Caché Release Notes

Version 2012.1
30 January 2012

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

Caché Release Notes
Caché Version 2012.1 30 January 2012

This book was assembled and formatted in Adobe Page Description Format (PDF) using tools and information from the following sources: Sun Microsystems, RenderX, Inc., Adobe Systems, and the World Wide Web Consortium at www.w3c.org. The primary document development tools were special-purpose XML-processing applications built by InterSystems using Caché and Java.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Customer Support

Tel:	+1 617 621-0700
Fax:	+1 617 374-9391
Email:	support@InterSystems.com

Table of Contents

About This Book

This book provides information on the major features that have been added to Caché with each release.

It contains the following sections:

- Caché 2012.1 Release Notes
- Caché 2011.1 Release Notes
- Caché 2010.2 Release Notes
- Caché 2010.1 Release Notes
- Caché 2009.1 Release Notes
- Caché 2008.2 Release Notes
- Caché 2008.1 Release Notes
- Caché 2007.1 Release Notes
- Caché 5.2 Release Notes
- Caché 5.1 Release Notes

Each presents the major features in that release. And a more detailed Table Of Contents.

The following books provide related information:

- Caché Upgrade Checklists discusses changes made to each release that may adversely affect existing Caché applications.
- Supported Platforms lists the platforms 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 Advanced Security 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.

1

New and Enhanced Features for Caché 2012.1

The following major new features have been added to Caché for the 2012.1 release:

- Rapid Application Development
  - iKnow Technology
- Performance and Scalability
  - Improvements To Stream Performance
- Reliability, Availability, Maintainability, Monitoring
  - Manage ZEN Report Render Servers
  - System Monitor
  - Task Manager Improvements
- Security
  - CSP Gateway To Caché Over SSL
  - Web Services - Secure Conversation
- Other
  - Zen And HTML5

In addition, many more localized 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.

1.1 Rapid Application Development

1.1.1 iKnow Technology

iKnow is a new technology addition to Caché that considerably enriches the ability of applications to analyze, handle, and use unstructured (textual) data. Without needing any predetermined expertise or knowledge about the data, iKnow automatically discovers the most important information locked in your unstructured data and opens it up for automated interpretation and exploitation.

Classic approaches to unstructured data analysis typically use keyword-based searching to match possible word groups to pre-built dictionaries and language models. Keeping this data current with the text requires continued, and often intensive, effort and ongoing maintenance.

The vision behind the iKnow approach is that unstructured data is composed of two different types of elements: concepts and relationships (expressing links between concepts). iKnow automatically discovers this information in your unstructured data and opens it up for interpretation and exploitation by end users, business intelligence analysis, and business processes. It can be used for any situation where there is the need to automatically transform unstructured data into structured views of concepts and how they relate topics to one another.

The iKnow technology is accessed through an easy-to-use interface that consists of a set of system classes defined in the %iKnow package. The capabilities are exposed in three different ways: as a set of Objectscript methods, a set of SQL stored procedures, and a set of web services.

1.2 Performance And Scalability

1.2.1 Improvements To Stream Performance

The %Library.GlobalBinaryStream, %Library.GlobalCharacterStream, %Stream.GlobalBinary, and %Stream.GlobalCharacter stream implementations have been optimized for performance by reducing the amount of data copied during certain operations, and by leveraging CACHETEMP-based storage more effectively. In benchmarking heavy stream-based activity, the new stream implementation shows up to a factor of 1.5 improvement over previous versions.

1.3 Reliability, Availability, Maintainability, Monitoring

1.3.1 Manage ZEN Report Render Servers

This version introduces a new background process (external to Caché) that will be triggered automatically if you are generating Zen Report PDFs. This process instantiates a Java Virtual Machine (JVM) that will run FOP, the PDF generator from Apache. Caché will launch this process when a request comes in for a PDF based report, and send it information about the material to be processed.

The process will remain running in memory until it is explicitly terminated. More than one server process may be in operation at the same time. Caché will configure the process(es) using the settings defined in the Management Portal for the Zen Report Render Server. An administrator can specify what port each renderer will listen for requests on, what port(s)

heartbeat monitor will keep track of, and logging details. If Caché detects a process crash, or a failure in the heartbeat monitor, then it will automatically relaunch this process.

Each server process is configured from the Management Portal parameters available at the time of its launch.

1.3.2 System Monitor

The Caché System Monitor functions as a Multivariate Process Control System for monitoring a Caché system and alerting when it is not running within the statistical boundaries of a "standard" system. The definition of "standard" is defined for each instance based on the application and user workflows in a given Caché environment. Deviations from the norm are measured using the WECO (Western Electric COmpany) statistical probability rules. The System Monitor provides reporting and alerting based on outlier and non-standard events as defined by the WECO rules.

1.3.3 Task Manager Improvements

This release adds a number of improvements to support email notifications, and consistency adjustments. Customers can now specify a port number for the SMTP connection, and use an API to programmatically access task manager information.

1.4 Security

1.4.1 CSP Gateway To Caché Over SSL

With this release, applications may now request a secure connection between the CSP Gateway and the Caché instance it connects to. This adds an important security layer for connections where the CSP Gateway does not reside on the same machine as the Caché instance.

1.4.2 Web Services - Secure Conversation

Many Web services applications accommodate frequent communication between the service and client. When this communication needs to be secured end-to-end using WS-Security, the application encounters additional overhead because WS-Security uses public key encryption to secure each message separately.

To mitigate this overhead, the Web services community has introduced WS-Secure Conversation. WS-Secure Conversation moves the overhead from securing each message to a single handshake. Once the secure session is established, the service and client enter a secure conversation until the session's expiration.

This release provides support for WS-Secure conversations in Caché.

1.5 Other

1.5.1 Zen And HTML5

With this release, InterSystems has added logic to allow all Zen pages served from its products to produce HTML5 output. Whether HTML5 output is produced or not is controlled by the setting of a global, *^%ISC.ZEN.cssLevel*.

If the value of this global is set to 3, all Zen pages written by InterSystems that are served to browsers that support it will produce HTML5 output (interpreted in strict rather than quirks mode). If the global is missing or has the value 2, HTML output will be the same as in 2011.1. The default for new installations and upgrades is to retain the previous behavior.

Those applications that wish more fine-grained control over HTML5 production can override the method, **%OnDetermineCSSLevel()** for a page or subclass to return the value 3. Certain pages in the management portals work in this mode already; and all will beginning in 2012.2.

Generating HTML5 will also automatically invoke the native SVG renderer built into Internet Explorer 9, bypassing the Adobe SVG plugin should it be installed.

2

New and Enhanced Features for Caché 2011.1

The following major new features have been added to Caché for the 2011.1 release:

- Rapid Application Development
  - Multiple Session Callback Events
  - WebStress Testing Facility
  - New DeepSee Implementation
- Performance and Scalability
  - Improved Class Compiler Performance
  - Compilation Using Multiple Jobs
  - Support For Large Routines And Classes
  - Journaling Additions
- Reliability, Availability, Maintainability, Monitoring
  - Caché Management Portal Improvements
  - Mirroring Enhancements
  - Caché Monitor History Database
- Security
  - Web Services Control Separate From CSP Control
  - Two-Factor Authentication For CSP
  - Web Service Licensing
  - Managed Encryption Keys

In addition, many more localized 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.

2.1 Rapid Application Development

2.1.1 Multiple Session Callback Events

In previous versions, the only user-defined Session Events triggered at the beginning, end or timeout of a CSP session occurred based on the last CSP application accessed by that user. In this version, this behavior has changed. Caché will now execute the SessionEvent logic for the current CSP application, plus the most recently accessed CSP Application that was used by this session prior to the event, if more than one application was accessed.

2.1.2 WebStress Testing Facility

This version introduces a new core utility called WebStress. InterSystems has used this tool in prior releases to record, randomize and playback HTTP-based scripts against various applications for the purpose of QA, scalability, and network load testing. The tool runs on a Caché or Ensemble system on any supported platform and can test any web-based application. It includes additional hooks required for correctly benchmarking CSP- and Zen-based applications making use of hyperevents. For recording scripts, users must employ a supported browser and the ability to define a proxy server.

2.1.3 New DeepSee Implementation

This release of Caché introduces a new version of DeepSee (previously referred to as "DeepSee II") with the following improvements:

- Data Modeling

  Data modeling has been simplified. This version uses Caché classes that reference application transactional classes. Therefore, there is no need to modify application classes before use as in the preceding version. DeepSee models are defined via these reference classes and can be edited using the DeepSee Architect or Studio. Furthermore, data models now support many MDX concepts including multi-level hierarchies.

- Query Engine

  This version of DeepSee uses the MDX query language for all queries. Its query engine has been optimized to support parallel query execution which takes advantage of the power of multi-core architectures. Multi-level result caching improves query performance by retaining the results of queries so the results can be used when the queries are run again.

- User Interface

  Built with the InterSystems Zen technology, the new DeepSee user interface supports multiple browsers including IE, FireFox, and Chrome. Control of the user interface is done via the DeepSee option on the Management Portal. The options include: Architect for creating DeepSee data models, Analyzer for exploring the data, User Portal for creating and viewing dashboards.

2.2 Performance And Scalability

2.2.1 Improved Class Compiler Performance

As a result of changes we have introduced in previous versions, and by moving performance-critical components into the system level, InterSystems has noticeably improved the performance of the class compiler.

2.2.2 Compilation Using Multiple Jobs

In addition to the gains from the Improved Class Compiler Performance in this release, Caché can now be directed to use multiple processes for the compilation of classes and the import of XML files. Users can specify to either use half of the available processor cores, or specify the number of cores that should be used.

2.2.3 Support For Large Routines And Classes

Large Routines

In prior releases, the maximum size of a routine was 64KB. Starting with this release, the maximum routine size has been extended to 8MB. For routines larger than 32KB, Caché will use up to 128 64KB routine buffers to hold the routine. These buffers will be allocated and managed as a unit.

The class compiler and the SQL processor have been changed to use this new limit. Customers can take advantage of this improvement merely by recompiling.

Large Classes

Beginning with this release the system now supports a larger class descriptor. Among the consequences are that classes now can contain a larger number of members to be declared in the class. The limits on class inheritance depth, and the number of superclasses allowed have also been defined. A complete list of the applicable bounds can be found in the Using Caché Objects documentation.

Please consult the Upgrade Checklist for further information.

2.2.4 Journaling Additions

This version of Caché now provides an API for journal restore. See the Journal.Restore class for information on how to use it. In addition, the process Id (PID) is now once again part of each journal record; journal restore in this release has been changed to deal with this difference in format across release boundaries.

2.3 Reliability, Availability, Maintainability, Monitoring

2.3.1 Caché Management Portal Improvements

The Management Portal now provides access to all functions using one interface, including DeepSee and Ensemble (for Ensemble installations). By providing a new path to each of the functional components, there is now a mechanism to specify access control on each navigational option and granular control for security-critical operations. In addition, users can now specify the most commonly used areas as “favorites” for even faster navigation.

2.3.2 Mirroring Enhancements

In this release, several enhancements have been added to mirroring:

- Asynchronous mirror members now purge mirror journal files that have been applied locally and are no longer needed
- The mirroring communication / data transfer process has been optimized for performance by sending larger chunks of data from the primary to the backup failover member
- The mirror Virtual IP (VIP) now supports IPv6 addresses

2.3.3 Caché Monitor History Database

The Caché Monitor History Database introduces a facility to capture and analyze historical system statistics such as performance metrics, and errors reported. It supplies a baseline for analyzing performance anomalies and provides historical data to facilitate capacity planning.

A default set of metrics is defined and the schedule for capturing these metrics can be defined by the user. These metrics are fully SQL-enabled, and an API is provided to query the results stored in the database.

2.4 Security

2.4.1 Web Services Control Separate From CSP Control

In this release, for each web application (formerly known as CSP application), users can specify if CSP and or Web Service access is enabled as part of the web application definition.

2.4.2 Two-Factor Authentication For CSP

In this version, InterSystems has broadened the use of two-factor authentication, introduced with 2010.2, to also be used with CSP applications. If enabled, users will, after successful authentication, be challenged to enter an additional code, which has been separately transmitted to their mobile device.

2.4.3 Web Service Licensing

Beginning with this release, Caché will now consume a license unit for anonymous connections, and will hold this license unit for a grace period of ten seconds, or for the duration of the web service connection, whichever is longer.

Web service connection with named users (login), already consumed a license unit, and there is no change for these type of connections.

2.4.4 Managed Encryption Keys

Based on the existing Caché implementation of data-at-rest (for example, database encryption) keys, this release enables application developers to use the same strong keys, and key management capabilities on more granular data. Managed encryption keys are loaded into memory and applications refer to these keys via a unique key identifier, therefore protecting access to the key itself.

The system is designed to load four keys into protected memory. In addition, Caché now provides a new encryption function which will embed the key identifier in the resulting cipher text. This enables the system to automatically identify the corresponding key, and allows application developers to design re-encryption methods, which are completely transparent to the application, without causing any down time.

This new mechanism is designed to encrypt special data elements (such as credit card numbers), and may or may not be used in conjunction with database encryption.

3

New and Enhanced Features for Caché 2010.2

The following major new features have been added to Caché for the 2010.2 release:

- Rapid Application Development
  - Java Event Processing
  - WS Policy
  - Soap Configuration Wizard
  - MultiValue Indexing Improvements
- Performance and Scalability
  - DB Expansion Improvements
  - Large Local Arrays
  - Improved Dejournal Performance
  - $PIECE Performance Improvements
- Reliability, Availability, Maintainability, Monitoring
  - Caché Database Mirroring
  - WS-Management
  - Database Compaction
  - Cache Database
  - Caché Database Blocksize Conversion
- Security
  - Two-factor Authentication
  - Support for OpenAM
  - Encryption Of Shadow Files

In addition, many more localized 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.

3.1 Rapid Application Development

3.1.1 Java Event Processing

This version of Caché introduces several new high performance Java integrations. Each is enabled by the modular in-process Caché kernel. Java can now communicate with Caché on the same computer process-to-process enabling extremely high data access rates.

A standard JNI (Java Native Interface) is used for the Java and Caché integration. In Java JDK 1.6 and beyond, we use the Java NIO features to leverage the evolution of speed and stability in JNI and thus provide:

- MDS – Multi-dimensional data storage

  This is a raw low-level API to access Caché globals. It is the fastest way to do data access (store / retrieve) between Java and Caché. The API provides support for transactions, locking, iterators, and other features; however, it does not directly enable SQL or object support. The API can be used in multi-threaded environments.

- XEP – Extreme Event Persistence

  XEP provides a lightweight Object API on top of MDS. It enables an application to use a lightweight restricted SQL dialect for fast queries and also provides traditional full SQL in support of complex queries. XEP makes use of the transaction and multi-threaded support built-in via MDS.

- In-Process JDBC

  This feature enables JDBC to run over JNI instead of TCP when the Java virtual machine and Caché are running on the same machine.

3.1.2 WS-Policy

To complement the Caché implementation of WS-Security, and to support general security requirements, this release now provides the most common elements of the WS-Policy and allows users to easily define policies for existing Web Services.

3.1.3 Soap Configuration Wizard

This wizard assists in the creation of a configuration class that applies to a selected Web Service or Web Service client class. This configuration contains WS-Policy expressions that describe the capabilities and requirements of that service or client. These expressions can refer to WS-Security, WS-Addressing, and MTOM.

3.1.4 MultiValue Indexing Improvements

This version of Caché contains major changes to ensure better coordination between the MV index verbs (for example, CREATE.INDEX) and PROTOCLASS. The core property definition code from PROTOCLASS and CREATE.INDEX has been combined into a common class method, %SYSTEM.MV.createMVProperty to ensure that the same property definition is created by either facility. In addition, all functions, commands and verbs that use the name of an index will now use the real name of the index as a preferred alternative to the mangled name generated from the name of the source attribute. This includes the functions:

- INDICES(fv), and INDICES(fv, name)

and the commands:

- BSCAN

- OPENINDEX

Furthermore, errors in any of these should now produce better diagnostic messages.

CREATE.INDEX and **PROTOCLASS** will both add a new parameter to any properties that they create so that **DELETE.INDEX** and re-execution of **PROTOCLASS** know that the property was a generated property and can be modified or deleted by them.

3.2 Performance And Scalability

3.2.1 DB Expansion Improvements

Previously, the Expansion Daemon (EXPDMN) would expand a database by allocating new blocks in 64KB chunks. While this algorithm worked well for smaller expansions, InterSystems found that larger expansions (>50 MB) would take longer than expected; in some cases, it could interfere with the read performance on the system.

Now, for expansions larger than 1MB, the EXPDMN has been enhanced to not only issue larger writes, but to use a sliding-scale expansion algorithm. The new algorithm performs writes with the following schedule: two 1MB chunks followed by one 2MB chunk followed by 4MB chunks till the requested expansion is complete.

3.2.2 Large Local Arrays

In previous releases, the performance of operations on local arrays could worsen as the array grew larger. This version of Caché implements a highly optimized algorithm for handling large local arrays that greatly speeds up performance when saving and retrieving values from such large in-memory (local) arrays.

3.2.3 Improved Dejournal Performance

Dejournal performance has been improved in this version of Caché by reducing the amount of contention between dejournal pre-fetcher jobs. This is achieved mainly by providing each pre-fetcher job a larger list of journal entries to process. In addition, each pre-fetcher process now works on approximately 100 chunks of the journal file at a time.

3.2.4 $PIECE/$FIND Performance Improvements

The $PIECE and $FIND functions have been enhanced to speed up the process of finding the delimiter or the search string within a string by utilizing SSE2 instructions present on all x64 processor chips

3.2.5 ZEN Reports Using HotJVM

In this release, the handling of the Java Virtual Machine (JVM) for Zen Reports has changed. In prior versions, Caché would

- Instantiate an instance of a JVM in the process handling the request
- Run the rendering application in this JVM
- Collect the final PDF
- Terminate the JVM

Beginning with 2010.2, Caché starts a dedicated back-end process along with the other CSP server-side daemons. This process instantiates a Java Virtual Machine (JVM) and handles all report generation tasks returning the result PDF. The JVM is never closed. This approach (avoiding the startup and shutdown costs) results in significant reductions in the time needed to produce a report.

3.3 Reliability, Availability, Maintainability, Monitoring

3.3.1 Caché Database Mirroring

Traditional availability and replication solutions often require substantial investment in infrastructure, deployment, configuration, and planning. Caché database Mirroring is designed to provide an economical solution for fast, reliable, rapid, robust, automated failover between two systems, making mirroring the ideal automatic failover and high availability solution for the enterprise.

Caché Mirroring not only provides an availability solution for unplanned downtime, but also offers the flexibility to incorporate planned downtimes (for example, Caché configuration changes, hardware or operating system upgrades) on a particular system without impacting the overall SLAs for the organization. Combining ECP application servers with Caché Mirroring provides yet an additional level of availability; application servers simply treat a failover as an ECP server restart, and allow processing to continue on the new system once the failover is complete, thus greatly minimizing workflow and user disruption.

By maintaining independent components on the two failover systems, Caché Database Mirroring also reduces the potential risks associated with physical replication, such as out-of-order updates and carry-forward corruption, which are sometimes present in traditional shared-resource failover technologies.

Caché Mirroring also introduces the mirror shadow which can be configured either as a reporting shadow or a disaster recovery (DR) shadow – a reporting shadow can be used for enterprise-wide reporting and business intelligence (BI) purposes; a DR Shadow, which is an up-to-date read-only copy of the production system, can be used as an integral part of an enterprise disaster recovery and business continuity plan.

3.3.2 WS-Management

This release adds several Web Service methods to allow monitoring of remote Caché systems via Web Services. The methods are all contained in the SYS.WSMon.Service class, which also includes the details of the implementation.

3.3.3 Database Compaction

This version of Caché includes a Database Compaction utility which compacts a database file by moving free-space from within the file to the end of the file. Once this process is complete, it is possible to run the existing Database Truncation utility to return this free-space to the underlying file system.

3.3.4 Cache Database

In this version, InterSystems adds a new database to support operational storage requirements of the engine itself. The Cache database will be used to store information like cached queries, or session-related information. At startup, this database is mounted as read-write; no customer application should directly interact with it. Please refer to the System Administration information for further details.

3.3.5 Caché Database Blocksize Conversion

This release offers a new class method, **SYS.Database.Copy()**, that customers can call as part of their application deployment scripts. It allows the specification of the database blocksize of the target database, and is a fast and efficient way to upgrade database files from the now-deprecated 2K block format. For details and options refer to the **class documentation**.

3.4 Security

3.4.1 Two-factor Authentication

Starting with this version of Caché, InterSystems provides two-factor authentication as an added security feature during login for client/server applications, console, and terminal. With two-factor authentication, after authentication using the selected mechanism (such as Kerberos or Caché login), Caché sends a security token to the registered mobile phone of the user. To gain access to Caché, the user must then enter the token at a prompt. This adds the possession of the registered mobile phone as a second factor, along with the secrecy of the user's password as the first.

3.4.2 Support for OpenAM

InterSystems has added support to allow Caché to use the OpenAM single-sign on (SSO) component. By using this feature, users that have already successfully authenticated do not have to re-authenticate. InterSystems has demonstrated interoperability with the OpenAM web policy agent, which allows Caché to determine the centrally authenticated ID of a user from the REMOTE_USER environment variable.

3.4.3 Encryption of Shadow Files

This augments the data-at-rest encryption provided by Caché. Journal data on the shadow will now also be encrypted if the shadow system is set up to use journal encryption. The primary server and the shadow server do not need to use the same key to encrypt data-at-rest (for example, database files). Each system will use the local key for this form of protection.

4

New and Enhanced Features for Caché 2010.1

The following major, new features have been added to Caché for the 2010.1 release:

- Class Compiler And Routine Dispatch Improvements
- Java Dynamic Object Interface

Furthermore, this version of Caché has been improved and enhanced in the following areas:

- Rapid Application Development
  - Dynamic SQL
- Performance and Scalability
  - ECP Scalability Improvements
- Reliability, Availability, Maintainability, Monitoring
  - DataCheck
  - Compression During Shadowing
  - ECP Responsiveness Metric
- Security
  - Separate Delegated Authentication and Delegated Authorization
- Documentation
  - Removal Of $ZUTIL Documentation
  - Change In PDF Page Size
- Planned Changes
  - Database Extents Deprecated
  - DCP, DDP And LAT Deprecated

In addition, many more localized 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.

4.1 Major New Features

4.1.1 Class Compiler And Routine Dispatch Improvements

Version 2010.1 contains many improvements to the performance and reliability of our object implementation. Many of them are changes to the underlying implementation and therefore invisible to developers. Their effects are evident as much improved compilation times – some early testers report up to twice as fast - and faster run-time performance. The class compiler has been improved to support richer development options, better ease of use and faster compile times. The corresponding object runtime and dispatch mechanisms have also been enhanced for better performance and scalability. Version 2010.1 now supports runtime superclass resolution, system implementations for commonly generated property methods (Get, Set, isModified, setModified), and a shared class cache.

There are new compiler and class dispatch mechanisms that avoid errors associated with compiling a class while instances of the class are in use. The compiler also has improved logic for determining when changes to a superclass require recompilation of subclasses; this avoids unneeded recompilation. Details may be found in the material on the Caché Object Model.

4.1.2 Java Dynamic Object Interface

This version of Caché introduces a new high-performance interface for Java programs that provides the following:

- Multidimensional Storage (MDS) API

  Java applications may now access and manipulate Caché globals directly via a JNI (Java Native Interface) without requiring a conversion to Java objects. It enables the Java application and Caché to execute in the same process space. Among the advanced features of Caché supported by this interface are:

  – Locking (incremental, non-incremental, exclusive, shared ...)

  – Transactions

  – Iterating over globals represented in the multidimensional data structure

  – Automatic parsing of Java arrays or strings to create or update Caché $LISTs

- Java Dynamic Object API

  The API provides very fast Java access to objects stored in Caché via an in-process connection based on JNI, the Light C++ Binding, and Caché callin. The API is dynamic; the classes accessed need not be known at Java application compile time, and no separate code generation step is required.

- In-process JDBC

  This enables JDBC to use a non-TCP/IP connection to Caché like the Dynamic Object API. Applications can execute SQL queries and other SQL statements via JDBC, and in the same transaction context as methods of the Java Dynamic Object API.

4.2 Rapid Application Development

4.2.1 Dynamic SQL

This release of Caché includes support for a new interface for dynamic SQL, which allows users to define and execute SQL statements at runtime. Dynamic SQL is now implemented through the %SQL.Statement and %SQL.StatementResult classes. %SQL.Statement includes support for the following methods:

- **%New** – for instantiating a new statement object
- **%Prepare** – for preparing an SQL statement, including one that supports runtime parameters
- **%Execute** – for executing a prepared statement

Execution of a dynamic SQL statement creates a result object, which is an instance of the %SQL.StatementResult class. %SQL.StatementResult can hold simple data, a single result set, or multiple result sets, depending on the SQL statement that was executed. It includes functionality for iterating through a result set, handling errors, and so on; it also includes functionality for examining statement metadata, manipulating the form of data for display, and related operations.

For more information on dynamic SQL, see the chapter *Using Dynamic SQL*.

4.3 Performance And Scalability

4.3.1 ECP Scalability Improvements

In prior versions, if a request buffer from an ECP application server contained a synchronous request (such as a Set, Kill, $INCREMENT, and so on) that resulted in a disk I/O, subsequent requests in that buffer would wait till the disk I/O for the synchronous request completed. Now, the ECP server process daemon on the database server will continue to service subsequent read (or get) instructions that can be serviced from the buffer, in parallel while the disk I/O is completing, thereby speeding up responses to the application server.

4.4 Reliability, Availability, Maintainability, Monitoring

4.4.1 DataCheck

The DataCheck facility provides a mechanism to compare the state of data on two systems to determine whether the two are consistent. It accounts for the situation where the data on each system may be in transition, and includes logic to re-check discrepant ranges. A typical use of this functionality would be to verify that the source and destination of shadowing are consistent.

4.4.2 Compression During Shadowing

Caché version 2010.1 introduces compression of journal data from the primary to the shadow. The source (primary) compresses a source journal block before sending it to the receiver (shadow); the receiver decompresses it upon receipt, prior to saving it to the shadow copy of the source journal file.

4.4.3 ECP Responsiveness Metric

Fast responsiveness of a Data Server is key to good performance on the Application Server. To permit measurement of response times, a new property, *ResponseTime*, has been added to the SYS.Stats.ECPAppSvr class that allows you to measure the responsiveness of the Data Server(s) that this application server is currently connected to. If there are multiple connections, the value is the overall response time for all Data Server connections. To compute the average response time, use the number of connections contained in *ResponseConn*.

4.5 Security

4.5.1 Separate Delegated Authentication and Delegated Authorization

With this release applications can now use different third party technologies for Kerberos and OS authentication using the Delegated Authentication mechanism. This allows customers to use LDAP (Lightweight Directory Access Protocol), for example, to define the Roles of a user, while using Kerberos to authenticate the user.

4.6 Documentation

4.6.1 Removal Of $ZUTIL Documentation

In this release, InterSystems provides class components (methods, or properties) that have the same information and functionality as each of the documented $ZUTIL functions. This has been done to provide a more modern interface to system services available to a wider range of languages used in Caché applications.

In accord with this transition, the documentation for the $ZUTIL functions has been excised from the ObjectScript reference book. It is now part of a separate document stored in the legacy documentation archive. No further updates to it will be made.

The $ZUTIL functions will remain available for applications to use. However, all future functionality updates which would have been done by adding new $ZUTIL functions will use a class paradigm instead.

InterSystems encourages applications to convert to the new usage. To assist in the transition, a table giving the $ZUTIL function identifier and the new equivalent is given in the conversion checklist for this release. The information will also be reproduced temporarily in the ObjectScript reference work as well.

Note: In a few instances, the $ZUTIL functionality is now provided by system variables ($NAMESPACE, and $DEVICE). Also, this information does not include those $ZUTIL functions affected by the removal of support for DDP and DCP.

4.6.2 Change In PDF Page Size

The page size for documentation PDFs has changed in the version. Previously, pages assumed a height of 9 inches and a width of 7 inches. Now they use a height of 11 inches and a width of 8.25 inches. The margins have also been been reduced slightly to 0.75 inches.

This new format permits larger tables and figures, and wider program listings. Waste paper as a result of cropping has been eliminated since the new format will fit on both A4 and U.S. letter-sized paper without need for cutting before binding.

4.7 Planned Changes

4.7.1 Database Extents Deprecated

The use of database extents was introduced in early releases to work around the fact that files sizes were severely restricted. Databases of that period often exceeded the limits on the size of an individual file. Database extents allowed a logical database to span multiple physical files.

The limits in the underlying technology are no longer relevant – operating systems, mature filesystems and volume managers currently provide support for extremely large files. Database extents have been rendered obsolete. In the interest of simplifying system management, database extents are no longer supported as of this release.

In a future version, Caché, Ensemble, HealthShare and TrakCare will no longer support database extents. A tool will be provided to assist any customers with extents to aggregate them into appropriate files.

4.7.2 DCP, DDP And LAT Deprecated

InterSystems introduced Enterprise Cache Protocol (ECP) in 2002. Since then, all of our major customers have adopted ECP to deploy scale-out architectures. Usage of DCP and DDP, earlier technologies used to network databases, has diminished considerably and is now almost exclusively used when data is shared between Caché and earlier M technologies.

We intend that, in the latter half of 2010, Caché and Ensemble releases will no longer support DCP and DDP. LAT support on Windows will be deprecated at the same time.

5

New and Enhanced Features for Caché 2009.1

The following features have been added to Caché for the 2009.1 release:

This version of Caché has been improved and enhanced in the following areas:

- Performance and Scalability
  - Class Compiler Optimizations
  - XML Performance Enhancements
  - Faster Namespace Activation
- Rapid Application Development
  - .NET Gateway
  - Studio Enhancements
  - Zen Reports
- Reliability, Availability, Maintainability
  - IPv6 Support
  - Configuration File Robustness
  - Truncate Caché Databases
  - Support For 64–Bit Macintosh
  - SNMP and WMI Shadow Monitoring
  - Management Portal Reorganization
  - Embeddable Installation
- Security
  - XML Encryption
- Distribution
  - Weblink

In addition, many more localized 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.

5.1 Performance And Scalability

5.1.1 Class Compiler Optimizations

Several improvements and changes were made to improve compile time for classes. The compiler now generates additional information that is used by later compilations (e.g. incremental) to reuse already compiled code.

Some of these improvement require changes to existing classes in order to execute on this release. Please refer to the Upgrade Checklist to assess the impact on your application.

5.1.2 XML Performance Enhancements

The internal content handler for the XML parser has been optimized to interact with Caché more efficiently, which can result in substantial performance improvements when loading complex XML documents.

5.1.3 Faster Namespace Activation

Namespace activation has been sped up in this release. The effect is most noticeable in Caché instances with many namespaces and/or complicated subscript-level mappings.

5.2 Rapid Application Development

5.2.1 Informix Stored Procedure Converter

This version adds the ability to convert Informix SPL routines to Caché class methods. The resulting class methods contain Caché ObjectScript code that replicates Informix SPL behavior, including temporary tables, deferred statement resolution, and exception handling. The converter will optionally log all converted routines, errors encountered, and the name of the class/method generated. A summary of all errors encountered during conversion is also placed at the end of the log.

5.2.2 .NET Gateway

The .NET Gateway provides an API for generating proxy classes for custom .NET components or proxy class mappings for packages such as ADO, Remoting, ASP.Net, and so on. The generated proxy classes can be accessed through Caché Basic, MVBasic, or ObjectScript from Ensemble or Caché. Applications can issue method calls out to the Microsoft Common Language Runtime (CLR) on systems located locally, or remotely over a TCP/IP connection.

5.2.3 Studio Enhancements

Caché Studio was reimplemented for this version using Microsoft Visual Studio 2008. Because of this, the look and feel of the user interface has been updated to a more modern look. The functionality provided by Studio remains the same but there are new graphics, icons and greater flexibility in how you manage your personal workspace in this version.

Additional significant work was done in 2009.1 to improve Studio performance. This version implements client side class caching and indexing that results in dramatic improvements in class load time. Each time the Studio is launched, the index is checked against the server. If no changes are detected, the classes are loaded from local cache. Classes that have been changed are invalidated and refetched from the server. This change should benefit most those who use the Studio over a network.

5.2.4 Zen Reports

The Zen Reports facility introduced in a prior version has been greatly enhanced for this version. New features include:

- The ability to use external XML data sources and XSL stylesheets
- Conditional Elements and Styles in a report
- Support runtime sorting in the report rather than predefined in the query
- Base level support for Pivot tables

For more details, please refer to Using Zen Reports.

5.3 Reliability, Availability, Maintainability

5.3.1 IPv6 Support

Caché has been enhanced to support the use of IPv6 addresses. When IPv6 is enabled, Cache accepts an IP address in either IPv6 or IPv4 format or as a host name, with or without domain qualifiers. IPv6 is supported for all TCP/IP connections. You may also wish to review a summary of IPv6 usage and references to related IETF (Internet Engineering Task Force) documents.

5.3.2 Configuration File Robustness

This release includes a number of changes to increase the robustness of the configuration file and now contains a reliable internal structure to support various editing techniques to modify the configuration file. Valid files from earlier releases will be automatically converted to the new format upon first upgrade to this version.

WARNING! Invalid CPF files may prevent Caché from installing or starting up.

Important: For those customers who subscribe to the Technical Assistance program, InterSystems provides a standalone program to check the validity of CPF files before installing 2009.1 to valid customers. The program is available from the WRC Direct page. Once you login, choose **CPF Validation** from the available options.

If you do not have a userid and password, contact InterSystems Worldwide Response Center (WRC).

5.3.3 Truncate Caché Databases

This version of Caché introduces a facility to return unused database space. It works by truncating the trailing free blocks in a database file and immediately releasing the reclaimed space back to the filesystem. The truncation can be requested via the ^DATABASE routine menu, or via the SYS.Database API.

Note: This feature is presently available on Windows and UNIX® platforms only; furthermore, it is not applicable to databases with 2KB block sizes, or databases with raw volumes.

5.3.4 SNMP and WMI Shadow Monitoring

This version of Caché adds the capability to monitor Shadow Journaling using either the SNMP or WMI interfaces. The added data objects make available the data from the System Management Portal **[Home] > [Configuration] > [Shadow Server Settings]** page, and include information about each Data Source and Destination Shadow Server connection.

The exact object names, structure, and details about the implementation can be viewed using the Caché MIB (ISC-CACHE.MIB) or MOF (IscProv.mof) files, which are installed with each Caché instance.

5.3.5 Support For 64–Bit Macintosh

This release adds Macintosh OS X 64–bit support.

5.3.6 Management Portal Reorganization

Several changes and enhancements have gone into the System Management Portal:

- Configuration

  The configuration pages have been reorganized to provide more clarity in the layout. This should assist in locating relevant information.

- SQL

  A separate page now allows for management of FILEMAN data sources.

- Databases

  An option to access the new feature that permits an administrator to reclaim freespace in a database was added. Administrators can also run Integrity check on selected globals inside a database from the Management Portal.

- Licensing

  There is now an option on the licensing page that directs the Managment Portal to display license usage by user.

5.3.7 Embeddable Installation

Beginning with this version, Caché installation on Windows will use the native .MSI installation facilities. An MSI editor, such as Orca from Microsoft, can be used to customize an installation to only install required components as well as incorporate application specific components.

In addition, there are two new packages, Config and %Installer that applications can use to create an installation manifest class. This class can be run during product installation to accomplish:

- Creation of namespaces and databases
- Definition and activation of global, routine and package mapping
- Loading of globals, routines and packages
- Compilation of loaded routines and packages
- Execution of specified routines and classes

The packages and their usage are described in the section on "Using the Caché %Installer Class" in *Caché Specialized System Tools and Utilities* .

5.4 Security

5.4.1 XML Encryption

This release introduces XML Encryption for SOAP messages sent by Caché Web services and Web clients. The implementation is based on WS-Security 1.1 and uses the EncryptedKey element in the message header to use X.509 certificates. Please refer to *Securing Caché Web Services* for further details.

5.5 Distribution

5.5.1 Weblink

Starting in 2009.1, Weblink is no longer packaged in the Caché installation kits. The latest Weblink packages for each platform are available free of charge to supported customers. They can be downloaded from the Worldwide Response Center distribution page.

These kits contain all the software components and documentation needed to install and use WebLink. Instructions for installing Weblink can be found in the \doc\ subdirectory of the Weblink kits. If you have any questions, please contact the Worldwide Response Center.

6

New and Enhanced Features for Caché 2008.2

This version of Caché has been improved and enhanced in the following areas:

- Security
  - Superserver Can Accept SSL Connections
  - Telnet over SSL for Windows
  - SQL Column-level Security
  - Web Services Security (WS-Security)

In addition, many more localized improvements and corrections are also included.

6.1 Performance and Scalability

6.1.1 Object Performance Improvements

In order to improve performance and reduce contention for method dispatch and property access, Caché now caches the locations of these items upon first reference. Subsequent requests for an item first examine the cache and, if found, avoid the effort involved in locating the requested item. This improves the speed of invocation and access to class members. It also eliminates contention on key class metadata when thousands of processes are running.

6.1.2 Language Performance Improvements

The routine compiler has been enhanced. The tokens marking the beginning of line have been removed and the bookkeeping of statements in the routines has been improved.

The net result of these changes is that compiled object code should be slightly smaller and faster. There is no longer any timing effects for the use of comments in block structures.

CAUTION: This is a significant change to the ObjectScript compiler and therefore increments the compiler minor version number. Code compiled on earlier versions of Caché will run on this version, but the reverse is not true. Code compiled on this version of Caché will NOT execute on earlier versions.

6.1.3 Support For Binary SOAP

This version of Caché enhances the SOAP facilities to allow the binary transmission of SOAP messages between Caché instances. The Web service will then support normal XML-based SOAP or Caché proprietary SOAP format over HTTP. The WSDL produced for a Cache Web service using the binary extension has been enhanced to carry the necessary information for the Cache Web client. This is also a performance improvement since the transmitted data is in a more compact form.

6.1.4 Routine Buffer Management Improvements

In prior versions of Caché, all routine buffers were the same size, configured to hold the largest routine to be executed. This release of Caché implements multiple routine buffer sizes, providing separate pools of 4KB, 16KB and 64KB buffers. In most systems, the size of compiled routines span a range; allocating several buffer pools of differing sizes provides better matching of buffers to routines sizes. This, in turn, provides more efficient use of memory, and allows the administrator to better optimize system resources.

6.1.5 Shadow Latency Improvements

Shadowing latency has been decreased in this version. Caché now pre-scans the journal file for database updates and pre-fetches the data to be applied to the database.

6.2 Rapid Application Development

6.2.1 Studio Enhancements

6.2.1.1 Studio

Studio now has an option "Track variables" which switches on a limited form of variable tracking that allows it to identify variables which are used when they don't have a value. This is called READ-UNDEFINED.

6.2.1.2 Studio Assist

The following improvements have been made to Studio Assist as aids in program development:

- When the indicator sequence for macros is typed, "$$$", Studio now displays the list of commonly-used system macro names. User-defined macros are also added to this list. Previously, Studio had displayed all defined macros in the system.
- When Studio Assist recognizes the beginning of a reference to a class parameter, (..#), it will allow a selection from among the currently defined parameters for the class.
- Studio will now also provide a list of available local variables for name completion when appropriate. It will also specially color variables used only once in the bounds of a routine or method since these may be typographic errors.

6.2.2 Zen Enhancements

In this version Zen has been enhanced. There are new menu formats and desktop components:

- Zen now supports a drag-and-drop API.
- Active groups may now be dynamically resized.
- Zen provides a "lazy" tree control that only loads the data for a sub-tree when it is expanded tree.
- A new tab control has been added.

6.2.3 Text Search Enhancements

6.2.3.1 Text Search in Long Streams

Previous versions of Caché provided the ability to search text represented as streams. However, searching streams longer than 32 KB required that long-string support be enabled. In this version of Caché that restriction has largely been lifted except for usage of %CONTAINS and %SIMILARITY.

6.2.3.2 Query Optimizations For Text Searches

The SQL optimizer has been improved for searches involving text queries. It now takes account of the fact that text strings are made of individual words that themselves can be indexed as collections.

6.2.4 Improved SQL Support for Streams

This version of Caché has improved the handling of streams as fields in SQL tables, namely:

- It is now possible to open the stream returned via a simple function that returns of OID value of the stream. This is true for stream fields which are NULL as well, though attempting to read them will result in an immediate EOF.
- The INSERT and UPDATE clauses now accept such an OID as the new value for a stream field in a table.
- The SUBSTRING function now accepts a stream as its first argument.
- The ODBC and JDBC gateways now provide this support as well.

6.2.5 Light C++ Binding

Caché has changed the semantics of the Light C++ Binding object references. In prior releases, each "open" of an object created a new reference to a copy of the instance. Now the semantics for "open" match the semantics of regular C++ binding object references; all references to the same object point to the same local storage.

This version also provides support for embedded objects and inheritance.

6.2.6 JIS2004 Support

This version extends Caché Unicode support to include characters known as surrogate pairs. These are pairs of 16-bit characters that, taken as a unit, represent more than a million Unicode characters outside the Basic Multilingual Plane (BMP). To allow applications to deal with the characters properly, this version of Caché also introduces specialized versions of the regular string functions that correctly handle surrogate pairs.

6.2.7 Subroutine-level Profiler

Caché now permits profiling at the level of individual subroutines, procedures, and functions. This feature allows the user to easily see the "busiest" routines, and then drill down to the subroutine line level to analyze and improve performance of any "hot spots." The facility extends the existing **^%SYS.MONLBL** profiler (and coverage analysis).

6.2.8 $DECIMAL

A number of improvements to numeric computation have been made in this version of Caché. These include:

- Improved support for $DOUBLE values.
- Better handling of conversions between $DOUBLE values and Caché numbers and strings.
- User control over conversion to Caché floating-point using the $DECIMAL function.

6.2.9 Updates To Third-party Software

This version of Caché contains updates to third-party software packages. The following are the version changes from 2008.1 to 2008.2 for all platforms except for OpenVMS:

Package	Version in 2008.1	Version in 2008.2
ICU	3.2	3.6
Xerces	2.6	2.7
Xalan	1.9	1.10+ (contains fixes not in the Apache release 1.10)

For OpenVMS, there are no version changes for these packages in 2008.2. The versions remain as: ICU 3.2, Xerces 2.1, and Xalan 1.5.

6.3 Reliability, Availability, Maintainability

6.3.1 CSP Enhancements

This version of Caché has several important, new features for CSP and the CSP Gateway:

- It is now possible to enable HTTP authentication programmatically from within Caché.
- This version supports the new native module interface to Microsoft Internet Information Services (IIS) version 7.
- The CSP Gateway now supports connection pooling when using Apache.
- CSP now supports communication using HTTP version 1.1
- The CSP Gateway now supports transmission of messages greater than 32 KB in length via "chunks."
- Updating the CSP password via the System Management Portal now automatically updates the password value in the CSP.ini file.
- In prior versions, requests longer than 32 KB were converted to streams. In this version, they are converted only if the length exceeds the maximum string length. That is, if long strings are enabled, the conversion to a stream will only happen if the request exceeds the maximum long string length.

6.3.2 Improved National Language Support

The administration of NLS has been extensively reworked in this version of Caché. A new page has been added to the Management Portal to provide a convenient means of administration, and the %SYS.NLS.Table, %SYS.NLS.Format, %SYS.NLS.Locale and %SYS.NLS.Device classes now contains the functionality of the previous NLS utilities. See the Windows installation documentation for an example. One can also use the line-oriented routine, ^**NLS**, from a terminal connection.

The Visual Basic-based NLS management application has been removed. The corresponding replacement page is now part of the Management Portal at **[Home] > [Configuration] > [NLS Settings]**.

6.3.3 Improved UNIX® Installation

The **cinstall** installer for UNIX® has been updated. It is now possible to embed a Caché UNIX® installation within a partner product and to provide the responses needed by the Caché sub-installation in a script file. This allows a seamless installation of Caché as part of a containing product.

6.3.4 Per Socket Keep Alive

This version of Caché refines the TCP keep-alive timeout for Windows and Linux down to individual processes (rather than relying on the machine-wide setting). This makes it possible to detect lost connections due to network failures, VPN trouble, and so on in a more reliable fashion.

6.3.5 Improved Protection Against Starting Multiple Copies of Caché

Under some circumstances, Caché database locking proved insufficient to prevent multiple instances of Caché from being started against the same databases. This version has added safeguards on UNIX® systems against that happening.

6.3.6 New Archive Interface Classes

This version of Caché now contains a package, %Archive, that permits connection to an external service for long-term document retention. The package contains classes that allow for the definition of content, %Archive.Content, as well as the management of the initiation of the data transfer, %Archive.Session.

In this first release, the archive interface has been tested for use with the EMC Centera™ servers.

6.4 Security

6.4.1 Superserver Can Accept SSL Connections

The Caché superserver has been enhanced to allow communications using either the Secure Sockets Layer (SSL, version 3) or Transport Layer Security (TLS, version 1). Ports can be configured to require the use of SSL or TLS by requesting clients.

The clients that have been upgraded to use this communications capability are:

- Shadowing (configured and enabled via a command-line utility)
- JDBC and Java binding

See the notes in the Upgrade Checklist for further details.

6.4.2 Telnet Over SSL for Windows

Caché will now accept Telnet connections if the superserver is configured to use SSL and the %TELNET/SSL configuration has been defined. This is done via the Management Portal.

6.4.3 SQL Column-level Security

Caché now permits access control to be applied to SQL table columns. The privileges control the ability of a user to select, insert, and update column values. This allows one to limit access by designated users to only specified columns in a table. See the SQL verbs GRANT, REVOKE, INSERT, UPDATE, and other related verbs for an explanation.

6.4.4 WS-Security 1.1

Caché now gives applications the ability to use WS-Security 1.1 to protect web services messages. This capability permits the canonicalization of XML, and the generation and verification of digital signatures for message content according to the X.509 standard (X.509 addresses public key and privilege management infrastructure).

WS-Security is unrelated to HTTP basic access authentication which Caché already supports in this area. The WS-Security support is for UsernameToken Profile with clear text password, as described in the following document:

`oasis-200401-wss-username-token-profile-1.0.pdf`

Online at the Organization for the Advancement of Structured Information Standards (OASIS) site:

`http://docs.oasis-open.org/wss/2004/01/`

InterSystems recommends that you use SSL with this profile because the password is transmitted in clear text. Support for SSL is documented in the "Using SSL/TLS with Caché" chapter of the *Caché Security Administration Guide*. InterSystems also recommends using a SOAP/XML gateway for security purposes; for further details, see *Securing Caché Web Services*.

7

New and Enhanced Features for Caché 2008.1

The following features have been added to Caché for the 2008.1 release:

- Caché Zen
- MultiValue
- Support For RIGHT and Full OUTER JOIN
- New Online Documentation Search

7.1 Caché Zen

The Caché Zen application framework makes the development of Web-based applications easier. It provides a simple way to rapidly create complex, data-rich Web applications by assembling prebuilt object components.

Zen is based on the established Caché Server Page (CSP) and Caché Object Database technologies that offer a robust, scalable, and portable platform for hosting Web applications. Zen builds upon the basic features provided by CSP: performance, data access, security, localization, and configuration.

Zen components automatically create standard HTML and JavaScript needed to render complex Web applications. They provide a common object model that is shared between the user's browser and the application logic running on the server.

7.2 MultiValue

This version of Caché extends support for MultiValue applications. Existing applications from more than a dozen MultiValue environments may be moved to Caché and run with little or no changes. Caché provides integrated support for MultiValue features such as accounts, terminal types and spooling. In addition, MultiValue applications have unimpeded access to all existing Caché facilities:

- web application development using Caché Server Pages and Zen
- language bindings to Java, Perl, Python, C++, .NET, and ActiveX applications
- web services and service-oriented architectures

- common data storage with Caché applications
- transaction processing
- enhanced security
- the ability to run in distributed, high-availability and/or replicated systems

MutltiValue data is fully available to Caché applications. In addition, Caché classes now support the use of MVBasic as an implementation language for methods along with Caché Objectscript and Caché Basic.

7.3 Support For RIGHT and Full OUTER JOIN

With Release 2008.1, Caché SQL now fully supports LEFT, RIGHT, and FULL outer join using the ON clause. Any standard operations and most Caché SQL extensions are allowed in the ON clause, including subqueries. Outer join operations may be nested, for example:

```
Table1 full join Table2 right join Table3 on Table3.x < Table2.y
     on Table1.x > Table3.x
```

Parentheses are supported wherever they are legal. For example, the previous join expression is equivalent to:

```
Table1 full join ( Table2 right join Table3 on Table3.x < Table2.y )
     on Table1.x > Table3.x
```

You may use views of any type anywhere in the join syntax in place of a table.

See the Conversion Checklist for remaining restrictions on JOINs.

7.4 New Online Documentation Search

The following changes have been made to the online documentation application, DocBook:

- Prior to this release, the online documentation created and used its own index for the documentation. In this release, the documentation now uses the %Library.Text class for indexing and searching documents. This allows automatic, incremental maintenance of DocBook word indices (so customers that load their own content do not have to rebuild indices).
- The changes implement a new Search page that uses the Caché text searching capabilities built into Caché SQL.
- The new search page is simple to use and better ranks the results of searches. For example:
  - The user interface is much simpler: you can enter a word or words; you can place double quotes around word(s) to force an exact match.
  - The application maintains a cache of recent searches and results (so you can remember what you already looked for).
  - Instead of finding each paragraph, the search finds “documents” (books, chapters, sections, and so on) and may show links to sections within such documents. This makes the results more concise and useful.
  - Search terms that appear in titles are ranked higher in value (and appear earlier in the list) than terms that appear in paragraphs or lists. Among titles, book titles take precedence over chapter titles; chapter titles rank higher than section titles; subsection titles follow sections.
- The search facility user interface is now implemented using Caché Zen classes.

8

New and Enhanced Features for Caché 2007.1

The following features have been added to Caché for the 2007.1 release:

8.1 Call In / Call Out Enhancements

The following call in / call out enhancements are included in this project:

- Applications can now call into Caché as a DLL / shared library, rather than as an executable, eliminating the need for static linking. This enhancement will be available for all platforms except OpenVMS.
- Call in is now multithreaded on some platforms. This enables multithreaded applications to call into Caché, with multiple threads effectively executing simultaneously but independently within a single Caché process. Initially, this capability will be available on the following platforms: Windows (32-bit, 64-bit Intel Extended Memory, 64-bit AMD, not Itanium), Linux (32-bit, 64-bit Intel Extended Memory, 64-bit AMD, not Itanium), and Solaris (64-bit SPARC and 64-bit AMD). Other platforms may be added in the future.
- For all other platforms, Caché is now thread safe. It can be called safely from multithreaded applications, with Caché automatically providing any necessary synchronization. While one application thread is executing within a Caché process, other threads are blocked from executing within Caché, although they may be executing other (non-Caché) application code.

8.2 New Error Handling Syntax

This project adds new Caché error handling syntax similar to that in Java, C++ and VB. It utilizes three commands: TRY, to identify the scope of an error handler; CATCH, to identify the code to execute on an error; and THROW, to explicitly signal an error.

Note: For those familiar with other language implementations, FINALLY is not supported.

8.3 Long String Support

With this project, the maximum length of local and global strings is extended to 3.6 million characters. Storage of long strings is only supported for databases using 8KB blocks.

Enabling support for long strings is done via the Management Portal at **[Home] > [Configuration] > [Advanced Settings].** The category is "Miscellaneous" and the setting name is "EnableLongStrings".

This can also be set by the function, $ZUTIL(69, 69).

8.4 Security Enhancements

- *LDAP User Management* — This enhancement enables user authentication and roles to be managed outside of Caché using LDAP.
- *Delegated Authentication* — The enhancement enables user authentication to be carried out using site-specific Caché code.
- *Row-Level Security* — This enhancement adds extended SQL row-level security capabilities to Caché. Essentially, a column in the table contains a list of roles for each row. When a query is run, the user must have at least one of the row's roles in order to see that row. Typically, the list of roles is calculated (based on other data in the table) and a method is automatically called to perform this calculation whenever a row is inserted or updated. This security column can be indexed to provide row-based security with minimum performance impact.
- *Journal Auditing* — Changes to the state of journaling (both system-wide and process-specific) are now recorded in the audit log.

8.5 SQL Gateway via JDBC

In previous releases, the Caché SQL Gateway required an ODBC driver for each “foreign” database on each Caché platform. While this is not an issue for Windows, where good ODBC drivers are available for all databases supported by the Gateway, it has been an ongoing problem for other platforms, particularly UNIX®.

To address this, with this release we will begin using JDBC, rather than ODBC, for Gateway connections. Because all of the databases supported by the Gateway have platform-independent JDBC drivers, this will significantly reduce the testing and deployment of the Gateway and Gateway-based applications.

Note: SQL Gateway is supported on all Caché platforms except OpenVMS. For Caché 2007.1, we will support both the ODBC-based Gateway (on the same platforms as 5.2) and the JDBC-based Gateway. In later releases, for all platforms other than Windows, only the JDBC-based Gateway will be supported. On Windows systems, both the ODBC-based and JDBC-based Gateway will continue to be supported.

8.6 Objective-C Binding

This enhancement provides an object binding to the Objective-C language on Mac systems.

8.7 New Distribution Mechanism for C++ Binding

With this release we are changing the way that the C++ binding is packaged and delivered, in order to reduce our dependence on specific C++ compilers and libraries.

8.8 Zen

Zen is an extensible framework for rapid development of Web applications. Pages are defined via high-level XML-based definitions using a rich set of components. Zen includes a strong, easy-to-use security model; built-in support for multi-lingual applications; server- and client-side event handling; and very good extensibility.

8.9 SQL Enhancements

- *Aggregate Optimizations* — This release includes performance enhancements for SQL queries with multiple aggregates or with a combination of aggregates and GROUP BY.
- *SQL Temporary Tables* — This release adds support for SQL temporary tables.
- *Outer Joins* — Left outer joins can now be based on non-equality conditions, e.g. greater than or %startswith.
- *Text Search* — Text search enhancements include %CONTAINSTERM, use of indices for processing %SIMILARITY, and multi-term substitutions in the thesaurus facility.

8.10 Enhanced T-SQL Support

This release includes enhancements in T-SQL support, including compilation and runtime performance improvements, support for statement-level triggers, exposure of Sybase-compatible system tables, improved error handling, and some additional T-SQL syntax support.

8.11 Routine Performance Enhancements

COS routine management has been extensively revised in 2007.1. As a result, the way routines are managed by Caché is now much more efficient. The amount of overhead spent invoking a new routine may be up to an order of magnitude less in this release. Furthermore, in prior releases, the number of routine buffers could become a source of contention in systems with dozens of cpus; this is nowfar less likely.

Among the changes is the addition of a new per-process cache identifying recently used routines. The cache speeds up the process of locating routines which are used frequently within the process. In addition, the cache changes the way routine buffers are managed by the system.

In prior releases, only the actual routine buffers being executed were protected against change. All others could be modified. A consequence of this was that prior routines on the call stack could be changed (for example, by recompilation). To illustrate the issue, suppose a routine, A, was executing and then called routine, B. When B started running, A was available to be changed. If A was re-compiled while B was executing, the buffer holding A became obsolete and was released. When B attempted to return to A, Caché would notice this fact and report an <EDITED> error.

With the per-process cache in place, the <EDITED> error no longer occurs. This is because all routines on the call stack of any process have their current versions locked. So the associated buffers do not become obsolete. B always returns to the proper version of its caller, A.

To carry the examination further, if B returns to A and, while A is executing, B is recompiled. The next call from A to B will execute the re-compiled version of B. Similarly, if A is re-compiled while B is executing and B then calls A, the new version of A will be invoked. When the new A returns to B and B subsequently returns, it will return to the prior version of A that was preserved in the routine buffer.

The initial size of the per-process cache (called the "routine vector") is set at system startup. The default is 256. This number can be changed via the function, **$ZU(96,29,<value>)**. The cache can be cleared of all routines not in use with **$ZU(96,31)**. Note that routines are cached by namespace; changing namespaces clears the cache. Changing the routine search path via **$zu(39,...)** also clears the cache.

Because the per-process cache causes system routine buffers to be held until they are no longer in use, it is possible for a large collection of processes to consume all available buffers. When this happens, Caché will force the reduction of each of the per-process caches by some amount, freeing unused buffers. This reduction may recur until sufficient buffers are available for sustained operation.

The command

```
cstat -s<MgrDirectory> -R2048
```

where <MgrDirectory> is the pathname of the Caché manager directory will display the routine buffer usage for running jobs in the “rbuf#” column. If the number of routines in use for most jobs exceeds 30, then the administrator should increase the shared heap size by 512 bytes times the number of expected simultaneously running jobs.

Note: By default each process may cache up to 256 routines. As the system starts running out of available buffers, it will adjust the number of cached routines allowed per process down by 10% every quarter-second. When the number of cached routines per process is 30% of the originally configured number, Caché will log a message to the console, indicating the system may no longer be running efficiently.

When the number reaches 20% of the original setting, Caché will log a warning; and at the 10% threshold will log a "severe" message to the console that indicates the system may hang.

The system will hang when it runs out of routine buffers and cannot free any for use.

CAUTION: The routines held in the cache by dead processes keep the buffers they hold in use until the dead process is removed or the system shuts down.

8.12 Journal Enhancements

With this release, a number of internal improvements have been made to the performance and reliability of synchronous journal writes. In addition, system management improvements have been made to enable journal files to be retained on shadow systems for a limited period of time.

8.13 Light C++ Binding

The Light C++ Binding is a high-performance object binding that operates "in process", i.e. the client application executes in the same process as Caché. With this binding, objects are fetched directly from the database to the client, without the requirement to maintain an open object in memory within Caché. The Light C++ Binding is available for the same platforms as multithreaded call in.

8.14 CSP Gateway on OpenVMS

With this release, support is added for the CSP Gateway on OpenVMS systems (Alpha and Itanium) running the Apache Web server.

8.15 Support for GB18030 Character Set

This release adds support for GB18030, the official character set of the People's Republic of China in the Simplified Chinese locale (chsw). Since the internal representation used by Caché is Unicode, only those code points in GB18030 that map to Unicode are supported. This includes all the 1, 2 and 4-byte sequences that map to characters in the Basic Multilingual Plane (characters U+0000 to U+FFFF; approximately 64K code points), plus all the 4-byte sequences that map to the additional Unicode planes (U+10000 to U+10FFFF; 1M code points).

This latter range is represented in Caché in UTF-16 as surrogate pairs. This is the same representation used by Microsoft since Windows 2000.

Note: There are approximately 500K code points in GB18030 that don't map to Unicode and are currently unassigned. These code points are not supported by Caché.

8.16 Other Changes

- *Increased file name length* — The maximum path length for databases and other files has been increased from 64 to 232 characters.
- *Licensing Improvements* — With this release, the value of $USERNAME can be used to identify users for licensing purposes. This enables more accurate counting in situations where the IP address cannot be used to reliably identify distinct users.
- *Double Literals* — Large numeric literals (>1E146) are now automatically stored as double values.
- *JOB Improvements* — A process started with the JOB command can now be terminated by its parent, regardless of security settings.
- *Visual Studio 2005* — This release makes use of Microsoft's Visual Studio 2005 for Windows client components. This change affects customers who make programmatic access (e.g. using call in) to Caché executables.
- *MultiNet Support* — We will support the MultiNet implementation of TCP/IP on OpenVMS Alpha systems (but not on OpenVMS Itanium). We may discover and document limitations with this software.
- *Hibernate* — The released version of Hibernate 3.2.1 contains support for Cache2007.1. For further information, or to obtain this version, please visit the Hibernate organization website.
- *Studio Improvements* — A number of smaller changes have been made to the Studio IDE:
  - The toolbar has a View WebPage button that allows the developer to see a preview of the page
  - The toolbar now has Forward and Backward buttons
  - The options dialog has been revised
  - The StudioAssist feature now works with selected XML documents, notably Zen

8.17 New Supported Platforms

The following platforms are added:

- Sun Solaris AMD
- OpenVMS 8.3 for Alpha and Integrity
- SUSE Linux Enterprise Server 10

9

New and Enhanced Features for Caché 5.2

The following features have been added to Caché for the 5.2 release:

- Jalapeño Java Persistance API
- Caché Managed Provider for .NET
- IEEE 8-byte Floating Point Support
- Direct FileMan Dictionary Converter
- Code Completion in Caché Studio
- Process-Private Globals
- Caché Journal File Encryption
- Version Checking (and Optimistic Concurrency)
- Dynamic Dispatch
- Free Text Search
- ODBC Multiple Result Sets
- WMI Support
- Enhanced Debugging Capabilities
- T-SQL Support
- Device Level SSL and TLS support
- Enhanced ECP Performance
- Enhanced Windows Cluster Resource Management
- Improved RPM Linux Installation

9.1 Jalapeño

Using our traditional object bindings, each client class has, in addition to its developer-defined properties and methods, a set of Caché supplied methods related to object persistence, e.g. Save() and OpenId(). With Jalapeño this paradigm is changed slightly by separating the persistence behavior into a separate "controller" class that can be used with any Java object.

- *Java Object Schema Import*

  As part of Jalapeño, we are adding Java Object Schema Import. Traditionally, our approach to language bindings has been to start with a Caché class definition from which Caché generates one or more "projection" classes. Java object schema import turns this around: it enables Caché class definitions to be created from existing Java class definitions.

- *Database Neutral Deployment*

  Jalapeño provides an additional feature: the ability to deploy applications on any database supported via the Caché SQL Gateway. It does this by substituting SQL requests for the object mechanisms normally used to link the Jalapeño client with the database (Caché) server. Of course the performance of this approach will be inferior to that of Caché, but it provides a convenient way to construct a database-independent application without giving up the power and performance of Caché.

9.2 Caché Managed Provider for .NET

The Caché Managed Provider for .NET supplies high performance relational and object database capabilities through a single client — something that no one else delivers for .NET. This release also provides a Visual Studio .NET plug-in that automatically generates .NET assemblies from Caché class definitions.

9.3 IEEE 8-byte Floating Point Support

Support has been added for IEEE 8-byte floating point (a.k.a. "double") values, making Caché more attractive for calculation-intensive applications.

This enhancement includes:

- A new internal data type for both scalars and list members
- A new intrinsic function to cast a value to a double
- Assembler optimizations to enable Caché to take advantage of processor-specific floating point instructions
- Object and SQL client enhancements to handle new server data types

9.4 Direct FileMan Dictionary Converter

The Direct FileMan Dictionary Converter enables Caché table definitions to be automatically created from FileMan file definitions. (Previously, the conversion was a two-step process, from Fileman to F DBMS and then to Caché.)

A more detailed description is contained in the section on "Converting FileMan Files into Caché Classes" in *Caché Specialized System Tools and Utilities* .

9.5 Code Completion in Caché Studio

Code completion simplifies editing in Caché Studio by enabling developers to pick from a list of context-sensitive choices when entering code. For instance, Studio will assist in specifying a class name when "##CLASS(…)" syntax is used or in specifying a property or method name when the type of a variable is known.

9.6 Process-Private Globals

Support has been added for process-private globals. Like local variables, they are accessible only by the process that creates them and are automatically deleted when that process halts. However, like globals they are essentially unlimited in size.

9.7 Caché Journal File Encryption

Support has been added for encryption and decryption of Caché journal files.

9.8 Version Checking (and Optimistic Concurrency)

At the object level, this feature adds automatic support for version checking. If turned on (via a class parameter) it increments the value of a version number property whenever an object is saved. You can use this set of features to implement optimistic concurrency.

9.9 Dynamic Dispatch

Dynamic dispatch enables a Caché class to respond to references to properties and methods that are not part of the class definition at compile time.

9.10 Free Text Search

Support has been added for indexing and searching textual data, specifically:

- A new %Text data type.
- A new SQL selection operator, %CONTAINS.
- Language-specific parsers for English, Spanish, French, Italian, German, Japanese, and Portuguese.

- Support for multi-word ("n-gram") indices. For instance, with an n-gram length of 1, all individual words are indexed. With an n-gram length of 2, all word pairs that occur together are also indexed.
- Support for "stemming" to map multiple forms of a word (go, goes, going, went, …) to a common root.
- Noise word filtering, to eliminate common words (a, and, the, …) from an index.

9.11 ODBC Multiple Result Sets

Support has been added to Caché ODBC for stored procedures that return multiple result sets.

9.12 WMI Support

Support has been added for Windows Management Instrumentation (WMI), the Microsoft implementation of Web-Based Enterprise Management. This enables Caché to be monitored by a variety of Windows management tools, including Microsoft Management Console.

9.13 Enhanced Debugging Capabilities

Caché debugging capabilities have been enhanced in a number of areas:

- Ability to debug macro routines, class definitions, and CSP files
- New system-level mechanism to make the debugger connection more robust
- New stack trace mechanism
- Ability to BREAK into a READ
- Additional debugging commands / options.

9.14 T-SQL Support

A number of capabilities have been added to simplify migration from SYBASE or SQL Server to Caché, including support for the T-SQL language in stored procedures. T-SQL is currently supported on 32 bit Windows, 32 bit Linux (RedHat and Suse distributions), 64 bit AIX®, and 64 bit Solaris.

9.15 Device Level SSL and TLS support

Device level support has been added for secure sockets layer communication using SSL 2, SSL 3, or TLS.

9.16 Enhanced ECP Performance

Improvements have been made to ECP and the ECP/jrnsync mechanism to enhance performance and scalability.

9.17 Enhanced Windows Cluster Resource Management

This enhancement enables Caché to be managed more easily in a Windows cluster.

9.18 Improved RPM Linux Installation

A new RPM install package has been created for Caché on Linux.

10

New and Enhanced Features for Caché 5.1

Welcome and thank you for using Caché!

This chapter provides an overview of the new and improved features in Caché 5.1.

See the Major New Features section for a description of the important new features and enhancements included in this release.

If you are new to Caché, you can refer to the Getting Started page which contains a variety of links to documentation organized by topic.

10.1 Upgrading and Installation

If you are upgrading existing applications and databases from prior versions, please read the Caché 5.1 Upgrade Checklist and the Upgrading chapter of the *Caché Installation Guide*.

For information on installing Caché, refer to the following sources:

- the *Caché Installation Guide* for Windows, OpenVMS, and UNIX®
- the list of Supported Platforms for this release

10.2 Major New Features

Caché 5.1 introduces a significant number of new features as well as enhancements to existing features. These features are focused on:

- Advanced Security
- Maximizing Scalability
- Maximizing Development Speed
- Minimizing Support Load

The new features are summarized here. For additional details, refer to the cited chapters or guides.

Caché Advanced Security

A host of new capabilities have been added to give Caché the most advanced security of any mainstream database.

System Management Portal

A new integrated system management interface, built with CSP, replaces Control Panel, Explorer, and SQL Manager. This removes the requirement for a Windows PC in order to manage Caché and, because no Caché client software is required, eliminates potential client/server version mismatch issues and simplifies management of multiple versions of Caché from a single device.

System Improvements

Caché system improvements include many new or enhanced classes and methods, plus major enhancements such as nested rollback and the ability to map class packages to namespaces.

- Nested Rollback — When nested TSTARTs are used, this enhancement enables the innermost TSTART to be rolled back, without rolling back the entire open transaction.
- Namespace Mapping for Class Packages — Namespace mapping has been extended with the ability to map class packages by name, just as routines and globals are mapped.

ObjectScript Language Improvements

The ObjectScript language now provides significantly improved runtime error reporting. Many other enhancements have been introduced, including the following items:

- New **$FACTOR** Function
- New **$LISTNEXT**, **$LISTTOSTRING**, and **$LISTFROMSTRING** Functions
- New **$ROLES** and **$USERNAME** Special Variables
- New Error Trapping Syntax
- More Efficient Code Generation
- Pattern-Match “E” Adapted For Unicode
- Faster **MERGE** Command

New Language Bindings

With this release, Caché introduces new Perl and Python bindings, as well as an improved version of the Caché ActiveX binding.

Object Improvements

The Caché 5.1 Class Library provides many new features and major enhancements.

- Index on Computed Fields — An index definition can now reference properties defined as CALCULATED and SQLCOMPUTED.
- Object Synchronization — Caché can now track records of all object filing events (insert, update and delete) for journaled classes, export the journaled object data, and synchronize it with other databases. Applications with no access to the original database can then resolve references to the synchronized objects.
- Studio Enhancements — New %Studio.Extension classes provide mechanisms for custom menus and user defined data entry. %Studio.SourceControl classes now provide enhanced source control hooks, allowing customized checkout and checkin to a source control system.

- Performance Improvements — Significant improvements have been made to the in-memory performance of relationships.
- Syntax for defining stream and collection properties has been improved, and enhancements have been made to the behavior of streams and collections.

SQL Improvements

Caché SQL support includes many new or enhanced features, including the following items:

- New SQL/XML Support Functions
- JDBC 3.0 Support
- **SAVEPOINT**: New Transaction Processing Feature
- **CREATE TABLE**: New IDENTITY Keyword
- **DROP VIEW**: New CASCADE Keyword
- **INSERT**: New DEFAULT VALUES Clause
- New RowId Counter Validation Option
- New Query Optimizer Plan Verification
- Subquery Flattening
- Enhanced Locking Behavior for Foreign Key References
- READONLY Tables and Fields
- Support for %%CLASSNAMEQ and %%TABLENAME
- CREATE BITMAP INDEX Support for Oracle Import Compatibility

Connectivity Improvements

Caché 5.1 introduces many new options for network connectivity.

- ECP Enhancements — A number of enhancements have been made to the Caché Enterprise Cache Protocol. It is now supported in shared disk cluster configurations with OpenVMS and Tru64 UNIX®.
- SNMP Support — Support for the Simple Network Management Protocol (SNMP) has been added to enable monitoring of Caché by a variety of systems management tools and frameworks.
- LDAP Client — Programmatic access to LDAP servers has been added.
- Mac OS X Server Support — Support has been added for Mac OS X as a server plus the following client components: ODBC, JDBC, Objects, CSP Gateway for Apache.

10.3 Caché Advanced Security

With version 5.1, InterSystems introduces Caché Advanced Security. This release of Caché contains a host of new capabilities that provide the most advanced security of any mainstream database. Caché Advanced Security provides a simple, unified security architecture that offers the following advantages:

- It offers a strong, consistent, and high-performance security infrastructure for applications.
- It meets certification standards.

- It makes it easy for developers to build security features into applications.
- There is a minimal burden on performance and operations.
- It ensures that Caché can operate effectively as part of a secure environment and that other applications and Caché can work together well.

See the Caché Security Administration Guide for detailed information on Caché Advanced Security.

10.3.1 Key Features

Here are a few of the more important new security features offered in Cache 5.1:

- Kerberos based Security Infrastructure

  Two Authentication models are now available. In addition to Caché Authentication (Username/Password), Cache now provides Kerberos based Security Infrastructure. Kerberos libraries are available on all supported platforms (Windows Single Sign-on for Win32/64 platforms in an Active Directory Domain = Kerberos Realm, since Microsoft uses Kerberos at the heart of their Authentication model).

- Security Management Interface

  The Caché Management Portal's web-based Security Management facility allows complete access to Users, Roles, Services, Resources (including Schemas), Auditing, and all other aspects of Caché security management.

- Security Advisor Utility

  The new Security Advisor utility makes recommendations for securing a Caché DBMS (Security settings, Applications and Auditing).

- Authentication in ODBC/JDBC

  ODBC and JDBC drivers now offer both Caché and Kerberos Authentication. Kerberos mode provides three levels of Encryption: Clear, Integrity (Source and Content Validation), and Encrypted (complete, end-to-end AES Encryption).

- Auditing Facilities

  Caché provides detailed auditing facilities that store audit information in a specially protected Audit Database. Auditing capabilities are available from an Automated/Management and Programmatic/API point of view.

- Encrypted Database Management Facility

  The new Encrypted Database facility allows you to create fully encrypted (AES, up to 256 bit) CACHE.DAT files that stay Encrypted on Disk at all times. I/O is encrypted and decrypted on the fly, with minimal performance impact. The database is encrypted with a Special Key file that is stored on removable devices (like USB Flash Drives) and must be present to mount the DB for use.

Security Advisor

To assist system managers in securing a Caché system, Caché includes a Security Advisor. This is a Web page that shows current security-related system configuration information, recommends changes or areas for review, and provides links into other system management Web pages to make recommended changes.

Caché 5.1 contains the initial version of the Security Advisor. Its function and range will expand in future versions. It is accessed through the System Management Portal at **[Home] > [Security Management] > [Security Advisor].**

InterSystems strongly recommends a review and resolution of the issues raised by the Security Advisor before allowing a secured system to attain production status.

Low-level Security Interfaces

System administrators can exercise low-level control over the security of Caché systems through two character-oriented interfaces:

- **^SECURITY** allows examination and editing of security data related to users, roles, domains, services, applications, and auditing. An overview of **^SECURITY** can be found in The CHUI-Based Management Routines.
- **^DATABASE** provides low-level management capabilities related to Caché databases. An overview of **^DATABASE** can be found in The CHUI-Based Management Routines.

Common Criteria Security Certification

Security certification is becoming an increasingly frequent requirement for government purchases, and is more and more requested for private sector purchases. Because of this, InterSystems has had Caché certified according to the Common Criteria standard. Specifically, effective February 15, 2007, Caché received certification according to the Common Criteria standard (EAL 3).

The Common Criteria provides a set of common security standards for a wide number of nations in North America, Europe, and the Far East. It provides an assurance scale from 1 to 4, where a product's rating indicates the rigor of testing to which it has been subjected; commercially available products are rated from 1 (least rigorous testing) to 4 (most rigorous). Caché is currently under consideration for a level-3 rating. Such a rating indicates that Caché can effectively serve as part of a highly secure operational environment.

10.3.2 Caché Advanced Security Concepts

Caché Advanced Security is based on authentication, authorization, and auditing:

- Authentication ensures the verification of the identity of all users.
- Authorization ensures that users can access the resources that they need, and no others.
- Auditing keeps a log of pre-defined system and application-specific events, to provide forensic information about the database activities.

Authentication: Establishing Identity

Authentication is how you prove to Caché that you are who you say you are. Without trustworthy authentication, authorization mechanisms are moot — one user can impersonate another and then take advantage of the fraudulently obtained privileges.

The authentication mechanisms available depend on how you are accessing Caché. Caché has a number of available authentication mechanisms:

- Kerberos — The most secure means of authentication. The Kerberos Authentication System, developed at MIT, provides mathematically proven strong authentication over an unsecured network.
- Operating-system–based — Available for UNIX® and OpenVMS, OS-based authentication uses the operating system's user identity to identify the user for Caché purposes.
- Caché login — With Caché login, Caché maintains a table of hashed password values for each user account; at login, Caché confirms user identity by comparing the value in the table with a hash of the password provided by the user.

Authorization: Controlling User Access

Once a user is authenticated, the next security-related question to answer is what that person is allowed to use, view, or alter. Authorization manages the relationships of users and *assets* such as databases, Caché services like ODBC access, and user-created applications.

In the most basic authorization model, there are all possible assets, a list of users, and all the relationships between the first group and the second.

Auditing: Knowing What Happened

Auditing provides a verifiable and trustworthy trail of actions related to the system. Auditing serves multiple security functions:

- It provides proof — the proverbial "paper trail" — recording of the actions of the authentication and authorization systems in Caché and its applications.
- It provides the basis for reconstructing the sequence of events after any security-related incident.
- Knowledge of its existence can serve as a deterrent for attackers (since they know they will reveal information about themselves during their attack).

The auditing facility automatically logs certain system events; it also allows you to enable logging for other system events, as well as site-defined application events. All audited events are placed in a tamper-resistant log file. Authorized users can then create reports based on this audit log, using tools that are part of Caché. Because the audit log can contain sensitive information (such as regarding positive or negative values for medical tests), running an audit report itself generates an entry for the audit log. The included Caché tools support report creation, archiving the audit log, and other tasks.

10.4 System Management Portal

Caché 5.1 now uses a browser-based interface, the System Management Portal, for system management. This new interface subsumes the functions previously distributed among Explorer, SQL Manager, Configuration Manager, and Control Panel functions of the Windows Caché Cube. In 5.1, these have been removed from the Cube.

An advantage of this approach is that it is no longer a requirement that Caché be installed on the system you use to manage an installation. Remote management of systems over the web, subject to access control established for the site, is now much easier. No Caché client software is required, simplifying management of multiple versions of Caché from a single device. Cross-release compatibility issues are minimized because both the data and its formatting information come directly from the system being managed.

See Using the System Management Portal for a detailed description of the new interface.

10.5 System Improvements

New Caché 5.1 system features and enhancements:

New Features:

- Nested Rollbacks
- Namespace Mapping for Class Packages
- New Method $SYSTEM.Util.CleanDeadJobs()
- New Class $SYSTEM.Monitor.Line
- New Method $System.Device.GetNullDevice()
- New Optional Argument for $ZF(-2)

Enhanced Features:

- Option to Filter Records before Dejournaling on a Shadow
- Callin Enhancements
- 64K Routine Buffer Support
- CVENDIAN Enhancements

10.5.1 Nested Rollbacks

This version of Caché introduces multiple transaction levels , which make it possible to roll back part of a transaction without losing all work completed to that point. When nested **TSTARTs** are used, this enhancement enables the innermost **TSTART** to be rolled back, without rolling back the entire open transaction. When two **TSTARTs** are issued without an intervening **COMMIT** or **TROLLBACK**, the transaction level ($TLEVEL) is incremented by 1 (limited to a maximum of 255). When a **TCOMMIT** or **TROLLBACK 1** is issued, the transaction level is decremented by 1. When an unqualified **TROLLBACK** is issued, the transaction level is decremented to 0, and the entire transaction is rolled back.

Transaction commands now work as follows:

- The argumentless **TROLLBACK** command works as usual, rolling back to the very top level transaction and closing the transaction.
- The **TROLLBACK 1** command rolls the current open transaction back one level. All the globals changed within this transaction will be restored, and $TLEVEL is decremented by 1. If there is no open transaction ($TLEVEL is zero) then no action is taken. **TROLLBACK 1** won't roll back globals mapped to a remote system that doesn't support nested transactions unless $TLEVEL is 1.
- The **TCOMMIT** command works as usual. In nested transactions, it decrements $TLEVEL and writes a 'PTL' (pending commit with transaction level) journal record to the journal file.
- The **TSTART** command also works as usual. In nested transactions, it increments $TLEVEL and writes a 'BT'(begin transaction) record in the journal file. If the new $TLEVEL is greater than 1, it writes a 'BTL'(Begin Transaction with level) instead of 'BT'.

Caché SQL now includes standard SQL commands that take advantage of nested rollbacks (see New SAVEPOINT Features).

10.5.2 Namespace Mapping for Class Packages

Namespace mapping has been extended with the ability to map class packages from a database to one or more namespaces, just as routines and globals are mapped. Automatic namespace mapping is provided for system classes. All the schemas that begin with '%' from %sys are mapped to all namespaces automatically. These mappings allow the user to access SQL Table, View, Procedures, and classes across multiple namespaces. For example, assume a class %Test that has the following query:

```
Select field1 From %Test
```

Without mapping, attempting to inherit from this class in a user namespace would result in error: "Table %Test not found". With mapping, the class will compile successfully in any namespace.

For detailed information, see Configuring Data in the *Caché System Administration Guide*.

10.5.3 New Method $SYSTEM.Util.CleanDeadJobs()

New class method **$SYSTEM.Util.CleanDeadJobs()** is used to roll back a dead job's open transaction (if any) and clean up the dead job's Process Table (pidtab) slot so it can be re-used.

10.5.4 New Class $SYSTEM.Monitor.Line

New class $SYSTEM.Monitor.Line is a programmer API for the line-by-line monitor (**^%MONLBL**). It allows integration with Studio, and is also generally useful as a programmable alternative to **^%MONLBL**. For details, see the Programming Interface section in the MONLBL chapter of the *Caché Monitoring Guide*.

10.5.5 New Method $System.Device.GetNullDevice()

New class method **$System.Device.GetNullDevice()** returns the name of the null device appropriate for the current operating system type (/dev/null for UNIX®, _NLA0 for OpenVMS, //./nul for Windows). It facilitates development of applications that reference the Null device, and provides an OS-independent method for obtaining the name of the Null Device.

10.5.6 New Optional Argument for $ZF(-2)

Function **$ZF(-2)** now has an optional fifth argument that specifies whether or not the spawned process ID should be stored in $ZCHILD. For example:

```
s rc=$zf(-2,"program","","",1)
s childpid=$ZCHILD
```

If the new argument is zero or not specified then $ZCHILD is unchanged, otherwise $ZCHILD is set to the spawned process ID when it is successfully spawned.

10.5.7 Option to Filter Records before Dejournaling on a Shadow

To filter journal records before they get dejournaled on a shadow, set the global node ^SYS("shdwcli",shdw_id,"filter") to the name of the filter routine (without the leading "^"). The input parameters of the filter routine are:

- `pid`: process ID of the record
- `dir`: SOURCE (not SHADOW) database directory
- `glo`: global reference in the form of global(subscripts) (without leading "^")
- `addr`: offset of the record in the journal file
- `type`: type of the record: "S" = SET, "s" = BITSET, "K" = KILL, "k" = ZKILL
- `time`: timestamp of the record

In compatible mode shadowing, the pid and timestamp parameters passed to the filter routine always have the value "". The filter routine should return 0 if the record should be skipped; otherwise the record will be dejournaled by the shadow. For example:

```
^SYS("shdwcli","MyShadow","filter")="MyShadowFilter"
MyShadowFilter(pid,dir,glo,type,addr,time) ;
i $e($qs(glo,0))="X" q 0 ;skip X* globals
d MSG^%UTIL(pid_","_dir_","_glo_","_type_","_addr_","_time,1,0) ;log
q 1
```

10.5.8 Callin Enhancements

The Callin include files ccallin.h and mcallin.h have been enhanced to merge common functionality and provide greater flexibility for building user-defined C and C++ Callin modules. Defines have been added to make building user Callin modules as independent of interface details as possible. Two features control the selection of interfaces:

#define ZF_DLL

If ZF_DLL is not defined, the Callin module is built for linking with the Caché engine. If it is defined, the module is built as a dynamic shared library using Callback and invoked through the Callout facility. This is the same define employed by cdzf.h.

#define CACHE_UNICODE

If CACHE_UNICODE is not defined, string handling functions and arguments are treated as 8-bit characters. If defined, strings are treated as 16-bit Unicode. String handling functions are available with the "A" suffix, meaning 8-bit (or ASCII), the "W" suffix, meaning 16-bit Unicode (or wide), and no suffix. In the last case the function resolves to either the "A" or "W" suffix according to the definition of CACHE_UNICODE.

New functionality has been implemented to permit NLS translation using the **CacheCvtInW()** and **CacheCvtOutW()** functions for Unicode Callin to 8-bit Caché. They will now convert data within the 8-bit character set range of the Caché engine, instead of reporting an "unimplemented" error. **CacheCvtInA()** and **CacheCvtOutA()** functions for 8-bit Callin to Unicode Caché are not currently implemented.

You can further refine 8-bit argument prototypes with the new macro **USE_CALLIN_CHAR**, which declares them as (char *) rather than (unsigned char *).

10.5.9 64K Routine Buffer Support

It is now possible to run with routine sizes up to 64K, by changing the Memory/RoutineBufSize value on the **[Home] > [Configuration] > [Advanced Settings]** page of the Management Portal from 32 to 64. The default and minimum value is still 32 (32K), but now values can be specified from 33..64 (rounded to the nearest 2K increment). Routines or class descriptors greater than 32K will be stored as two global values, the first chunk in ^rOBJ(<routine name>) as currently, and the second chunk in ^rOBJ(<routine name>,0).

10.5.10 CVENDIAN Enhancements

The cvendian database endian conversion utility has been enhanced to allow for positive identification of the desired endian orientation, or to optionally just inform the current endian orientation with no conversion. The command syntax is:

```
cvendian [-option] file1 [file2 ... file8]
```

where *option* is one of the following:

- -big — convert the database to big-endian
- -little — convert the database to little-endian
- -report — report the endian orientation of the database

The options may be shortened to their initial letter. If this is a conversion request, and the database already is of the specified endian orientation, a warning message is displayed and no further processing is done. Prior **cvendian** call formats remain supported.

10.6 Object Improvements

New Caché 5.1 object features and enhancements:

Object Enhancements

- New Option to Index on Computed Fields
- New Object Synchronization
- New Studio Extension Classes and Source Control Hooks
- New Stream Syntax
- New %SwizzleObject Class
- Extended POPSPEC Syntax

- Performance Improvements for Relationships
- Enhanced VisM OCX

10.6.1 New Option to Index on Computed Fields

An index definition can now reference properties defined as CALCULATED and SQLCOMPUTED. The property value calculation must be deterministic, always returning the same value for a given set of parameters. For example, it would be a mistake to use a function such as $Horolog, which returns different values depending on when it is called. Indexing on a property whose computation is nondeterministic will result in an index that is not properly maintained.

To support this option, properties defined as SQLCOMPUTED are now computed in Caché Objects. A new class method, Compute, is called by the property's Get method. The Compute method generates a return value by scanning SQLCOMPUTECODE for field references and converting those references to property or literal values. If the property also has SQLCOMPUTEONCHANGE, the Compute method is called whenever the property is changed.

10.6.2 New Object Synchronization

This new feature enables Caché to synchronize objects between databases. All object filing events (insert, update and delete) for journaled classes are automatically tracked. Object synchronization utilities provide methods to export the journaled object data and synchronize it with other databases. Applications with no access to the original database can then resolve references to the synchronized objects.

A new class, %SYNC.SyncSetObject, supplies methods to externalize an object and apply it to the target database. All references to persistent objects from the object being externalized are converted to GUID (Globally Unique Identifier) values. The GUID values are used to look up the corresponding object on import.

Another class, %SYNC.SyncSet, implements methods to manage the set of objects being synchronized. A 'synchronization set' is a set of externalized object values which guarantee that all object references can be resolved, either because the referenced object is in the same sync set, or because it already exists in the target database.

10.6.3 New Studio Extension Classes and Source Control Hooks

This release enhances the flexibility of Caché Studio by introducing the %Studio.Extension classes, which provide mechanisms for custom menus and user defined data entry. The %Studio.SourceControl classes now provide enhanced source control hooks, allowing customized checkout and checkin to a source control system.

When the user performs an action in Studio that may require user interaction with the server (for example, attempting to edit a document that is in source control but is not checked out), Studio now calls the **UserAction** method.

```
UserAction (Type, Name, InternalName, SelectedText, .Action, .Target, .Msg)
```

`Type` options are:

- Server defined menu item selected
- Other Studio action

`Name` is the menu item name if `Type` is a menu item, otherwise `Name` indicates one of the following options:

- User has tried to change a document that is locked in source control
- User has created a new document
- User has deleted a document

`InternalName` is the name of the document this action is concerned with.

`SelectedText` contains any selected text in the document that has focus.

`Action` returns an action that Studio should perform:

- Do nothing (this method can still perform some action such as check an item out of source control, but Studio will not ask for user input).
- Display the default Studio dialog with a yes/no/cancel button. The text for this dialog is provided in the `Target` return argument.
- Run a CSP Template. `Target` is the start page name for the template. The template will be passed the current document name, any selected text, the project name, and the namespace.
- Run an EXE on the client. `Target` is the name of an executable file on the client machine.
- Insert the text in `Target` in the current document at the current selection point
- Studio will open the documents listed in `Target`

You can define custom menus for Studio to display. Studio obtains the menus when it first connects to a namespace by running two queries, **MainMenus** and **MenuItems**. **MainMenus** returns the list of top level menu names. After this top level menu is selected, **MenuItems** is used to return the list of items on a specific menu. **MainMenus** can be either a regular menu or a context submenu that is added to all the context menus. The **MenuItems** query is passed the current document name and any selected text in case you wish to vary the menu based on these arguments.

By default, the source control class inherits these queries from %Studio.Extension.Base, where they are defined as SQL queries against prebuilt tables. To load data into these tables, define an XData block called `Menu` in your source control class. When the source control class is compiled, this data is loaded and used automatically. Queries defined in the source control subclass can be changed or completely customized. When data is being returned from the **MenuItems** query, each menu name will generate a call to an **OnMenuItem** method in the source control class, where you may disable/enable this menu item. This allows simple modification of the menus without having to write a custom query.

10.6.4 New Stream Syntax

The class hierarchy for current stream classes has been changed so that %Stream.Object is the top class. This change does not alter stream runtime behavior.

In prior versions of Caché, it was necessary to define a stream property as `type = %Stream`, with a collection value of `binarystream` or `characterstream`. Now a stream property is defined by specifying the actual stream class as the type, and the collection keyword values of `binarystream` and `characterstream` are no longer used. A stream class is declared with a `classtype = stream`. This declaration is automatic for any class that extends a new class, %Stream.Object. For backward compatibility, the classes %Library.GlobalCharacterStream, %Library.GlobalBinaryStream, %Library.FileCharacterStream, and %Library.FileBinaryStream have been converted to use the new representation, and are to be used for all existing stream data.

For more detailed information, see the Streams chapter in *Using Caché Objects*.

10.6.5 New %SwizzleObject Class

A new class, %SwizzleObject, is now the primary (and only) superclass of both %Persistent and %SerialObject. The purpose of the new class is to define the swizzling interface and implement the parts of that interface that are common to both %Persistent and %SerialObject.

See the %Library.SwizzleObject class documentation for more detailed information.

10.6.6 Extended POPSPEC Syntax

The syntax of POPSPEC has been extended to allow an SQL table name and an SQL column name to be specified. When they are specified, the **Populate()** method constructs a dynamic query to return the distinct column values from the table. The requested number of values will then be randomly selected from the distinct column values and placed in a value set. The property will then be assigned values randomly from the resulting value set.

See The Caché Data Population Utility for more detailed information.

10.6.7 Performance Improvements for Relationships

The in-memory performance of relationships has been significantly improved by using additional in-memory indices to keep track of oref's and oid of items already in the relationship. Previously, when a new item was inserted into the relationship (either using the **Insert** method, or indirectly via the **Relate** method) it would scan the entire relationship to avoid inserting a duplicate item. By keeping an index of the oref's and oid's in the relationship, the cost of checking for duplication items is kept very low even for large numbers of items.

Partition memory used is lower, speed is significantly faster (94x in the second insert of 1000 items) and **%Save** time is faster. When measured with a small number of items in the relationship, there was no measurable slowdown in performance associated with the upkeep of the additional in-memory indices.

10.6.8 Enhanced VisM OCX

This release contains a new version of the Caché Direct control (VISM.OCX) that features enhancements such as security upgrades, support for multithreading, and improved error handling.

10.7 Language Improvements

New Caché 5.1 ObjectScript features and enhancements:

- Improved Runtime Error Reporting
- New **$FACTOR** Function
- New **$LISTNEXT**, **$LISTTOSTRING**, and **$LISTFROMSTRING** Functions
- New **$ROLES** and **$USERNAME** Special Variables
- New **$ZUTIL(62,1)** Function
- New **$ZUTIL(69)** Configuration Functions
- New **$ZUTIL(158)** Function
- New **$ZUTIL(186)** Function
- New **$ZUTIL(193)** Function
- New Error Trapping Syntax
- More Efficient Code Generation
- Pattern-Match “E” Adapted For Unicode
- Faster **MERGE** Command

New Language Bindings:

- New Perl Binding
- New Python Binding
- New ActiveX Bindings

10.7.1 Improved Runtime Error Reporting

Many runtime errors now report additional information. For instance, an "<UNDEFINED>" error will now report the name of the undefined variable.

Error information is stored in the system variable **$ZERROR**, which now returns more information than before. For example, when a routine attempts to use a variable that has not been defined, **$ZERROR** now includes the name of the undefined variable. Whereas in previous versions of Caché the value of **$ZERROR** might look like this:

```
<UNDEFINED>zMethodName^Pkg.Class.1
```

in version 5.1, it looks generically like this (adding " *someinfo"):

```
<ERRCODE>Tag^Routine+line *someinfo
```

A consequence of this change is that error handling routines that made assumptions about the format of the string in **$ZERROR** may now require redesign to work as before. For further information, see the Cache Conversion Guide, and the $ZERROR special variable in the *Caché ObjectScript Reference*.

10.7.2 New $FACTOR Function

$FACTOR is a new ObjectScript function for 5.1 that converts a numeric value to a bitstring. Its primary use is for the creation of bitslice indices. For further information, see the $FACTOR function in the *Caché ObjectScript Reference*.

10.7.3 New $LISTNEXT, $LISTTOSTRING, and $LISTFROMSTRING Functions

Caché 5.1 adds three new functions for processing list structures: **$ListNext**, **$ListToString** and **$ListFromString**.

$ListNext(`list`, `ptr`, `val`) allows extremely rapid traversing of a list structure (up to 400x faster than doing a loop with **$LIST**).

Before the first call to **$ListNext**, `ptr` should be initialized to 0. After each call, `ptr` will contain the position of the next element in `list` (0 if the end of the list was reached), and `val` will contain the value of the element at that position (undefined if there was no value at that position). **$ListNext** will return 1 if it found another list element, or 0 if it is at the end of the list.

$ListToString(`list`[,`delim`]) takes `list`, and returns the elements as a string separated by `delim` (default ",").

$ListFromString(`string`[,`delim`]) takes `string`, delimited by `delim` (default ","), and returns the pieces as a list.

For further information, see the $LISTNEXT, $LISTTOSTRING, or $LISTFROMSTRING function in the *Caché ObjectScript Reference*.

10.7.4 New $ROLES and $USERNAME Special Variables

At Caché 5.1, the **$ROLES** special variable lists the security roles currently assigned to the user. The **$USERNAME** special variable list the user name for the current process. For further information, see the $ROLES, $USERNAME special variables in the *Caché ObjectScript Reference*.

10.7.5 New $ZUTIL(62,1) Function

The **$ZUTIL(62,1)** function performs syntax checking on a line of Caché ObjectScript code. It returns the character position of the error and the text of an error message. For further information, see the $ZUTIL(62,1) function in the *Caché ObjectScript Reference*.

10.7.6 New $ZUTIL(69) System Configuration Functions

Caché 5.1 documents the following additional system-wide configuration functions: **$ZUTIL(69,19)**, **$ZUTIL(69,21)**, **$ZUTIL(69,31)**, **$ZUTIL(69,35)**, **$ZUTIL(69,37)**, **$ZUTIL(69,44)**, **$ZUTIL(69,49)**, and **$ZUTIL(69,60)**.

Caché 5.1 also supports the new **$ZUTIL(69,63)** and **$ZUTIL(68,63)** functions that control whether a lowercase "e" should be interpreted as an exponent symbol.

For further information, see the $ZUTIL(69) functions in the *Caché ObjectScript Reference*.

10.7.7 New $ZUTIL(158) Function

The **$ZUTIL(158)** function can be used to return the number of installed printers and the pathname of a specified printer. For further information, see the $ZUTIL(158) function in the *Caché ObjectScript Reference*.

10.7.8 New $ZUTIL(186) Function

The **$ZUTIL(186)** function can be used to specify the information displayed as part of the Terminal prompt. For further information, see the $ZUTIL(186) function in the *Caché ObjectScript Reference*.

10.7.9 New $ZUTIL(193) Function

The **$ZUTIL(193)** function inter-converts Coordinated Universal Time and local time values. For further information, see the $ZUTIL(193) function in the *Caché ObjectScript Reference*.

10.7.10 New Error Trapping Syntax

This version of Caché implements a special syntax that allows an error trap to pass control up the program stack to a previously established error trap. The syntax is **ZTRAP $ZERROR**. This command will pop entries off the program stack until a level is found with an error trap. Then that error trap will be executed with **$ZERROR** and **$ECODE** unchanged.

This command replaces the two commands `ZQUIT 1 GOTO @$ZTRAP`, which did not work in new-style procedures. This new command syntax can be used in both procedures and old-style subroutines. The old style of passing control up to a previous error trap will continue to work in old-style subroutines. If a **ZQUIT** command is issued in a procedure, it will now result in a `<COMMAND>` error.

The **ZQUIT** command is obsolete as of 5.1, and should not be used for new programming.

10.7.11 More Efficient Code Generation

The CacheBasic compiler now uses an improved algorithm that generates significantly smaller and faster code.

10.7.12 Pattern-Match "E" Adapted For Unicode

In prior version of Caché, the options used with the pattern-match operator(s) assumed 8–bit characters. This caused the "E" pattern (match every character) to fail when Unicode characters above **$CHAR(255)** were present in the string.

In Caché 5.1, the “E” pattern matches all characters.

10.7.13 Faster MERGE Command

The **MERGE** command is now much faster and more efficient when merging two local variables.

10.7.14 New Perl and Python Bindings

The Caché Perl and Python bindings provide a simple, direct way to manipulate Caché objects from within Perl or Python applications. They allow binding applications to establish a connection to a database on Caché, create and open objects in the database, manipulate object properties, save objects, run methods on objects, and run queries. All Caché datatypes are supported.

See Using Perl with Caché and Using Python with Caché for more detailed information.

10.7.15 Improved ActiveX Bindings

Caché 5.1 includes a new version of the Caché ActiveX binding, CacheActiveX.dll. Internally this new version uses the Caché C++ binding to get object-level access to a Caché server. Using this new binding provides the following benefits:

- access to the client/server security model available within Caché 5.1 (for example, the ability to use Kerberos authentication)
- better performance in some cases due to more sophisticated object caching.

While every attempt has been made to make this new DLL functionally compatible with the older CacheObject.dll it is not 100% binary compatible.

To preserve complete compatibility with existing applications, Caché installs two ActiveX bindings; the newer CacheActiveX.dll as well as the original CacheObject.dll. By default, existing applications will continue to use the original CacheObject.dll. If you wish to use the newer binding you have to modify your existing application to reference this new DLL and test that your application performs as expected.

10.8 SQL Improvements

New Caché 5.1 SQL features and enhancements:

New Features

- New SQL/XML Support Functions
- **SAVEPOINT**: New Transaction Processing Feature
- **CREATE TABLE**: New IDENTITY Keyword
- **DROP VIEW**: New CASCADE Keyword
- **INSERT**: New DEFAULT VALUES Clause
- New RowId Counter Validation Option
- New Query Optimizer Plan Verification

SQL Enhancements

- JDBC 3.0 Support

- **GRANT** and **REVOKE** Command Changes
- **CREATE USER** Command Changes
- Subquery Flattening
- Enhanced Locking Behavior for Foreign Key References
- READONLY Tables and Fields
- SQLCODE Changes
- Support for %%CLASSNAMEQ and %%TABLENAME
- CREATE BITMAP INDEX Support for Oracle Import Compatibility
- Extended Support for Milliseconds
- Date and Time Function Enhancements

10.8.1 New SQL/XML Support Functions

5.1 implements a collection of new built-in SQL functions for transforming "flat" relational queries into hierarchical XML documents. Application programs that need to generate HTML, or that need to export data in XML format, now have a general and portable interface that has wide industry support (ANSI/ISO SQL-2003 standard).

The following SQL/XML functions are available:

- **XmlElement** – Creates an XML element of the form: <tagName>body</tagName>, with optional attributes. **XmlElement** creates one tagged element that can contain multiple concatenated values.
- **XmlAttributes** – Specifies attributes for an XML element. **XmlAttributes** can only be used within an **XmlElement** function.
- **XmlConcat** – Concatenates two or more XML elements.
- **XmlAgg** – Aggregate function that concatenates the data values from a column.
- **XmlForest** – Creates a separate XML element for each item specified. **XmlForest** provides a convenient shorthand for specifying multiple elements nested within another element, where element instances that are NULL are omitted.

For more detailed information see XMLELEMENT, XMLAGG, XMLCONCAT and XMLFOREST in the *Caché SQL Reference*.

10.8.2 New SAVEPOINT Features

With version 5.1, Caché introduces multiple transaction levels (see Nested Rollbacks), which make it possible to roll back part of a transaction without losing all work completed to that point. Caché SQL now offers the following standard SQL commands that take advantage of this ability:

- SAVEPOINT <savepointName> — establishes a savepoint within a transaction.
- ROLLBACK TO SAVEPOINT — rolls back to the most recent savepoint.
- ROLLBACK TO SAVEPOINT <savepointName> — rolls back to the specified savepoint.
- COMMIT – commits only the current sub-transaction when $TLEVEL > 1.

For more detailed information see SAVEPOINT in the *Caché SQL Reference*.

10.8.3 CREATE TABLE: New IDENTITY Keyword

Caché SQL now supports the ability to define a column with a system-generated numeric value in a **CREATE TABLE** statement. An IDENTITY column is an exact non-negative integer column whose values are system-generated, and may not be assigned by the user in either **INSERT** or **UPDATE** statements. It may, however, be viewed using **SELECT ***. The syntax is:

```
CREATE TABLE <tablename> (
   [ other-table-elements , ]
   <columnname> [ <datatype> ] IDENTITY
   [ UNIQUE | NULL | NOT NULL |
      DEFAULT [(]<default-spec>[)] |
      [COLLATE] <sqlcollation> |
      %DESCRIPTION <literal>
   ]
[ , other-table-elements ]
)
```

An IDENTITY column is always data type INTEGER with unique non-null values. You can specify a datatype and constraints, but these are ignored by Caché.

This syntax is consistent with Microsoft SQL Server and Sybase syntax.

For more detailed information, see CREATE TABLE in the *Caché SQL Reference*.

10.8.4 DROP VIEW: New CASCADE Keyword

Caché SQL now supports the ability to cascade the deletion of a view to also delete any view that references that view. The new keywords are CASCADE and RESTRICT. The RESTRICT keyword is the default and is the same as prior **DROP VIEW** behavior.

For more detailed information, see DROP VIEW in the *Caché SQL Reference*.

10.8.5 INSERT: New DEFAULT VALUES Clause

Caché SQL now supports the ability to use default field values when inserting a row into a table. The syntax is:

```
INSERT INTO <tablename> DEFAULT VALUES
```

The statement will insert a single row into the table. Each field that has a default value will have the value assigned to the column. Fields without default values will be NULL for the row.

For more detailed information, see INSERT in the *Caché SQL Reference*.

10.8.6 New RowId Counter Validation Option

A new configuration option now makes it possible to validate new system-assigned ID values. The option is activated by setting `^%SYS("dbms","validate system-assigned id")` to 1. Although such validation is not normally necessary, it is possible that the ID could be invalid if the user has modified the value manually, or if objects are inserted into the table without using the object or SQL filer. Other system recovery errors could also allow this condition to exist (bad recovery of a journal file, disk failure, etc.).

When this option is enabled, the table compiler will generate a uniqueness check on insert for the Id value. If validation fails, `SQLCODE=-119`, will be returned to the caller and a message will be written to the console log. After writing a message to the Console.log file and before returning from the filer, the user-defined routine **^%ZOIDERROR** will be called. It is important to review the console log when this error is reported.

When this error is reported, it will be necessary to bring the ID counter back into sync with the data. Each failure will cause the system ID counter to be incremented, so it is possible that the problem will correct itself over time. At the point the

error is reported it is not necessarily true that the counter is wrong, since the data itself may be incorrect. It is the responsibility of the user to determine how the counter became invalid.

10.8.7 New Query Optimizer Plan Verification

Regression tests based on **TestSQLScript** now have an easy way to verify query plan stability. Defining the class parameter `SHOWPLAN=1` in **%UnitTest.TestSQLScript** will cause the query optimizer plan to be written to an output file.

10.8.8 JDBC 3.0 Support

Cache 5.1 supports JDK 1.4 and JDBC 3.0. All required features and most optional features are supported.

10.8.9 GRANT and REVOKE Command Changes

Due to the extensive improvements to Caché security at 5.1, the SQL **GRANT** and **REVOKE** commands no longer support the following syntactical forms:

- GRANT ACCESS ON *namespace*
- GRANT %THRESHOLD *number*
- The %GRANT_ANY_PRIVILEGE, %CREATE_USER, %ALTER_USER, %DROP_USER, %CREATE_ROLE, %GRANT_ANY_ROLE, and %DROP_ANY_ROLE privileges

The **GRANT** and **REVOKE** command support the following additional options:

- Granting a role to a role, creating a hierarchy of roles
- The EXECUTE object privilege
- The granting of object privileges to stored procedures, as well as tables and views
- The use of the asterisk (*) to grant EXECUTE object privileges to all stored procedures

For more detailed information, see GRANT and REVOKE in the *Caché SQL Reference*.

10.8.10 CREATE USER Command Changes

At 5.1, issuing a **CREATE USER** does not automatically assign any roles or privileges to the user, regardless of the privileges held by the creator. Privileges and roles must be assigned to a new user using the **GRANT** command.

For more detailed information, see CREATE USER in the *Caché SQL Reference*.

10.8.11 Subquery Flattening

In many cases the SQL engine will now attempt to “flatten” certain types of SQL queries. That is, a query will be internally converted into an equivalent form that does not contain a subquery. In many cases, it is easier for the SQL optimizer to recognize this equivalent form, and a better execution plan is generated.

10.8.12 Enhanced Locking Behavior for Foreign Key References

Locking behavior during table filing has been changed in the following ways:

- During SQL **DELETE**, for every foreign key reference a long-term shared lock will be acquired on the row in the referenced table. This row will be locked until the end of the transaction. This ensures that the referenced row is not changed before a potential rollback of the SQL **DELETE**

- During SQL **INSERT**, for every foreign key reference a long term shared lock will be acquired on the referenced row in the referenced table. This row will be locked until the end of the transaction. This ensures that the referenced row is not changed between the checking of the referential integrity and the end if the **INSERT**'s transaction.
- During SQL **UPDATE**, for every foreign key reference which has a field value being updated, a long-term shared lock will be acquired on the old referenced row in the referenced table. This row will be locked until the end of the transaction. This ensures that the referenced row is not changed before a potential rollback of the SQL **UPDATE**.
- During SQL **UPDATE**, for every foreign key reference that is being changed, a long term shared lock will be acquired on the new referenced row in the referenced table. This row will be locked until the end of the transaction. This ensures that the referenced row is not changed between the checking of the referential integrity and the end if the **UPDATE**'s transaction.

10.8.13 READONLY Tables and Fields

Prior to this version of Caché, trying to INSERT, UPDATE, or DELETE into a ReadOnly table would not result in an error until the statement was executed. In this version, an `SQLCODE=-115` error will be raised during compilation.

When a property is defined as ReadOnly, the field in the corresponding SQL table is also now defined as ReadOnly. READONLY fields may only be defined via an initialexpression or SQL Compute code; they may never be explicitly insert or updated via SQL statements. Any attempt to INSERT or UPDATE a value for the field (even a NULL value) will result in an `SQLCODE=-138` error ("`Cannot INSERT/UPDATE a value for a ReadOnly field`").

10.8.14 SQLCODE Changes

The following SQLCODE error codes have been added for 5.1:

- -129: This error is raised when you attempt to set a Caché Locale setting to an invalid value. See SET OPTION in the *Caché SQL Reference* for further details.

  ```
  SQLCODE = -129: Illegal value for SET OPTION locale property
  ```

- -138: This error is raised when you attempt to compile an INSERT or UPDATE that references a read-only field. See INSERT in the *Caché SQL Reference* for further details.

  ```
  SQLCODE = -138: Cannot INSERT/UPDATE a value for a ReadOnly field
  ```

- -142: This error is raised when the **CREATE VIEW** command contains a mismatch between the number of columns in the view definition and number of columns in the query. See CREATE VIEW in the *Caché SQL Reference* for further details.

  ```
  SQLCODE = -142: Cardinality mismatch between the View-Column-list and View Query's
  SELECT clause
  ```

- -308: This error is raised when you attempt to define more than one IDENTITY field for a table. See CREATE TABLE in the *Caché SQL Reference* for further details.

  ```
  SQLCODE = -308 Identity column already defined for this table
  ```

- -316: This error is raised when a Foreign key references a non-existent column.

  ```
  SQLCODE = -316 Foreign key references non-existent key/column collection
  ```

- -321: This error is raised when you attempt to drop a view when another view references that view. See DROP VIEW in the *Caché SQL Reference* for further details.

  ```
  SQLCODE = -321 Cannot DROP view - One or more views reference this view
  ```

- -356 and -357: These two errors may be raised by an attempt to use a user-defined SQL function.

```
SQLCODE = -356: SQL Function (function Stored Procedure) is not defined to return
a value

SQLCODE = -357: SQL Function (function Stored Procedure) is not defined as a function
procedure
```

- -375: This error is raised when you attempt to roll back to a savepoint that was either never established or has already been rolled back.

  ```
  SQLCODE = -375 Cannot ROLLBACK to unestablished savepoint
  ```

- -417: This error is raised when login fails. Usually this is due to username and password checking failure. It can also occur if the username is not privileged.

  ```
  SQLCODE = -417 Cache Security Error
  ```

- -431: This error is raised when you attempt to pass a literal as a stored procedure parameter when the underlying argument type is an object type.

  ```
  SQLCODE = -431 Stored procedure parameter type mismatch
  ```

- -459: This error is raised when you try to connect using Kerberos and security authentication fails. Possible reasons include: the Kerberos security executable cconnect.dll is missing or fails to load; your connection is rejected because of the Kerberos credentials you supplied.

  ```
  SQLCODE = -459 Kerberos authentication failure
  ```

The following obsolete SQLCODE values have been removed:

SQLCODE -340, -341, -342, -343, -344, -345, -346, -347

For a complete list of SQLCODE values, refer to Error Codes in the *Caché SQL Reference*.

10.8.15 Support for %%CLASSNAMEQ and %%TABLENAME

Caché SQL now supports {%%CLASSNAMEQ} and {%%TABLENAME} references in class definition SQL specific COS code in the following locations:

- SQL Computed field code
- SQL Trigger code
- %CacheSQLStorage conditional map condition expression.

{%%CLASSNAMEQ} (not case-sensitive) will translate to the quoted string for the name of the class which projected the SQL table definition.

{%%TABLENAME} (not case-sensitive) will translate to the quoted string for the qualified name of the table

For example, assume the following trigger in the class User.Person:

```
Trigger AfterInsert1 [ Event = INSERT, Order = 1, Time = AFTER ]
{
Set ^Audit("table",{%%TABLENAME},$j,"AFTER INSERT TRIGGER")=1
Set ^Audit("class",{%%CLASSNAMEQ},$j,"AFTER INSERT TRIGGER")=1
}
```

If User.Employee extends User.Person, the following SQL trigger code will be pulled as an AFTER INSERT trigger in the SQLUSER.EMPLOYEE table:

```
Set ^Audit("table","SQLUser.Employee",$j,"AFTER INSERT TRIGGER")=1
Set ^Audit("class","User.Employee",$j,"AFTER INSERT TRIGGER")=1
```

10.8.16 CREATE BITMAP INDEX Support for Oracle Import Compatibility

When loading an Oracle SQL script file through **$SYSTEM.SQL.DDLImport()** or **$SYSTEM.SQL.Oracle()**, Caché SQL now recognizes the **CREATE BITMAP INDEX** statement.

10.8.17 Extended Support for Milliseconds

Caché SQL now supports fractional seconds in all date/time functions. The **DATEADD**, **DATEDIFF**, **DATENAME**, and **DATEPART** functions now support a `datepart` of "ms" or "milliseconds". The ODBC Scalar functions **{fn TIMESTAMPADD()}** and **{fn TIMESTAMPDIFF()}** now support the `SQL_TSI_FRAC_SECOND` parameter.

See DATEPART in the *Caché SQL Reference* for more detailed information.

10.8.18 Date and Time Function Enhancements

- The SQL Scalar functions TO_DATE and TO_CHAR now accept %Library.TimeStamp logical values as input. In addition, the following format codes have been added for support of TimeStamp values:
  - `HH` – hour of day (1-12)
  - `HH12` – hour of day (1-12)
  - `HH24` – hour of day (0-23)
  - `MI` – minute (0-59)
  - `SS` – second (0-59)
  - `SSSSS` – seconds past midnight (0-86388)
  - `AM` – meridian indicator
  - `PM` – meridian indicator

- There is a new configuration setting for the default format value for the **TO_DATE()** function. The default format is still "`DD MON YYYY`", but it can be changed using the following commands:

  ```
  Do $SYSTEM.SQL.SetToDateDefaultFormat(<value>)
  ```

  or

  ```
  Do SetToDateDefaultFormat^%apiSQL(<value>)
  ```

  For example:

  ```
  Do $SYSTEM.SQL.SetToDateDefaultFormat("YYYY-MM-DD HH24:MI:SS")
  ```

  The current setting for the **TO_DATE()** default format can be displayed with:

  ```
  Do CurrentSettings^%apiSQL
  ```

  or

  ```
  Do $SYSTEM.SQL.CurrentSettings()
  ```

- The following **CAST** and **CONVERT** operations are now supported for **%FilemanDate** and **%FilemanTimestamp**:
  - `CAST (<%FilemanDate value> AS CHAR)`
  - `CAST (<%FilemanDate value> as DATE)`
  - `CAST (<%FilemanDate value> as TIMESTAMP)`

- CAST (<%FilemanDate value> as VARCHAR)
- {fn CONVERT(<%FilemanDate value>, SQL_DATE)}
- {fn CONVERT(<%FilemanDate value>, SQL_TIMESTAMP)}
- {fn CONVERT(<%FilemanDate value>, SQL_VARCHAR)}
- CAST (<%FilemanTimeStamp value> AS CHAR)
- CAST (<%FilemanTimeStamp value> as DATE)
- CAST (<%FilemanTimeStamp value> as TIME)
- CAST (<%FilemanTimeStamp value> as TIMESTAMP)
- CAST (<%FilemanTimeStamp value> as VARCHAR)
- {fn CONVERT(<%FilemanTimeStamp value>, SQL_DATE)}
- {fn CONVERT(<%FilemanTimeStamp value>, SQL_TIME)}
- {fn CONVERT(<%FilemanTimeStamp value>, SQL_TIMESTAMP)}
- {fn CONVERT(<%FilemanTimeStamp value>, SQL_VACHAR)}

10.9 Connectivity Improvements

New Caché 5.1 connectivity features and enhancements:

- New ECP Cluster Support
- New SNMP Support
- New LDAP Client
- New Mac OS X server support

10.9.1 New ECP Cluster Support

Enterprise Cache Protocol is now supported in shared disk cluster configurations with OpenVMS and Tru64 UNIX®.

Differences between ECP cluster and failover cluster:

- Faster failover
- Active shared disk(s).
- No network reconfiguration.
- Roll in and out cluster member for repair, upgrade, maintenance and etc.
- All cluster members are live.

Features

- ECP Cluster server will provide higher availability.
- Locks and transactions are preserved during failover.
- Only the cluster master serves the ECP clients.

- The cluster members could be used for other applications.

InterSystems strongly recommends the use of ECP for clustered systems. ECP represents a significant advance over predecessor networking approaches such as DCP. Customers currently using DCP for communications among members of a cluster will see improvements in performance, reliability, availability, and error recovery by converting to ECP.

10.9.2 New SNMP Support

To enable monitoring of Caché by a variety of systems management tools and frameworks, support for the Simple Network Management Protocol (SNMP) has been added. The %SYSTEM.MonitorTools.SNMP class allows for control of SNMP agents and functions. This class contains methods to start and stop the Caché SNMP agent, as well as the **CreateMIB()** method which generates a custom MIB file based on an application description in the Monitor Framework.

For details, see Using SNMP to Monitor Caché in the *Caché Monitoring Guide*.

10.9.3 New LDAP Client

Programmatic access to LDAP (Lightweight Directory Access Protocol) servers has been added. See the %Net.LDAP.Client.Session class documentation for details.

10.9.4 New Mac OS X server support

This version of Caché now installs and executes natively on Mac OS X 10.3. The installation kit is a standard ".dmg" distribution produced by PackageMaker.

Support has been added for Mac OS X as a server plus the following client components:

- ODBC
- JDBC
- Objects
- CSP Gateway for Apache

A native Objective-C binding is also available.