Caché 5.1 Release Notes

Version 5.1
15 June 2006
# Table of Contents

1 Caché 5.1 Release Notes .................................................................................................. 1
   1.1 Installation ................................................................................................................ 1
   1.2 First-Time Users ...................................................................................................... 1
   1.3 Upgrading ................................................................................................................ 2
   1.4 Major New Features ................................................................................................. 2

2 Caché Advanced Security ............................................................................................... 7
   2.1 Key Features ............................................................................................................. 7
      2.1.1 Security Advisor ............................................................................................. 8
      2.1.2 Low-level Security Interfaces ......................................................................... 9
      2.1.3 Common Criteria Security Certification ........................................................ 9
   2.2 Caché Advanced Security Concepts ........................................................................ 9
      2.2.1 Authentication: Establishing Identity ........................................................... 10
      2.2.2 Authorization: Controlling User Access ....................................................... 10
      2.2.3 Auditing: Knowing What Happened ............................................................ 10

3 System Management Portal .......................................................................................... 13

4 System Improvements ................................................................................................... 15
   4.1 New Features .......................................................................................................... 16
      4.1.1 Nested Rollbacks .......................................................................................... 16
      4.1.2 Namespace Mapping for Class Packages ..................................................... 16
      4.1.3 New Method $SYSTEM.Util.CleanDeadJobs() ........................................... 17
      4.1.4 New Class $SYSTEM.Monitor.Line ............................................................ 17
      4.1.5 New Method $System.Device.GetNullDevice() .......................................... 17
      4.1.6 New Optional Argument for $ZF(-2) ........................................................... 17
   4.2 Enhanced Features ................................................................................................. 18
      4.2.1 Option to Filter Records before Dejournaling on a Shadow ....................... 18
      4.2.2 Callin Enhancements .................................................................................... 18
      4.2.3 64K Routine Buffer Support ........................................................................ 19
      4.2.4 CVENDIAN Enhancements ......................................................................... 19

5 Object Improvements .................................................................................................... 21
   5.1 New Object Features .............................................................................................. 21
      5.1.1 New Option to Index on Computed Fields ................................................... 21
      5.1.2 New Object Synchronization ........................................................................ 22
5.1.3 New Studio Extension Classes and Source Control Hooks ........................................ 22
5.1.4 New Stream Syntax .................................................................................................. 24
5.1.5 New %SwizzleObject Class .................................................................................. 24
5.1.6 Extended POPSPEC Syntax .................................................................................. 24
5.2 Performance Improvements ...................................................................................... 25
  5.2.1 Performance Improvements for Relationships .................................................. 25
  5.2.2 Enhanced VisM OCX ......................................................................................... 25

6 Language Improvements .............................................................................................. 27
  6.1 ObjectScript Enhancements .................................................................................... 28
    6.1.1 Improved Runtime Error Reporting ............................................................... 28
    6.1.2 New $FACTOR Function .................................................................................. 28
    6.1.3 New $LISTNEXT, $LISTTOSTRING, and $LISTFROMSTRING Functions .............................................................................................................. 28
    6.1.4 New $ROLES and $USERNAME Special Variables ....................................... 29
    6.1.5 New $ZUTIL(62,1) Function .......................................................................... 29
    6.1.6 New $ZUTIL(69) System Configuration Functions ......................................... 29
    6.1.7 New $ZUTIL(158) Function .......................................................................... 30
    6.1.8 New $ZUTIL(186) Function .......................................................................... 30
    6.1.9 New $ZUTIL(193) Function .......................................................................... 30
    6.1.10 New Error Trapping Syntax ......................................................................... 30
    6.1.11 More Efficient Code Generation .................................................................. 30
    6.1.12 Pattern-Match “E” Adapted For Unicode ................................................... 31
    6.1.13 Faster MERGE Command ............................................................................ 31
  6.2 Language Bindings .................................................................................................. 31
    6.2.1 New Perl and Python Bindings ....................................................................... 31
    6.2.2 New ActiveX Bindings .................................................................................... 31

7 SQL Improvements ....................................................................................................... 33
  7.1 New features .......................................................................................................... 34
    7.1.1 New SQL/XML Support Functions ................................................................. 34
    7.1.2 New SAVEPOINT Features ............................................................................. 34
    7.1.3 CREATE TABLE: New IDENTITY Keyword .................................................. 35
    7.1.4 DROP VIEW: New CASCADE Keyword ......................................................... 35
    7.1.5 INSERT: New DEFAULT VALUES Clause ...................................................... 36
    7.1.6 New RowId Counter Validation Option ......................................................... 36
    7.1.7 New Query Optimizer Plan Verification ......................................................... 36
  7.2 SQL Enhancements ................................................................................................ 37
7.2.1 JDBC 3.0 Support .................................................................................................. 37
7.2.2 GRANT and REVOKE Command Changes ......................................................... 37
7.2.3 CREATE USER Command Changes ..................................................................... 37
7.2.4 Subquery Flattening .......................................................................................... 38
7.2.5 Enhanced Locking Behavior for Foreign Key References ................................. 38
7.2.6 READONLY Tables and Fields ............................................................................ 38
7.2.7 SQLCODE Changes .......................................................................................... 39
7.2.8 Support for %CLASSNAMEQ and %TABLENAME ............................................ 40
7.2.9 CREATE BITMAP INDEX Support for Oracle Import Compatibility ............... 41
7.2.10 Extended Support for Milliseconds .................................................................. 41
7.2.11 Date and Time Function Enhancements ......................................................... 41

8 Connectivity Improvements ....................................................................................... 45
8.1 ECP Enhancements ................................................................................................. 45
  8.1.1 New ECP Cluster Support ................................................................................ 45
8.2 New Protocol and OS Support ............................................................................... 46
  8.2.1 New SNMP Support ........................................................................................ 46
  8.2.2 New LDAP Client ............................................................................................ 46
  8.2.3 New Mac OS X server support ......................................................................... 46

9 Caché History .............................................................................................................. 49
9.1 Caché 2.0 .............................................................................................................. 49
9.2 Caché 2.1 .............................................................................................................. 49
9.3 Caché 3.0 .............................................................................................................. 50
9.4 Caché 3.1 .............................................................................................................. 50
9.5 Caché 3.2 .............................................................................................................. 50
9.6 Caché 4.0 .............................................................................................................. 51
9.7 Caché 4.1 .............................................................................................................. 51
9.8 Caché 5.0 .............................................................................................................. 51
9.9 Caché 5.1 .............................................................................................................. 52
Welcome and thank you for using Caché!

These Release Notes provide an overview of the new features in this release, information on getting started with Caché, and details on upgrading from previous versions.

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

1.1 Installation

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

1.2 First-Time Users

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.
1.3 Upgrading

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.

1.4 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.
Major New Features

- **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.
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.

### 2.1 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.

### 2.1.1 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.

2.1.2 Low-level Security Interfaces

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

^SECURITY

^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

^DATABASE provides low-level management capabilities related to Caché databases. An overview of ^DATABASE can be found in The CHUI-Based Management Routines.

2.1.3 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 is in the process of having Caché certified according to the Common Criteria standard. In fact, InterSystems is aiming to have Caché be the first product certified against the Common Criteria DBMS security profile.

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.

2.2 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.

2.2.1 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.

2.2.2 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.

2.2.3 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.
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.

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
4.1 New Features

4.1.1 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).

4.1.2 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.

### 4.1.3 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.

### 4.1.4 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.

### 4.1.5 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 VMS, //./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.

### 4.1.6 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.

### 4.2 Enhanced Features

#### 4.2.1 Option to Filter Records before Dejournaling on a Shadow

To filter journal records before they get dejournalled 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)
1 $e($qs(glo,0))="X" q 0 ;skip X* globals
d MSG^%UTIL(pid_,"_dir_","_glo_","_type_","_addr_","_time",1,0) ;log q 1
```

#### 4.2.2 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 *).

### 4.2.3 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).

### 4.2.4 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
System Improvements

- -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.
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

5.1 New Object Features

5.1.1 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.

5.1.2 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.

5.1.3 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.

5.1.4 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.

5.1.5 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.

5.1.6 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.
5.2 Performance Improvements

5.2.1 Performance Improvements for Relationships

The in-memory performance of relationships has been significantly improved by using additional in-memory indexes 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 indexes.

5.2.2 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.
6

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
6.1 ObjectScript Enhancements

A variety of new features and enhancements have been added to the Caché ObjectScript language.

6.1.1 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.

6.1.2 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.

6.1.3 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.

6.1.4 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.

6.1.5 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.

6.1.6 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.
6.1.7 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.

6.1.8 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.

6.1.9 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.

6.1.10 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.

6.1.11 More Efficient Code Generation

The CacheBasic compiler now uses an improved algorithm that generates significantly smaller and faster code.
6.1.12 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 $\text{CHAR}(255)$ were present in the string.

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

6.1.13 Faster MERGE Command

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

6.2 Language Bindings

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

6.2.1 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.

6.2.2 New 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.
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
7.1 New features

7.1.1 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*.

7.1.2 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:
New features

- 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.

7.1.3 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:

```sql
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.

7.1.4 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.
7.1.5 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.

7.1.6 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.

7.1.7 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.
7.2 SQL Enhancements

7.2.1 JDBC 3.0 Support

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

7.2.2 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.

7.2.3 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.
7.2.4 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.

7.2.5 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 of 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 of the UPDATE's transaction.

7.2.6 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").

7.2.7 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
SQL Improvements

-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.

7.2.8 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\} (case insensitive) will translate to the quoted string for the name of the class which projected the SQL table definition.

\{%%TABLENAME\} (case insensitive) 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

7.2.9 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.

7.2.10 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.

7.2.11 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)
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()
- CAST (<%FilemanTimeStam value> as VARCHAR)
- {fn CONVERT(<%FilemanTimeStam value>, SQL_DATE})
- {fn CONVERT(<%FilemanTimeStam value>, SQL_TIME})
- {fn CONVERT(<%FilemanTimeStam value>, SQL_TIMESTAMP})
- {fn CONVERT(<%FilemanTimeStam value>, SQL_VACHAR})
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

8.1 ECP Enhancements

8.1.1 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.

8.2 New Protocol and OS Support

8.2.1 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.

8.2.2 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.

8.2.3 New Mac OS X server support

This version of Caché now installs and executes natively on Macintosh 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.
Caché History

The development of Caché began in 1995 as successor product to InterSystems' family of ANSI-Standard M-based database products. The goal was, and remains, to create the world's highest performance database product coupled with rapid application development capabilities.

The brief history of major Caché releases is outlined below:

9.1 Caché 2.0

Released in early 1997. This was the first public release of the Caché Data Engine. Highlights include:

- Distributed Cache Protocol
- Windows support
- UNIX support

9.2 Caché 2.1

Released in September 1997. The first release to be called “Caché”. Highlights include:

- Caché Objects
- ActiveX binding
• ODBC
• OpenVMS support

9.3 Caché 3.0

Released in January 1998. This release focused on Unicode and Internationalization and was released primarily within Asian markets. Highlights include:

• Native UNICODE support
• Localization support

9.4 Caché 3.1

Released in January 1999. This was a significant release with major revisions to every area of the product. Highlights include:

• Unified Data Architecture
• Native object support
• Caché Explorer
• Caché Control Panel
• Caché Configuration Manager

9.5 Caché 3.2

Released in January 2000. The follow-up to the very successful v3.1 release included many improvements in all areas. Highlights include:

• JDBC
• Java Binding
• Linux support
• Caché SQL Manager
• Caché Studio

9.6 Caché 4.0

Released in December 2000. A major breakthrough for developing web-based database applications. Highlights include:

• Caché Server Pages
• Major syntax improvements for Caché ObjectScript
• Caché SQL Gateway
• Class Packages

9.7 Caché 4.1

Released in September 2001. This release was focused on radically improving the underlying scaling, performance, and capacity of Caché. Highlights include:

• New Caché database engine
• New Lock Manager

9.8 Caché 5.0

Released in December 2002. This release was focused on boosting developer productivity with Caché. It also offered greater scalability via ECP as well as dramatic performance enhancements in SQL (due to bitmap index technology). It included significant XML-based features. Highlights include:

• Integrated Caché Studio
• Online Documentation System
• ECP Distributed Database Support
The most recent version, released in November 2005. This version of Caché contains powerful new features that provide the most advanced security of any mainstream database. It now uses a browser-based interface, the System Management Portal, that allows systems to be managed from any platform. It also contains a host of other new features and major enhancements. Highlights include:

- Caché Advanced Security
- Browser-based System Management Utilities
- System Enhancements:
  - Enhanced shadowing with improved system performance and new transaction support
  - New SNMP support
  - ECP performance improvements
  - ECP shared disk cluster support for OpenVMS and Tru64 UNIX
  - New nested transaction and rollback support
  - Namespace mapping for class packages
- Object data synchronization – export object data and synchronize it with other databases
- New language bindings for Perl and Python
- Enhanced error reporting with greater clarity and detail