icell developers guide

# INTRODUCTION

This guide will be aimed at developers wanting to get a start in the structure of the iCell suite of software, it won’t give a detailed dive, but rather point the developer at where to find various parts of the iCell system, and how to set up to be able to build the software suite.

The developer should have extensive experience in Delphi 10 or later, experience in SQL Server and using TCP connections. There is a lot of advanced technology, especially in the iCell POS software itself.

# PREREQUISITES

To build the icell suite of software, the following prerequisites are required:

* Embarcadero Delphi 10.4.2 Enterprise Edition
* Microsoft SQL Server Client (and connection to SQL Server Database with iConnect Database)
* Component ACE ZipForge 6.93 or later (with Source code)
* TMS Software VCL UI Pack 10.6.3.0 or later
* TMS Software Cryptography Pack 4.3.2.0 or later
* TMS Software Async32 2.3.0.0 or later
* KLUUG OXML 2.10 or later source code
* Fastreports 6.9.14 or later
* ImageEn 10.2.0 or later with source code
* TortiseSVN 1.14.1 or later
* Workdrive Client for Windows (connected to Workdrive folder iCell Development)
* 7-Zip compression software
* Konopka Signature VCL Controls 6.3.0 or later (installed via Delphi Get It package manager)

The installs for these libraries (except Konopka) will be in the installs folder of the source tree, the licence information and username/passwords will be in the licences subdirectory of the installs folder.

# getting started

The source tree will already be structured as follows, under the root path “D:\Work Projects\Finchcorp Systems”, note that this path can be changed, but may require some adjustments of paths within Delphi and the project files.

Within this directory the following four main folders exist (or should be created if starting from scratch).

* Installs – contains all the library installations
* Delphilibs – is where the libraries will be installed (except Konopka)
* Icons – just stores various ICONS libraries (ICO, PNG, BMP etc)
* iCell – the main source tree

Within the iCell source tree there will be a ‘current’ folder with the current version of the iCell suite, and possibly other folders with different iCell source trees (e.g. 2.6.2365.4710). Each of these folders is treated in a similar manner. There is also a batch file (used for deployment) called ‘deploy.bat’. This file can be found in the installs folder, and needs to be edited to reflect the location of Workdrive on your Windows system.

Within each source tree, there will be a build and source folder. The build contains the output binaries, and sufficient configuration directories to be able to run the software from within the Delphi environment (that is – under the binaries folder will look similar to an actual installation of iCell). The source folder contains the actual source and will contain two main project groups “iCell Development” which is the main suite, and “iCell Support” for support utilities.

# INSTALLATION

Firstly ensure the directory structure is in place (as above), the do as follows:

1. Install TortiseSVN, then connect the following subversion points (see Matthew for usernames/passwords):  
   **iCell\Source** -> https://finchcorpsvn.cloudapp.net:18443/svn/Finchcorp/iCellSrc  
   **iCell\Build** -> https://finchcorpsvn.cloudapp.net:18443/svn/Finchcorp/iCellBuild  
   **DelphiLibs** -> https://finchcorpsvn.cloudapp.net:18443/svn/Finchcorp/DelphiLibs  
   **Installs** -> https://finchcorpsvn.cloudapp.net:18443/svn/Finchcorp/Installs
2. Install Workdrive for Windows ensure you have access to the iCell Development folder and at least the Binaries folder is synchronised to your local computer.
3. Copy the deploy.bat file from the install folder to iCell\source\deploy.bat
4. Workdrive will create a folder name that can’t be easily used within the batch file, so you will need to create a junction to another folder (instructions are in deploy.bat), then edit deploy.bat and set the DROPBOX variable to that junction name (a junction is just an alias for another directory).
5. Install 7-Zip standard installation
6. Install Delphi, select just Windows (32+64), examples and help to install.
7. Copy FinchcorpGray.vsf (Finchcorp Style for icjv) to the Delphi Styles folder (e.g. C:\Users\Public\Documents\Embarcadero\Studio\21.0\Styles)
8. Use Delphi Getit Package Manager to install the latest version of Konopka Controls.
9. For the following steps, ensure the final install directory is exactly as shown (the installers tend to add their own suffix to the path you enter, that needs to be removed), each path will be a subdirectory of the DelphiLibs folder.
10. Install TMS Async32 into the ‘Async’ subfolder
11. Install TMS Crytography Pack into the ‘Crypto’ subfolder, edit the tmscrypt.inc folder to enable Win64 for Delphi > 10.2.1.
12. Install TMS VCL UI Pack into the ‘TMS’ subfolder
13. Install ZIpforge into the ‘ZipForge’ subfolder (examples can go where ever you want)
14. Install Fastreports into the ‘FastReports’ subfolder (you may need to install this as Administrator)
15. Install ImageEn into the ‘ImageEn’ subfolder
16. Unzip Kluug XML into the ‘oxml’ subfolder, ensure that it isn’t extracted into a subfolder of that.
17. Using Delphi, add the units\_XXXXX folders under oxml to the 32 bit and 64 bit Windows libraries paths (e.g.):  
    Text, letter

    Description automatically generated
18. Test compile iCell(32 bit), icjv(64 bit) and iConnectDBA(64bit).
19. Make sure the .7z of the binaries ended up in the workdrive folder under binaries/current/Win32 and Win64
20. At this point, you will need to follow a normal install procedure (ask Finchcorp Support Staff) to set up an iConnect database for you to connect to, before you can run most of the software (iCell runs without database connection).
21. Review the other guides: iCell and iConnect Data Technical Summary, iCell Behaviours Guide, iCell Button Script Guide, iCell Macro Guide and iCell Report Template Guide.

# OVERVIEW OF THE ICELL SUITE

The core processes are:

* iConnect (middleware, includes the database and a number of software modules)
* iCell (till software, peer to peer, operates serverless)

The general flow of data is shown in this diagram:Diagram

Description automatically generated

The main three types of data involved:

* Transaction Data is collected from the tills by iConnect Service and accessed by iControl (and external views), these tables are all prefixed with ‘ic’ (e.g. icTransaction) and are sourced from the transaction XML flies collected by iConnect Service.
* Back Office Data is sent to iConnect by iControl and contains data the till requires for its operation (e.g. product lists, cashier lists etc). This data is delivered to iCell by iConnectConfigService. These tables are all prefixed with ‘db’ (with a few exceptions).
* Configuration Data is edited and configured via iConnect Management Studio and contains all the settings and non core data required by the till (e.g. till user interface definitions, payment gateway configurations, printer configurations etc). These tables are all prefixes with ‘cf’. Configuration data plays no part in reporting and will be outside the scope of this document.

Database Setup:

* SQL Server Express 2016 or higher/later.
* An SQL database schema ‘iConnect’ which holds the core data for iCell
* An SQL database schema ‘iControl’ which is a transition database used to move data from Firebird iControl to iConnect
* Transaction Data, Back Office Data and Configuration data tables
* A number of support views and stored procedure including the XML to table translation procedures.

# THE ICELL ECOSYSTEM

For the most part I’ll be focusing on the iCell ecosystem, as that is where most of the complexity lies within the suite. A map of all main applications and their main source files will be provided in the last section of this document.

## The INSTANCE

The instance is subsystem within iCell and iCaaS which takes care of all the business logic and is for the most part independent of a user interface. Within iCell the instance functions are performed by the Layout, within iCaaS multiple instances are handled via the Instance Manager.

This subsystem contains two general groups of objects and functions:

* The Core Functions encapsulate the business logic and processes
* The Core Objects encapsulate the support data and support functions.

The below diagram shows the overview of these two groups (it is an overview, showing the major systems and flows but not all).

Diagram

Description automatically generated

The command processor takes input and the command hub responds to specific commands (see the Command Button Script Guide), the specific commands will then call the transaction manager. The transaction manager coordinates all the functionality for acting on transactions, including support functions (including logging cashiers in, authenticating customers etc).

The primary functions of the Transaction manager involve adding and adjusting the items on the transaction (either manually or automatically via behaviours), adding payments to the transaction (via the many payment interfaces), the printing of the transaction and the sharing of the transaction with other tills and the server.

The core objects manages collection of configuration data, and back office data from the server and saving to the local file system. It also provides access to local scratch storage. Logging and file management are utility functions provided within the core objects group as well.

## CORE SOURCE FILES

|  |  |
| --- | --- |
| Source File | Comments |
| icInstanceManager.pas or icLayout.pas | iCell Core Instance Container |
| fCommandInterpreter.pas | Command Processor |
| fCommands.pas | Available Commands |
| fCommandHub.pas | Send Commands to Transaction Manager |
| fBusinessInterface.pas | Abstract Class for Tx Manager |
| fTransactionManager.pas | Transaction Manager |
| fXMLTransaction.pas | Transaction Classes |
| fTransactionAccount.pas | Transaction Account Classes |
| fBehaviourEngine.pas | Behaviour Processing Engine |
| fAdjustEngine.pas | Adjustment Processing Engine |
| fConditionEngine.pas | Condition Processing Engine |
| fBehaviourObject.pas | Root behaviour objects |
| fSetLine.pas | Behaviour class for ‘SET’ |
| fAdjustFormula.pas | Behaviour class for ‘ADJUST’ |
| fConditions.pas | Behaviour class for ‘CONDITION’ |
| fTodoLine.pas | Behaviour class for ‘TODO’ |
| fPrintManager.pas | Print/Report manager/processor |
| fScreenPrinter.pas | Printer class for print to screen |
| fESCPOSPrinter.pas | Printer class for ESCPOS printer |
| fFilePrinter.pas | Printer class for print to file |
| ifPaymentManager.pas | Payment manager/controller |
| fPaymentObjects.pas | Payment related classes |
| fPaymentInterface.pas | Root payment interface + cash |
| ifAristocrat.pas | Aristocrat Loyalty Interface |
| ifAsk.pas | Ask (cash with popup) Interface |
| ifBluize.pas | Bluize Charging System |
| ifCougar.pas | Cougar Loyalty System |
| ifEase.pas | Ease Charging System |
| ifEBet.pas | EBet Charging System |
| ifFinchcorp.pas | Finchcorp Accounts |
| ifIVend.pas | iFVend Charging System |
| ifMaxETag.pas | MaxETag Charging System |
| ifMews.pas | Mews Charging System |
| ifMicros.pas | Micros Charging System |
| ifNewBook.pas | Newbook Charging System |
| ifPassslot.pas | Passslot Coupon/Voucher System |
| ifPCEft.pas | PC EFTPOS EFTPOS System |
| ifRoomMaster.pas | RoomMaster Charging System |
| ifSpice.pas | Spice EFTPOS System |
| ifTitan.pas | Titan Charging System |
| ifTyro.pas | Tyro EFTPOS System |
| ifZipPay.pas | Zip Pay payment system |
| fHiveObjects.pas | Hive Classes |
| hiveClientServer.pas | Hive Broadcast (send and receive) |
| hiveTillServerContainer.pas | Hive Direct Incoming (container) |
| hiveTillServerMethods.pas | Hive Direct Incoming (functions) |
| hiveDirectTillClient.pas | Hive Direct to Till Methods |
| hiveDirectRawTillMethods.pas | Hive Direct to Till Raw calls |
| hiveDirectServerClient.pas | Hive Direct to Server Methods |
| hiveDirectRawServerMethods.pas | Hive Direct to Server Raw Calls |
| fCoreContent.pas |  |
| fConfigObjects.pas | Registry based config class |
| fLogger.pas | Local Logging system |
| fFileManager.pas | Local File/ZIP, Config Retrieval |
| fUpdaterEngine.pas | Controls receiving/applying updates |
| fConfigMgmtDefaults.pas | Default content for config |
| fConfigMgmt.pas | Root classes for XML,JSON,TXT Content |
| fXMLObject.pas | Root class for persistent XML Object |
| fXMLCashier.pas | Backoffice cashiers.xml |
| fXMLCustomer.pas | Backoffice customers.xml |
| fXMLProduct.pas | Backoffice catalog.xml |
| fXMLInstructions.pas | Backoffice instructions.xml |
| fErrorMessages.pas | iConnect messages.xml |
| fGatewaysObjects.pas | iConnect gateways.xml |
| fSettingsObjects.pas | iConnect settings.xml |
| fPrintingObjects.pas | iConnect printing.xml |
| fXMLMachine.pas | iConnect network.xml |
| fBehaviourInstances.pas | iConnect Behaviours[0-2].json |
| fTemplates.pas | iConnect templates.txt |
| fButtonObjects.pas | iConnect buttons.txt |
| fHardwareObjects.pas | iConnect hardware.xml |
| fScheduleObjects.pas | iConnect schedule.xml |
| fClientSideMacros.pas | iConnect clientmacros.ini |
| fGlobals.pas | Scratch storage.xml |
| fStockLevel.pas | Scratch stock.xml |
| fMoneyRecord.pas | Scratch money.xml |

# ICELL

## USER INTERFACE

The iCell user interface is designed by the end user, and loaded at run time by icell. The following is an overview of how the user interface works and how it interacts with the core functions above.

The configuration data for the user interface is retrieved from the configuration server (via the File Manager), this includes the layouts (in layout.zip), button scripts (buttons.txt) and other information. The layout manager then creates the on screen visuals live from this data. In response to either button presses, or hardware inputs (scanners, proximity sensors etc) commands or command scripts are sent to the command processor. In addition – any visual commands (refresh, message etc) are sent up to the GUI handler in the Layout manager.

The image below is an overview of the critical parts of iCell user interface.

Graphical user interface

Description automatically generated with medium confidence

## UI SOURCE FILES

|  |  |
| --- | --- |
| Source File | Comments |
| icMain.pas | Main iCell, initialisation, layout container and error message display. |
| icSplash.pas |  |
| fUserInterface.pas |  |
| icLayout.pas |  |
| fXMLLayout.pas | Configuration layout.zip |
| fXMLStyles.pas | Configuration theme.xml |
| icTheme.pas | Theme controller, manager |
| icTillObject.pas | Base Till Object and simple objects (e.g entry) |
| icTillButtons.pas | Buttons, Button Pages and Button Books |
| icTillReceipt.pas | On screen receipt |
| icContent.pas | Content objects (images, playlists, html etc) |
| icOSReceipt.pas | Main till version of receipt frame |
| fTxViewer.pas | Generic receipt frame |
| icFields.pas | Handles fields used in the receipt display |
| fFieldsObjects.pas | Field objects, including populating receipt display |
| fXMLFields.pas | Configuration - Fields.xml |
| icTillBookPropertyEditor.pas | Property editor for Button books and pages |
| icTillButtonPropertyEditor.pas | Property editor for Buttons |
| icTillLabelPropertyEditor.pas | Property editor for Label/Entry objects |
| icTillLayoutPropertyEditor.pas | Property editor for overall layout |
| icTillReceiptPropertyEditor.pas | Property editor for receipt object |
| icConfig.pas | Loads configuration from config and/or registry |
| icUIGlobals.pas | Globals used throughout iCell (including a core object) |
| icDrawerManager.pas | Manages cash drawers |
| icDisplayManager.pas | Manages external two line displays |
| icQuestionManager.pas | Manages popup questions |
| qCalculator.pas | Modal Popup question Form |
| qCustomerPicker.pas | Modal Popup question Form |
| qDualTx.pas | Modal Popup question Form |
| qExpiryEditor.pas | Modal Popup question Form |
| qGenericQuestion.pas | Modal Popup question Form |
| qGroupPicker.pas | Modal Popup question Form |
| qIMAGEPicker.pas | Modal Popup question Form |
| qInstructionPicker.pas | Modal Popup question Form |
| qKioskInstructionPicker.pas | Modal Popup question Form |
| qNumberEditor.pas | Modal Popup question Form |
| qProductDetail.pas | Modal Popup question Form |
| qProductPicker.pas | Modal Popup question Form |
| qRecovery.pas | Modal Popup question Form |
| qStockQuery.pas | Modal Popup question Form |
| qTablePicker.pas | Modal Popup question Form |
| qTextEditor.pas | Modal Popup question Form |
| qTinyPicker.pas | Modal Popup question Form |
| qTxIdEditor.pas | Modal Popup question Form |
| qTxPicker.pas | Modal Popup question Form |
| qURLPicker.pas | Modal Popup question Form |
| qVoucher.pas | Modal Popup question Form |
| icPersistence.pas | Manages local persistence information on layouts. |
| icRegistration.pas | Manages registration/licencing for iCell |
| icSecondMonitor.pas | Second monitor layout container |
| icSwitchHelp.pas | Popup help screen for iCell command line switches |
| icGlobalImages | Global images and icons used by iCell |
| fHardwarePorts.pas | Loaded (hardware.xml), for input devices and serial output. |
| fBehaviourEditorLocal.pas | Local behaviour debugger within iCell. |

# OTHER APPLICATIONS

## ICAAS – iCell as a service

ICaas is the rest/web API for iCell instances. It relies on the entire core instance (described above) and a connection to iConnect Config Server (to serve configuration information like product lists etc).

ICaaS itself consists of two parts:

* An instance manager which controls up to 10 distinct instances – with common paths across the instances.
* A redirection to the config server for catalog, pricing, stock levels etc.

The core source files for iCaaS (excluding the instance objects are):

* iCaaSMain.pas – iCaas when run in UI mode (/ui switch)
* ICaaSServiceMain.pas – iCaaS when run in service mode
* iCaaSWebModule.pas – the main web module
* icInstanceManager.pas – the instance (which encapsulates the core functions and core objects) and the instance manager which manages a list of such instances.

## iccs – iConnect config service

ICCS is the core service that reads the database information and allows tills to obtain configuration and backoffice data. It also handles start/stop signals from the tills and collects till info (screen resolution, till version etc) as part of the startup signal.

The core source files for iCCS are:

* icConfigServiceMain.pas – the service application
* iConnectWeb.pas – the web service which interprets the URLS and redirects requests to the database
* fMSSql.pas – all of the MSSQL routines to collect configuration and return as web friendly content.

## ICS – ICONNECT SERVICE

ICS is the core service that collects transactions from the tills and processes them into the database. Whilst processing transactions it can also send emails from the transactions. It also performs several scheduled activities, config backups, purges. ICS also serves as a hub for database activities required by the till (e.g. transaction recall, stock levels etc). ICS is very CPU and database intensive and is frequently split onto different machines at large venues (which ICS is responsible for tills/locations etc is configured via the dbTills, dbLocations etc tables.

The core source files for ICS are:

* icServiceMain.pas – main service application
* icServiceControl.pas – Threads for other timed/scheduled tasks run by ics.
* icTillThread.pas – Thread class for operating on each till (one thread per till is created)
* fMSSql.pas – Database functions used by ics (and other applications) including for inserting transactions
* hiveServerContainer.pas/hiveServerMethods.pas – Datasnap server for incoming till requests to Server Hive Direct
* hiveDirectTillClient.pas/hiveDirectRawTillMethods.pas – Datasnap client for outgoing hive direct to Tills.

## iCWS – iCONNECT WATCHDOG SERVICE

ICWS is a simple application that performs two functions:

* Monitors ICCS and ICS to ensure they are running and restarts them if required.
* Provides a web interface for controlling ICCS, ICS and ICWS remotely and for identifying the address of the config service via response to broadcast.

It is mostly self contained in iConnectWatchDogWeb.pas, with access to utility libraries.

## ICMS – ICONNECT MANAEGEMENT STUDIO

ICMS is a suite of applications run from the dashboard the allow configuration and monitoring of the tills under its control. Each of these apps (with the exception of icmsLayouts) is a fairly typical Delphi database application.

The core source files for ICMS are:

* icmsDashBoardMain.pas – Main dashboard application and login screen)
* icmsAdminMain.pas – Admin level functions
* icmsSettingsMain.pas – Hierarchical till configuration module
* icmsStylesMain.pas – Review/edit/add styles
* icmsControlMain.pas – Monitor, add and configure tills
* icmsEMailMain.pas – Monitor outgoing email queue
* icmsImagesMain.pas – Manage images and playlists
* icmsLayoutsMain.pas – Create, modify layouts (uses an embedded iCell UI)
* icmsMenusMain.pas – Create, Modify and Preview venue menus for display on large screens
* icmsPromosMain.pas – Create, Modify behaviours and promotions (instances)
* icmsScript, icmsScriptError.pas, icmsScriptListMain.pas, icmsScriptMain.pas, icmsScriptPromptMain.pas – Allows scripted installers and updaters to be created.
* icmsUtilities.pas - Contains various utility functions used by icmsScript and other icms modules
* icmSplash.pas – Splash screen for icms modules (except icmsScript)
* icConfigDM.pas – Data module containing Delphi objects reflecting the iConnect schema
* fMSSql.pas – Data base functions
* fSQLSchema.pas – Schema load, compare and modify functions

## ICKM – Icell kitchen monitor

When ICKM is installed on a PC – it acts like a printer, so iCell can print to it like any other printer. It interprets the prints it gets and turns them into on screen kitchen monitor dockets. These dockets can be processed by kitchen staff.

The main source files for icKM are:

* ickmMain.pas – Main ickm display, with bump buttons and docket grid
* ickmConfig.pas - ickm popup configuration option dialog.
* ickmConfirm.pas – Confirmation dialog
* ickmReport.pas – Report/Summary dialog

## ICJV – ICELL JOURNAL VIEWER

ICJV allows a user to search for transactions, then preview or save (to PDF) single transactions or transaction summaries.

The main source files for icJV are:

* icvMain.pas – main search form for icjv allowing a direct database search returning an in memory list of transactions.
* icvResults.pas – allows the user to browse and filter and preview the returned list of transactions.
* icvSplash.pas – Splash screen for icjv.

## REGISTRATION UTILITIES

Some key support utilities:

* finReg, finRegMain.pas – Office only registration code generation utility
* finUserReg, finUserRegMain.pas – End user registration of software utility.
* finSign , finSignMain.pas – Office only create signed update patches.

# CONCLUSION and programmer notes

The above is not a conclusive list of all modules, there are dozens of support units used throughout (e.g. fUtils which contains various utility functions). However it will guide you to where to look for the primary functions of each of the applications.

For the most part, if you are looking for a particular file – the following will guide to where the source file is:

* ./fLib - files starting with lower case 'f' globsally available units
* ./ifLib - files starting with lower case 'if' payment gateway implementations
* ./hlib - files starting with lower case 'h' hive broadcast and hive direct files
* ./qlib - files starting with lower case 'q' are popup questions
* ./iCell - main application projects and units

Also, some projects and some units are no longer used – but have not yet been formally deprecated.

Overall – the best approach is to start from the top, look for the function you are investigating (e.g. hive) and work from that source file down.

Additional notes:

* The software is frozen in a transition point, the transaction manager was going to make it into an instance used by iCell and by iCaas, but this project was not completed, so you will find some parts of iCell refer to the core objects, and others refer to an abstract business manage object which is passed around everywhere. The intention was to have access to an instance object which would be available everywhere, rather then the dual business interface/core.
* The command processor passes commands up to the GUI and down to the Transaction manager. Which ever ‘answers’ the command is the one that processes it.
* It is important that you read the other documents especially the **iCell and iConnect Data Technical Summary**

Christopher Burke

4th of October 2021