObjectScript Reference* and see “Regular Expressions” in *Using ObjectScript*.

%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” in *Using ObjectScript*.

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 “Pattern Matching” in *Using ObjectScript*.

REST (Web Services and Clients)

Create REST web services and web clients.

Background Information

REST is a simple stateless architecture that generally runs over HTTP. The REST acronym represents Representational State Transfer.

Available Tools

InterSystems IRIS support for REST web services

Enables you to create web services in InterSystems IRIS. The REST support includes the following items:

- `%CSP.REST` class
- URL maps that specify the InterSystems IRIS method that is executed for a REST URL.
- JSON and XML support provided by InterSystems IRIS.

For information on creating REST web services, see *Creating REST Services*. For information on adding REST web services and REST web clients to a production, see *Using REST Services and Operations in Productions*.

Availability: All namespaces.

See Also

- [XML](#)

Routines

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 the following books:

- *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” in the chapter “Configuring InterSystems IRIS” in the *System Administration Guide*. 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.

Reminder

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()`.

SAML Tokens

Add SAML tokens to WS-Security headers.

Background Information

SAML (Security Assertion Markup Language) is an XML-based standard for exchanging authentication and authorization data. There are multiple uses of SAML; one use is to include SAML tokens in SOAP messages as a security element.

Available Tools

InterSystems IRIS SOAP support

With InterSystems IRIS SOAP support, you can add a SAML token to the WS-Security header element. See *Securing Web Services*. For information on supported security standards, see “Standards Supported in InterSystems IRIS” in that book.

Full SAML support is not implemented.

Availability: All namespaces.

SASL

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

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

Background Information

For an introduction to security in InterSystems IRIS, see “About InterSystems Security” in the *Security Administration Guide*.

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 “[Using Character-based Security Management Routines](#)” in the *Security Administration Guide*.

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

Reminder

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

Obtain information about the InterSystems IRIS 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” in the *Web Gateway Configuration Guide*.

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.

Reminder

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()`.

Siebel

(Production-enabled namespaces) Communicate with a Siebel server.

Background Information

Siebel is a third-party customer relationship management (CRM) solution.

Available Tools

Siebel adapter

Enables a production to send HTTP requests to a Siebel server and receive responses.

See `EnsLib.Siebel.HTTPOutboundAdapter` in the *InterSystems Class Reference*.

Availability: All interoperability-enabled namespaces.

SOAP (Web Services and Clients)

Create SOAP-compliant web services and web clients.

Background Information

SOAP, originally called Simple Object Access Protocol, is a protocol specification for exchanging structured information between clients (called web clients) and services (called web services) over a network.

Available Tools

Basic SOAP support in InterSystems IRIS

Enables you to create web services and web clients in InterSystems IRIS. InterSystems IRIS SOAP support includes the following items:

- Support for [XML](#)
- %SOAP.WebService class
- %SOAP.WebClient class
- Other classes in the %SOAP package
- Support for [SSL/TLS](#) and [X.509 certificates](#)

For basic information on InterSystems IRIS web services and clients, see *Creating Web Services and Web Clients* and *Securing Web Services*. For information on the supported standards, see “Standards Supported in InterSystems IRIS” in that book.

Also see *Securing Web Services*. For information on supported security standards, see “Standards Supported in InterSystems IRIS” in that book.

Availability: All namespaces.

SOAP support for productions

Enable you to create production web services (which are business services) and production web clients (which are business operations).

See *Adding SOAP Services and Web Clients to Productions*.

Availability: All interoperability-enabled namespaces.

See Also

- [XML](#)

SQL

Use SQL within InterSystems IRIS; 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

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 InterSystems IRIS 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 *System Administration Guide*.)

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.

Reminder

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](#)

- [Dynamic SQL](#)
- [SQL Gateway Connections](#)

SQL Gateway Connections

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

Background Information

The SQL Gateway allows InterSystems IRIS 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.

Reminder

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

- [ODBC](#)
- [Java](#)

SSH

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 [SSL/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” in *Using Internet Utilities*.

Availability: All namespaces.

SSL/TLS

Use SSL/TLS to communicate securely; obtain information about SSL/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 uses the term *SSL/TLS* to refer collectively to these protocols.

Available Tools

SSL/TLS configurations

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

See the *Security Administration 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 SSL/TLS connection in use on the principal device, if any. These methods include:

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

Availability: All namespaces.

Reminder

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()`.

Startup and Shutdown Behavior

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 the *System Administration Guide*.)

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 Terminal*.

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” in *Specialized System Tools and Utilities*.

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.

Status Values

Work with the %Status data type.

Background Information

InterSystems classes use the %Status data type class to represent status information. Many methods return a %Status value to indicate success or failure (along with reason for failure).

Available Tools

System-supplied macros

ObjectScript provides some macros for working with %Status values. These include:

- `$$$ADDSC`
- `$$$EMBEDSC`
- `$$$ERROR`
- `$$$ISERR`
- `$$$ISOK`

For information, see “System-supplied Macro Reference” in the chapter “ObjectScript Macros and the Macro Preprocessor” in *Using ObjectScript*.

%SYSTEM.Status class

This class provides methods for working with %Status values. These methods include:

- **AppendStatus()**
- **DecomposeStatus()**
- **DisplayError()**
- **GetErrorText()**
- **IsError()**
- **IsOK()**
- And others

Availability: All namespaces.

%SYSTEM.OBJ class

This class provides the following method for working with %Status values:

- **DisplayError()**

Reminder

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

- [Working with %Status Values](#)

Streams

Work with streams.

Background Information

Streams are used to create properties containing extremely large amounts of data (beyond the string length limit).

Available Tools

%Stream package

Provides the recommended interface to streams. These classes include:

- `%Stream.FileBinary`
- `%Stream.FileCharacter`
- `%Stream.GlobalBinary`
- `%Stream.GlobalCharacter`

See “Working with Streams” in *Defining and Using Classes*.

Availability: All namespaces.

%Library package

Includes `%CharacterStream` and other older, deprecated classes for working with streams.

Availability: All namespaces.

See Also

- [Files](#)

Tasks

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 in the Management Portal; see “Using the Task Manager” in the *System Administration Guide*.

Available Tools

%SYS.TaskSuper class

Persistent class that represents tasks. This class includes methods like the following:

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

It also provides class queries including the following:

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

Availability: All namespaces.

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

These classes define an API to schedule tasks to run in the background.

Availability: All namespaces.

TCP/IP

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 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, discussed in other books.

Availability: All interoperability-enabled namespaces.

Reminder

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

Use Telnet from within InterSystems IRIS.

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 NT Telnet.exe.

See %Net.TelnetStream in the *InterSystems Class Reference*.

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 *System Administration Guide*.)

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 in the *InterSystems Class Reference*.

Availability: All interoperability-enabled namespaces.

Testing

Perform automated testing of your code.

Available Tools

%UnitTest package

Provides classes that you can use to define and execute unit tests. See the *%UnitTest Tutorial*.

Availability: All namespaces.

%WebStress package

Provides classes that you can use to record, randomize and playback HTTP-based scripts against various applications for the purpose of QA, scalability, and network load testing. See the *InterSystems Class Reference*.

Availability: All namespaces.

Transact-SQL

Use Transaction-SQL in InterSystems IRIS.

Background Information

Transact-SQL is a third-party implementation of SQL. Transact-SQL is used with Microsoft SQL Server (MSSQL) and Sybase Adaptive Server.

Available Tools

InterSystems TSQL

Enables you to use Transact-SQL in InterSystems IRIS. InterSystems TSQL supports many of the features of both the Microsoft and Sybase implementations of the Transact-SQL language.

See the *Transact-SQL (TSQL) Migration Guide*.

Availability: All namespaces.

UDDI

(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.

Unstructured Data

Work with unstructured data.

Background Information

The term *unstructured data* refers to information that either does not have a predefined data model. Commonly, unstructured data consists of a large number of text sources, such as a collection of newspaper articles or a collection of doctors' notes.

Available Tools

InterSystems IRIS Text Analytics

Enable you to access and analyze unstructured data.

See *Using InterSystems Natural Language Processing (NLP)*.

Availability: All analytics-enabled namespaces.

See Also

- [Business Intelligence](#)

URLs

Parse a URL into its component parts.

Available Tools

%Net.URLParser class

Provides the **Parse()** method.

Availability: All namespaces.

User Portal

Provide a user portal for your end users.

Background Information

The term *user portal* is commonly used to refer to a web page or pages that provide self-service access to users; the portal enables a user to perform specific tasks.

Available Tools

InterSystems User Portal

The InterSystems User Portal is intended for direct use by end users (in contrast to such back end tools as Studio and the Management Portal). You can use this as part of your application, or you can create your own user portal if wanted.

The User Portal is designed to enable users to do the following tasks, depending on their permissions:

- View and use dashboards displaying key information. This information can include business metrics. Dashboards are typically intended to address specific business needs.
- (For Analytics users) View other Analytics data items.
- Organize these items into folders for easier management.
- Attach keywords to these items, so that they can find items again more quickly.
- Create dashboards.
- (For Analytics users) Access the Analyzer, in which users can create pivot tables or perform ad hoc analysis of data.
- (For workflow users) Manage their [workflow](#) tasks.

The *Using Dashboards and the User Portal* explains how to use the User Portal and how to work with dashboards; this book is intended for your end users.

Availability: All namespaces except for %SYS.

See Also

- [Dashboards](#)
- [Workflow](#)

Version

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. The other languages do not provide analogous variables.

Available Tools

%SYSTEM.Version class

Provides methods like the following:

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

Availability: All namespaces.

Reminder

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

Manage the Web Gateway programmatically.

Background Information

The Web Gateway serves requests to REST services defined in InterSystems IRIS. 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 Configuration 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.

Workflow

(Production-enabled namespaces) Incorporate human interaction into automated business processes.

Background Information

A workflow management system defines and manages a series of tasks performed by users. Users interact with the system to indicate when they have completed tasks. Typical uses of workflow might include order entry, order fulfillment, contract approval, or help desk activities.

Available Tools

Workflow Engine

Enables you to incorporate human interaction into automated business processes, within a production.

See *Using Workflow with InterSystems IRIS*.

Availability: All interoperability-enabled namespaces.

See Also

- [User Portal](#)

X12

(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 provides classes that enable productions to receive, work with, and send X12 documents as virtual documents. See the *Routing X12 Documents in Productions*.

Availability: All interoperability-enabled namespaces.

X.509 Certificates

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 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*.

XML

Project objects as XML documents; read, transform, and write XML documents; send and receive XML documents from productions; and so on.

Background Information

XML (Extensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Available Tools

InterSystems IRIS XML support

Enables you to project objects as XML documents, and read and write those documents. You can also work with arbitrary XML documents, perform XSLT transformations, evaluate XPath expressions, and so on.

InterSystems IRIS support for XML includes the following elements:

- The SAX Parser, which validates and parses inbound and outbound XML documents (SAX means “Simple API for XML”). This parser is a built-in parser using the standard Xerces library. SAX communicates with an InterSystems IRIS process using a high-performance, in-process call-in mechanism. You can fine-tune the parser or provide your own custom SAX interface classes.
- The %XML.Adaptor class, which enables you to represent instances of that class as XML documents.
- Other specialized classes in the %XML and %xsd packages.
- InterSystems IRIS support for [X.509 certificates](#) (which you can use for encryption and digital signatures)

See *Projecting Objects to XML* and *Using XML Tools*.

For information on the supported standards, see “Standards Supported in InterSystems IRIS” in *Using XML Tools*.

Availability: All namespaces.

XML virtual documents

InterSystems IRIS provides classes that enable productions to receive, work with, and send XML documents as virtual documents. See the *Routing XML Virtual Documents in Productions*.

Availability: All interoperability-enabled namespaces.

See Also

- [SOAP \(Web Services and Clients\)](#)

XPATH

Evaluate XPATH expressions.

Background Information

XPath (XML Path Language) is a query language for selecting nodes from an XML document.

Available Tools

%XML.XPATH.Document class

Enables you to evaluate XPATH expressions.

See “Evaluating XPath Expressions” in *Using XML Tools*.

Availability: All namespaces.

See Also

- [XML](#)

XSLT

Perform XSLT transformations.

Background Information

XSLT (Extensible Stylesheet Language Transformations) is a language used to generate a transformed version of an XML document.

Available Tools

%XML.XSLT.Transformer class

Enables you to perform XSLT transformations.

See “Performing XSLT Transformations” in *Using XML Tools*.

Availability: All namespaces.

See Also

- [XML](#)