Caché Release Notes and Upgrade Checklist

Version 2017.2
2020-06-25

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

Caché Release Notes and Upgrade Checklist
Caché Version 2017.2 2020-06-25

For Support questions about any InterSystems products, contact:

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

Table of Contents

About This Book

This book provides information on the major features that have been added to Caché in this release, as well as information needed to perform a successful upgrade to this release.

It contains the following sections:

- New and Enhanced Features for Caché 2017.2
- General Upgrade Information
- Upgrade Checklist for Caché 2017.2 — READ THIS INFORMATION BEFORE UPGRADING

The following books provide related information:

- Caché Release Notes and Upgrade Checklist Archive provides information on previous releases — both new features as well as information needed for upgrades.
- The online *InterSystems Supported Platforms* document for this release lists the technologies supported by this release of Caché.
- Caché Installation Guide describes the process of installing Caché on your system.
- Introduction to Caché provides an overview of the features and major components of Caché.

General Licensing Notes

InterSystems makes its products and features available under license to customers. While InterSystems may or may not enforce the use of said products or features consistent with the purchased license capabilities, customers are expected to comply with terms of their licenses. Moreover, InterSystems may tighten enforcement at any release without notice.

Developers must be aware that certain license types are required in order to use specific product features such as Multi-Server capability, Mirroring, and Web Services features. The specific requirements are noted in the InterSystems Price List and the Terms and Conditions for licensing. These are available from your local InterSystems representative.

Application Use Of InterSystems Web Server

InterSystems installs an Apache-based web server (often referred to as the "private web server") to assure that the Management Portals for its products are always available. The private web server is built and configured to meet the management needs of InterSystems administrative servers and is configured to only connect to InterSystems products. The options selected are not, in general, suitable for production use - in particular, security is minimal and the options used are generally unsuitable for a high volume of HTTP requests. Testing, by InterSystems, of the private webserver only covers use of the private web server for managing Caché, Ensemble, HealthShare, and other InterSystems products.

Customers are not required to use this web server to manage our products. You may also use a web server of your choice, located on whatever server you elect to use. The private web server is provided as a convenience to simplify installation and installation dependencies. Many developers also find it useful to use the private web server for unit testing.

UNIX®

The parameters used for the UNIX® build are:

```
--prefix=<installation_location>
--disable-actions
--disable-asis
--disable-auth
--disable-autoindex
--disable-cgi
--disable-cgid
--disable-charset-lite
--disable-dir
--disable-env
--disable-imap
--disable-include
--disable-negotiation
--disable-setenvif
--disable-shared
--disable-status
--disable-userdir
--enable-access
--enable-alias
--enable-log-config
--enable-mime
--enable-so
--without-berkeley-db
--without-gdbm
--without-ndbm
```

The server produced has defaults using the Apache Group's prefork Multi-Processing Module (MPM). This is the non-threaded server model. The number of requests that can be concurrently served is directly related to the number of Apache worker processes in the pool. The private web server is configured to occupy the smallest possible footprint by allowing a maximum of two worker processes to be created for the pool. The following settings will be found in the Apache configuration (httpd.conf) for the server:

- MinSpareServers: 1
- MaxSpareServers: 2

By contrast, the default Apache configuration for a production grade build is usually the following:

- StartServers: 5
- MinSpareServers: 2
- MaxSpareServers: 20
- ServerLimit: 256
- MaxClients: 256

This configuration will allow Apache to create 5 worker processes at start-up time, increasing to a maximum of 256 as the concurrent load increases. Because of these differences in configuration, the performance of the private web server will be noticeably inferior to that of a production grade Apache build as the concurrent load increases.

Windows

Windows-based Apache installations use the official binary distribution for Windows and a special multithreaded Multi-Processing Module (MPM) which is more suited to the way the operating system is optimized. However, since InterSystems installs and loads only a small subset of modules (mod_alias.so, mod_authz_host.so, mod_log_config.so and mod_mime.so.), their functionality is limited.

Conclusion

If you expect very low volume of HTTP traffic, have limited demands for high availability and secure operation, the private web server may be suitable for your deployment needs. However, if you expect a high amount of HTTP traffic, require high availability in your webserver, need to integrate with other sources of web information, or need a high degree of control over your web server, InterSystems recommends installing your own separate copy of Apache, ideally on its own server,

and configuring it to use our CSP gateway to communicate with Cache, Ensemble, or HealthShare. Review the options above to determine if this is so.

1

New and Enhanced Features for Caché 2017.2

This chapter includes:

1.1 Parallel Dejournaling For Mirroring and Journal Restore

In this release, the throughput of dejournaling is improved in Mirroring and Journal Restore. This is aimed at improving the scalability of mirrored systems that incur high rates of database updates.

Dejournaling is the process of applying records from journal files to the databases. Prior to this release, a single job performed all of the database updates in a dejournal session, aided by a series of prefetcher jobs. In this release, multiple updater jobs (up to four of them) can work in parallel to apply records to different databases. This feature is used if there are sufficient CPUs and shared memory heap available .

In Mirroring, use of parallel dejournaling can be restricted by member type with a new configuration setting. By default, the feature is off for Reporting Async members. If reports include data from multiple databases that is in the process of being updated by dejournaling, the differences in timing of updates to different databases could result in more variability in the results of those reports than in previous versions. The character of that variability is not substantially different than what could be expected in any report run against changing data, so it is expected that most reporting applications could use this feature with no negative impact.

For details on parallel dejournaling, see "Configuring Parallel Dejournaling" in the "Mirroring" chapter of the *Caché High Availability Guide*.

For journal restore, parallel dejournaling is not used if certain non-default options are selected. In particular, it is not used in conjunction with the option to abort after any error, nor with the option to journal applied updates. For more information, see "Restore Globals From Journal Files Using ^JRNRESTO" in the "Journaling" chapter of the *Caché Data Integrity Guide*.

Neither shadowing nor system startup make use of parallel dejournaling.

Finally, this release includes two improvements that apply regardless of whether parallel dejournaling is used. This release includes internal improvements to dejournal prefetching efficiency and it limits the memory consumption for the "dejournal queue" to 50MB per updater; prior releases may have used substantially more memory for the dejournal queue.

1.2 DeepSee New Features

DeepSee Folder Manager — To simplify the deployment of DeepSee components, there is a new Folder Manager option to export related items. When exporting to a container class, this new option will not only export the selected items but also other items that are related to the selected items. Related items for a dashboard include the pivot tables and termlists used by the dashboard. Related items for a pivot table include named filters, pivot variables, and shared calculated members.

Prior to this release the Folder Manager always used the server file system for exporting/importing files. Now folder manager provides the option of using the local file system or the server file system

Dashboard Filters — Named filters can now be used as the default value for dashboard filters.

1.3 iFind and iKnow New Features

iFind now supports cooccurrence search, allowing you to look for records where the search terms appear close to one another, but not necessarily in a particular order, as with the more strict positional search option. Use square brackets around a comma-separated list of search terms to enable cooccurrence search, optionally including a range within which the search terms need to occur. For example, the query [Boston, New York, 5] will look for records where "Boston" and "New York" appear within at most 5 positions of one another.

This change also includes a couple of performance improvements for building iKnow domains, especially when leveraging systems with high core counts. Depending on your hardware and dataset, you may see improvements from 10 to 30% in overall processing time.

There are a few extensions to the iKnow REST APIs. For example, you can now retrieve just the result count or just the count along with full results for most endpoints querying domain data.

1.4 SQL Enhancements

This release includes the following SQL enhancements:

- SQL Query Auditing
- Query Optimizations
- Optional ANSI SQL Operator Precedence
- Frozen Plan Evolution

1.4.1 SQL Query Auditing

This release adds the ability to audit the execution of SQL queries. There are three new system events for auditing SQL queries:

- %System/%SQL/DynamicStatement — for Dynamic SQL Queries
- %System/%SQL/EmbeddedStatement — for Embedded SQL Queries
- %System/%SQL/XDBCStatement - for xDBC Queries

To enable auditing for these events, go to the **System Audit Events** portal page by selecting **System Administration** > **Security** > **Auditing** > **Configure System Events**.

1.4.2 Query Optimizations

SQL queries have several improved optimizations. The query optimizer now considers outlier selectivity when calculating the selectivity of join conditions, resulting in improved plans for some queries. Outer joins can now take advantage of all optimizations available to inner joins. In particular, outer join conditions that could only partially be satisfied by an index no longer require the construction of a temporary file, often significantly improving query performance.

1.4.3 Optional ANSI SQL Operator Precedence

There is a new option to set SQL operator precedence to the ANSI SQL standard instead of the left-to-right CACHE SQL operator precedence. CACHE SQL operator precedence is still the default. You can change SQL operator precedence to ANSI SQL via an API or in the **General SQL Settings** portal page. Select **System Administration** > **Configuration** > **SQL and Object Settings** > **General SQL Settings**.

1.4.4 Frozen Plan Evolution

Previous releases of Caché included the ability to freeze SQL query plans and to automatically freeze query plans when you upgrade to a new version. If a new release provided improved optimization for a query plan, it would not be applied when you upgraded. With this release, Caché identifies any query with a frozen plan that might benefit from new optimizations and flags queries where the frozen plan may benefit from a new optimization. On the SQL Statement index page, these queries are identified in the New Plan column. You can then unfreeze these plans to take advantage of the new optimizations.

1.5 Where to Find Information on New Features in Atelier, the Eclipse-Based IDE

Atelier, the Eclipse-Based IDE for Caché, is available on an independent release cycle from Caché. Consequently, new features are described in the Atelier documentation provided with each new Atelier release. The Atelier IDE brings together the powerful and popular Eclipse development environment and the InterSystems Caché database. Atelier allows you to develop Caché applications using a modern file-based IDE on a client system. Atelier handles uploading the application to the Caché server where it can be run and debugged.

The focus of future development will be on the new Eclipse based IDE. Studio will remain an option to install and developers can continue to develop code with it. However, it will be treated as a maintenance product and will not see new functionality added as we move forward with Atelier. Some minor bugs may not be addressed either depending on resources required versus the severity of the issue.

Atelier is available as a separate download in addition to Caché or Ensemble. You can choose to install either a stand-alone Rich Client Platform (RCP) application, or a plug-in that can be added to an existing Eclipse installation. Users of the RCP application can add additional Eclipse plug-ins. Atelier uses the Eclipse auto-update mechanism to help users get the latest changes. For information on downloading Atelier and for the Atelier documentation, see http://www.intersystems.com/atelier, the Atelier home page.

1.6 Other Items of Note

In addition, many more minor improvements and corrections are also included. In particular, if you are upgrading an existing installation, please review the detailed list of changes in the Upgrade Checklist.

Areas of improvement include:

- XML Schema Wizard provides an option allowing cascading delete of subclasses when the instance of the superclass is deleted.
- Caché Nodejs supports node 7.
- New option to force CSP buffer to flush even if the buffer is not yet filled.

2

General Upgrade Information

InterSystems' ultimate goal is to have a release which can be installed with no, or little, effect on the applications it supports. This release provides an easier upgrade path if you are upgrading from version 2013.1 or later. If you are upgrading from a release prior to 2013.1, you must still follow the prior recommendations; for details, see "Pre-2014.1 Upgrade Information" in the *Caché Release Notes and Upgrade Checklist Archive*.

2.1 Important Considerations

2.1.1 Compatibility

The goal of each release is a smooth upgrade to new and improved functionality with no changes required to existing applications. However, because of error corrections, significant enhancements, or changes to applicable standards, this goal cannot always be met. In this case, InterSystems discloses all identified instances where changes to applications are necessary so that customers can gauge effort required to move to the new release.

This book covers the upgrade to the current version from the previous one. The information for upgrading from earlier releases is in the *Caché Release Notes and Upgrade Checklist Archive*. Among the items listed are: changes to the way the system operates; enhancements to the class and routine compilers; detailed changes made to system classes in terms of parameters, properties, and methods removed; changes to method call signatures; and differences in method returns.

You may, after reading the release-specific changes, conclude that none of them affect your application. Even in this case, regardless of how robustly designed and how well implemented your application is, there is no substitute for quality assurance test results that confirm your judgement and demonstrate the application remains unaffected by the upgrade.

Important: InterSystems recommends that each application be thoroughly tested in the upgraded environment **before** it is deployed to customers and begins processing live data.

2.1.2 Technology Previews

In recent releases, InterSystems made available software labeled as "experimental", or "preview". The purpose of these offerings is to allow customers to participate more closely in the development process by giving them access to software during a period where their feedback can influence the course of future evolution.

As long as its status remains "experimental", this software will change during the course of a release and, depending on how long it remains in that state, even from one release to another. Some of these changes may be significant enough to make the latest software version incompatible with prior versions. Therefore, customers who wish to take advantage of

such software in their application should consult with InterSystems beforehand. InterSystems will be happy to provide details on the implementation and its probable future path toward becoming a standard part of a release.

2.1.3 Field Test

Toward the end of development for each release, InterSystems makes available successively more complete copies of the final product to its customers. The notifications are published on the website and on public blogs. The purpose of this is two-fold:

- It provides an early opportunity for customers to determine how the changes and enhancement in the release affect existing applications, to report issues found, and verify those issues have been resolved.
- It also provides exposure of important proposed features (the experimental software) which may become part of future releases. Customer have a chance to try out the proposed ideas and give feedback on the usefulness of this feature for their business area.

InterSystems strongly encourages customers to plan on obtaining one or more instances of the field test and to test their application against it.

Important: InterSystems does not support upgrading from a field test version.

Unresponsive Systems

One of the goals for field test is to expose the new release to real-world operating challenges to assure its reliability. Therefore, it is possible, although unlikely, that an unanticipated sequence of events may render Caché unresponsive. In this situation, it is extremely important to gather system diagnostic information while in the hung state for InterSystems to analyze. Should an instance of Caché become unresponsive,

- Log in as an administrator
- In a terminal window, run the CacheHung script in the directory, <install-dir>/bin.

  The scripts corresponding to supported systems are:

  – Windows: CacheHung.cmd

  – UNIX®, Linux, AIX, and so on: CacheHung.sh

- Send the resulting output file to the InterSystems Worldwide Response Center. You can email the file to support@inter-systems.com, open a new problem using the WRC Online, or call the Center directly for additional assistance.

2.2 Upgrade Specifics

This section contains specific instructions applicable to this transition.

2.2.1 Classes

InterSystems recommends that customers recompile all their classes contained in each namespace. This will assure that:

- Subclasses derived from the InterSystems product library will see improved product behavior if a method call results in executing code in its superclass(es).
- All embedded SQL will use the latest versions of the SQL infrastructure.
- All projections involved in language bindings will be updated.

- All generated routines and classes will be updated.

2.2.1.1 Class compiler version utility

To assist customers in determining which class compiler version a class or classes in a namespace have been compiled with, InterSystems provides two assists

- Method – $System.OBJ.CompileInfoClass(<classname>)

  This method returns the version of the class compiler used to compile this <classname> and the datetime the class was compiled

- Query – $System.OBJ.CompileInfo(<sortby>)

  This query generates a report for the current namespace that includes all classes, the version of the compiler used to compile each one, and the datetime each was compiled. The first argument <sortby> may have the following values:

  – 0 – the time the class was compiled

  – 1 – the class name

  – 2 – the version of Caché the class was compiled in

2.2.2 Routines

ObjectScript routines, and MultiValue and Caché Basic programs do not need to be recompiled after upgrade with the following exceptions:

- Routines containing embedded SQL must be recompiled.
- MultiValue paragraphs containing queries must be recompiled.

2.2.3 Cached Queries

Cached queries are always purged during upgrade. They are recompiled and cached as needed.

2.2.4 Web Services and SOAP

It is not necessary to re-import any Web Service Definition (WSDL) files.

3

Upgrade Checklist for Caché 2017.2

The purpose of this chapter is to highlight those features of Caché 2017.2 that, because of their difference in this version, affect the administration, operation, or development activities of existing systems.

Those customers upgrading their applications from earlier releases are strongly urged to read the upgrade checklist for the intervening versions as well. This document addresses only the differences between 2017.1 and 2017.2.

Important: If you are upgrading from Caché 2008.2 or an earlier version, you cannot upgrade directly to Caché 2017.2, but must first upgrade to a version between Caché 2009.1 and Caché 2016.2. For details, see Supported Upgrade Paths in the *Caché Installation Guide*.

The upgrade instructions listed at the beginning of this document apply to this version.

3.1 Administrators

This section contains information of interest to those who are familiar with administering prior versions of Caché and wish to learn what is new or different in this area for version 2017.2. The items listed here are brief descriptions. In most cases, more complete descriptions are available elsewhere in the documentation.

3.1.1 Operational Changes

This section details changes that have an effect on the way the system operates.

3.1.1.1 Changes to Permissions Required for Studio Access to a Namespace

In this release, Studio must have a login role that allows database access to the default database for the namespace in order to connect to a namespace. If Studio does not have such a role, it will get an error when it attempts to connect to the namespace. In previous releases, Studio access was allowed if the login role had any SQL privilege to the default database for the namespace.

3.1.1.2 Feature Tracker Enabled by Default for New Installs

Feature Tracker is a software utility in Caché that gathers statistics on software module usage. The statistics track whether or not software modules are present and used in a given Caché instance. Feature Tracker sends this information via https to InterSystems weekly. These statistics help us plan development and support. The information gathered does not include any application data. In previous releases, Feature Tracker was disabled by default. In this release, Feature Tracker is enabled by default for new installs. If you are upgrading an existing installation, the upgrade preserves the existing state of Feature Tracker.

If you wish to disable Feature Tracker, you suspend the task that runs it. For details, see "How to Deactivate Feature Tracker" in the *Caché System Administration Guide*.

3.1.1.3 Parallel Dejournaling Changes the Order of Dejournaling

This release includes parallel dejournaling, which makes it possible for up to four jobs to perform the dejournaling. A database is always dejournaled by a single job, so parallel dejournaling does not impact the order in which a single database is dejournaled. But it can change the order in which one database is dejournaled relative to the dejournaling in another database. This is because databases being updated by separate dejournaling jobs are likely to be at slightly different places in the dejournaling sequence; for example, database A may contain updates made on the primary at 11:45:30 when database B is only up to the updates from 11:45:28. For details, see "Configuring Parallel Dejournaling" in the "Mirroring" chapter of the *Caché High Availability Guide*.

3.1.1.4 Changes to Journal Buffer Pool Size

In this release, the journal buffer pool size is configurable and the default has been increased to 64MB. In previous releases, the default was 8MB on non-Unicode systems and 16MB on Unicode systems. An increase in this buffer pool size can improve efficiency, but it also increases the amount of data that may be lost if a system crash occurs.

3.1.2 Platform-Specific Items

This section holds items of interest to users of specific platforms.

There are no known platform-specific changes.

3.2 Developers

This section contains information of interest to those who have designed, developed and maintained applications running on prior versions of Caché.

The items listed here are brief descriptions. In most cases, more complete descriptions are available elsewhere in the documentation.

3.2.1 Class Changes

3.2.1.1 Class Deletions

The following classes were present in the previous version and have been removed in this release:

- %Library.RunJava

3.2.1.2 Removed Methods

The following methods were present in the previous version and have been removed in this release:

- %DeepSee.UI.Dialog.ShowQuery
  - EncryptQuery
- %DeepSee.Utils
  - %ClearUpdateBuffer

- %Library.IResultSet
  - %ToDynamicArray
  - %ToDynamicObject
  - %ToJSONValue
- %Net.Remote.Utility
  - ClearPassphrase
  - GetPassphrase
  - RecordPassphrase
- %SQL.IResultSet
  - %ToDynamicArray
  - %ToDynamicObject
  - %ToJSONValue
- %SQL.StatementMetadata
  - %ToJSONValue
- Config.Shadows
  - SourceDatabasesClose
  - SourceDatabasesExecute
  - SourceDatabasesFetch

3.2.1.3 Removed Properties

The following properties were present in the previous version and have been removed in this release:

- %DeepSee.Component.Widget.controlPanel
  - isPdfEnabled
- %DeepSee.Component.Widget.widget
  - isPdfEnabled

3.2.1.4 Removed Parameters

No parameters have been removed in this release that were present in the previous version.

3.2.1.5 Removed Indices

No indices have been removed in this release that were present in the previous version.

3.2.1.6 Modified Methods

The following methods have different signatures in this version of Caché:

- %Atelier.v1.Utils.MetaData

  - old method: `Build (pDataBaseName:%String) As (none)`
  - new method: `Build (pDataBaseName:%String) As %Status`

- %CSP.UI.Portal.Dialog.EncAddAdmin
  - old method: `SaveData (What, File, Username1, Password1, Username2, Password2, Description) As %String`
  - new method: `SaveData (What, File, Username1, Password1, Username2, Password2, Description, KeyLen) As %String`

- %CSP.UI.Portal.ViewLog
  - old method: `DrawLogContent (filename) As %Status`
  - new method: `DrawLogContent (tmp) As %Status`

- %DeepSee.AbstractKPI
  - old method: `%GetKPIValue (pKPIName:%String, *pValue:%String, pKPIProperty:%String="", pSeries:%String="", &pFilters:%String, pCellContext:%String="", &pCacheKey:%String, *pPctComplete:%Integer, pParentQueryKey:%String="") As %Status`
  - new method: `%GetKPIValue (pKPIName:%String, *pValue:%String, pKPIProperty:%String="", pSeries:%String="", &pFilters:%String, pCellContext:%String="", &pCacheKey:%String, *pPctComplete:%Integer, pParentQueryKey:%String="", *pKPIStatus:%Status) As %Status`

- %DeepSee.Component.pivotTable
  - old method: `getFilterInfo (fnames, fvalues) As (none)`
  - new method: `getFilterInfo (fnames, fvalues, flabels) As (none)`

- %DeepSee.PMML.Model.AbstractModel
  - old method: `%ExecuteModelInternal (pInput:%DeepSee.PMML.ModelInput, *pOutput:%DeepSee.PMML.ModelOutput) As %Status`
  - new method: `%ExecuteModelInternal (&pInput:%DeepSee.PMML.ModelInput, *pOutput:%DeepSee.PMML.ModelOutput) As %Status`

- %DeepSee.PMML.Model.GeneralRegression
  - old method: `CalculateXBeta (pObservation:%DeepSee.PMML.ModelInput, *pXBeta, *pBestTarget:%String="", *pBestScore:%Double="", pAddZero:%String="") As %Status`
  - new method: `CalculateXBeta (&pObservation:%DeepSee.PMML.ModelInput, *pXBeta, *pBestTarget:%String="", *pBestScore:%Double="", pAddZero:%String="") As %Status`

- %DeepSee.ResultSet
  - old method: `%PrepareObject (pQuery:%DeepSee.Query.query) As %Status`
  - new method: `%PrepareObject (pQuery:%DeepSee.Query.query, &pVariables) As %Status`

- %DeepSee.Utils

  - old method: `%GetDimensionMembers (pCubeName:%String, pSpec:%String, pContext:%List, *pMembers, pMaxMembers:%Integer=100, *pMemberClass:%String, &pRelatedFilters, pCalcMode:%Integer=0, pSearchKey:%String="") As %Status`
  - new method: `%GetDimensionMembers (pCubeName:%String, pSpec:%String, pContext:%String="", *pMembers, pMaxMembers:%Integer=100, *pMemberClass:%String, &pRelatedFilters, pCalcMode:%Integer=0, pSearchKey:%String="") As %Status`

- %Exception.SystemException
  - old method: `%OnNew (pName:%String="", pCode:%String="", pLocation:%String="", pData:%String="", pInnerException:%Exception.AbstractException=$$$NULLOREF, pStack:%String="") As %Status`
  - new method: `%OnNew (pName:%String="", pCode:%String="", pLocation:%String="", pData:%String="", pInnerException:%Exception.AbstractException=$$$NULLOREF, pStack:%String) As %Status`

- %Library.GlobalEdit
  - old method: `GetGlobalSizeBySubscript (Directory:%String, StartNode:%String, EndNode:%String="", &Size:%String=0) As (none)`
  - new method: `GetGlobalSizeBySubscript (Directory:%String, StartingNode:%String, EndingNode:%String="", &Size:%String=0) As (none)`

- %Library.Routine
  - old method: `CompileAll (Flags:%String=0, IO:%String=$p, &Count:%Integer, &Errors:%Integer, MultiCompile:%Integer=1, Journal:%Integer=1, KeepSource:%Boolean=1) As %Status`
  - new method: `CompileAll (Flags:%String=0, IO:%String=$p, &Count:%Integer, &Errors:%Integer, MultiCompile:%Integer, Journal:%Integer, KeepSource:%Boolean=1) As %Status`
  - old method: `CompileSelected (Mask:%String="*.*", Flags:%String=0, IO:%String=$p, &Count:%Integer, &Errors:%Integer, MultiCompile:%Integer=1, Journal:%Integer=1, KeepSource:%Boolean=1) As %Status`
  - new method: `CompileSelected (Mask:%String="*.*", Flags:%String=0, IO:%String=$p, &Count:%Integer, &Errors:%Integer, MultiCompile:%Integer, Journal:%Integer, KeepSource:%Boolean=1) As %Status`

- %Library.SQLCatalog
  - old method: `SQLClassname (qh:%Library.SQLProcContext, table:%String) As %Library.String`
  - new method: `SQLClassname (qh:%Library.SQLProcContext, table:%String(MAXLEN=257)) As %Library.String`

- %SYSTEM.SQL
  - old method: `SetSQLStats (flag:%Library.Integer=1) As %Library.Integer`
  - new method: `SetSQLStats (flag:%Library.Integer=0) As %Library.Integer`

- %ZEN.ComponentEx.svgImageProvider

  - old method: `dumpSVGNode (e, svgDoc, src, intro, coda, NSPrefix, maxWidth) As (none)`
  - new method: `dumpSVGNode (e, svgDoc, src, intro, coda, NSPrefix, maxWidth, maxHeight) As (none)`
  - old method: `extractXSLFOSource (svg, intro, coda, maxWidth) As (none)`
  - new method: `extractXSLFOSource (svg, intro, coda, maxWidth, maxHeight) As (none)`

- %ZEN.Template.AddInWizard.SOAPWizard
  - old method: `GetSRC (filetype:%String, url:%String, sslConfig:%String="", username:%String="", password:%String="") As %String`
  - new method: `GetSRC (filetype:%String, url:%String, sslConfig:%String="", sslCheckServerId:%Boolean, username:%String="", password:%String="") As %String`
  - old method: `SaveLast (filetype, url, sslConfig="") As (none)`
  - new method: `SaveLast (filetype, url, sslConfig="", sslCheckServerId) As (none)`

- SYS.Stats.WriteDaemon
  - old method: `Sample (DaemonID:%Integer) As SYS.Stats.Resource`
  - new method: `Sample (DaemonID:%Integer=1) As SYS.Stats.WriteDaemon`

3.2.2 Class Compiler Changes

3.2.2.1 Hard Errors in delete() are Reported with %Status Value

In previous releases, if the delete() method in the class compiler encountered a hard error, it would throw the error to the caller. In this release, these errors are handled by the delete() method, which reports the error to the caller with a %Status value.

3.2.3 DeepSee Changes

3.2.3.1 Change Default to Display Mode in DeepSee Listing Selection

In previous releases, if a DeepSee field was not explicitly declared %EXTERNAL(field) or %INTERNAL(field), DeepSee would display its internal value in the listing output. In this release DeepSee will use its external value. To have the internal value used, you must wrap the field in %INTERNAL().

3.2.4 Java and Gateway Changes

3.2.4.1 Do not use JMS Gateway as Message Listener

This release removes the JMSGateway class from the Java Gateway. This class was intended as an example on how to use the JMS listener in Ensemble. If you have used this class to implement a JMS listener, you should replace this with the mechanism demonstrated in the EnsLib.JavaGateway.JMSTest class.

3.2.4.2 Object Gateway Load^%apiGTW Does Not Make err Variable Public

In previous releases, the call to Load^%apiGTW would deliberately make the variable `err` public so that it could be used in the Management Portal and other InterSystems APIs. In this release, it does not automatically make it public but instead

returns a status value. If your code depended on this previous behavior, you must explicitly define `err` as a public variable by calling:

```
Set err=$$Load^%apiGTW
```

3.2.5 .NET Language Bindings

3.2.5.1 Replace SysList, SysListUtil, and CacheListRO Classes Used With ODBCx APIs

If you use the SysList, SysListUtil, and CacheListRO Classes directly with the ODBCx APIs, replace them with other classes that provide similar functionality. Although the classes have been deprecated, they have not yet been deleted. If you use these classes but not use them with any other ADO APIs, you can defer replacing them until it is convenient. But if you use them with the ODBCx APIs, you must replace them before using with this release.

3.2.6 Object Library

3.2.6.1 %RunJava Utility Class Removed

In this release, the %RunJava utility class has been removed. If you call methods in this class, you must replace them with calls to the utility methods in %Net.Remote.Service.

3.2.7 ObjectScript Changes

3.2.7.1 Limit Number of Subscripts for Indirection

In this release, applications are limited to 254 subscripts for indirection. It is unlikely that there is any existing code doing indirection with more than 254 subscripts. Such code would create nodes that are not easily accessible.

3.2.8 SQL Changes

3.2.8.1 Changes in API to Create QuickStatement

The QuickStatement class syntax has changed and the class was moved from cache-jdbc to cache-db jar. The new syntax is

```
QuickStatement qs = QuickStatement.createQuickStatement((CacheConnection) connection);
```

If you use the previous syntax, "`QuickStatement qs = (QuickStatement) rs.getObject("**QuickStatement**");`", you must update your code to the new syntax.

3.2.8.2 Changes in %Date LogicalToOdbc and LogicalToDisplay

In previous releases if a timestamp value was passed through %Date LogicalToOdbc or LogicalToDisplay, it was not altered. As part of a fix to support dates prior to December 31, 1840, this behavior has changed. In this release if a timestamp value was passed through %Date LogicalToOdbc or LogicalToDisplay, it removes the time portion of the value and returns only the date portion. If your application logic depends on the previous behavior, you may need to modify your code. For example, if you query data in Display or Odbc mode from a table linked to Oracle and use a function like CAST(field AS DATE) or TO_DATE(...), Oracle returns a datetime value. In this release, Caché converts it to a DATE.

3.2.8.3 Enforce REFERENCES Privileges for Foreign Keys

In previous releases, it was possible to create a foreign key constraint through a DDL statement without holding the REFERENCES privilege for the referenced table. In this release, for a user to define a foreign key constraint that references

table T, the user must hold the REFERENCE privilege on table T, or the REFERENCE privilege on the column(s) of table T that are being referenced by the foreign key. If an application creates foreign keys without holding REFERENCES privileges, it will encounter errors.

3.2.9 System Changes

3.2.9.1 Change in oref Numbers Ordering

In this release, we have improved the efficiency of processing OREFs. As a consequence, the OREF number assignments, which were previously assigned in the order created are now assigned in an independent way. Consequently, if your code relies on the way in OREF numbers were assigned, you must modify the logic. The previous ordering was an implementation detail and was not a documented feature. For example, if OREF string values are used to index a Caché multidimensional array, the subscript order of the OREF string values will no longer have any relation to the order in which those OREF values were created. If an application is using OREF value order to control order of execution or order of data handling, then that application will need to be modified to use other techniques, like $INCREMENT or $SEQUENCE, to generate the desired ordering.

Similarly, this change affects the order of items in relationships (for example, the order of children within a parent). Note, however, that it is not supported to rely on the order of items in relationships.

3.2.9.2 Console Log Changes

In some cases, long console log messages that were broken into two messages in previous releases will now be written as a single message. If you have code that parses for specific messages in the console log, you should update it.

3.2.10 Web Services

3.2.10.1 SOAP Wizard Includes Option to Generate Shorter Array Names

In previous releases, the default type name for an array item includes both the item name and type name or the key name and type name, even when the names are identical. In this release, by default, if the item name and type name are the same, only one is included in the default array type name. Similarly, if the key name and type name are identical, only one is included in the default array type name. For example, in previous releases, an XML schema (or a WSDL for a web service) might include something like the following:

```
<element minOccurs="0" name="PropName" type="s01:ArrayOfSimpleObjectSimpleObject" xmlns:s01="mytypes"/>
```

As of this release, the schema or WSDL will instead include this:

```
<element minOccurs="0" name="PropName" type="s01:ArrayOfSimpleObject" xmlns:s01="mytypes"/>
```

In some cases, you can revert to the previous default behavior:

- The AllowRedundantArrayName property of %XML.Schema allows the old form in an XML schema. If true, the previous longer type names are generated.
- The *ALLOWREDUNDANTARRAYNAME* parameter of a web service class allows the old form of array name in the XML schemas in the WSDL for the web service. If this parameter is true, then the previous longer type names are generated.
- There is no way to restore the previous default of long array names, if you are generating the schema by using the **XMLSchema()** method of %XML.Adaptor. The schema would need to be created using the %XML.Schema class.

3.2.10.2 Change to Invoking Web Methods Using HTTP Requests

In previous releases you could call a Caché SOAP web service directly without making an HTTP SOAP request. This shortcut avoids using a SOAP client but it bypasses the security available using HTTP SOAP requests. This shortcut is used by the catalog and test pages that Caché generates for a SOAP web service class. InterSystems recommends that you not use this shortcut but always use a SOAP web client to generate an HTTP SOAP request to call Caché SOAP web services. If you have code that uses the %SOAP.WebServiceInvoke class in a URL to invoke a SOAP web service, InterSystems recommends that you replace this with an HTTP SOAP request generated by a SOAP client.

Although it is not recommended, you can continue to use this shortcut mechanism, but you must explicitly enable this access. To use the %SOAP.WebServiceInvoke class and the soap_method query parameter, you must:

- Enable %SOAP.WebServiceInfo and WebServiceInvoke with commands such as:

  ```
  set ^SYS("Security","CSP","AllowClass",webapplicationname,"%SOAP.WebServiceInfo")=1
  set ^SYS("Security","CSP","AllowClass",webapplicationname,"%SOAP.WebServiceInvoke")=1
  ```

  Where *webapplicationname* is the web application name with a trailing slash, for example, `"/csp/mynamespace/"`.

- Ensure that the user accessing the CSP page has USE permission for the %Development resource.

3.2.11 XML Changes

3.2.11.1 XSLT2 Transformation Returns Error Status for Fatal Errors

In previous releases, when an XSLT2 transformation encountered a fatal error, it returned $$$OK. In this release, it correctly returns an error status. If your code relies on the default error handling, you may have to modify it to handle this change. If you have overridden the default with custom error handling, the custom error handling will continue to work as it did in previous releases.

3.2.11.2 Change in XML Output for Property with Element Qualified

In previous releases, if a class with ELEMENTQUALFIED=1 had a property which is a literal datatype with ELEMENTQUALIFIED, the property was not output with a namespace specified. This change corrects this, and, in Caché 2017.2 and later releases, the property is output by XMLExport or %XML.Writer with a namespace qualification.

For example, in previous releases, the XML output for a property would be:

```
<MyProp xmlns="">0
```

In Caché 2017.2, the XML output for this property is:

```
<s01:MyProp xmlns="" xmlns:s01="http://my.namespace.com/test">
```