2014

NinjaTrader™ Integration Specification

CUSTOM APPLICATION INTEGRATION WITH NINJATRADER™

MR.DOE

CORALINC | Quebec, CANADA

Disclaimer: This document has been produced without any affiliation to NinjaTrader™.

USING CUSTOM APPLICATION WITH NINJATRADER (SPEC):

Project Goals:

- Creation of a standalone Application which communicates with NinjaTrader
- Application shall perform all of its functions independent of NinjaTrader. NinjaTrader shall be used only as a front-end to display information from Application (i.e. via Annotations) and Quote data delivery.
- Data from Application to NinjaTrader shall be able to be used alongside indigenous NinjaTrader indicators and Application shall be able to be used with NinjaTrader back testing.
- Application shall be able to send trading commands to NinjaTrader in Real-time and/or back testing.
- Application shall be easily integrated with other mainstream trade applications like MetaTrader etc.)

Project Definition: A custom standalone Application platform (AP) shall be used along with NinjaTrader. The AP shall be launched, when a custom NinjaTrader Script (Indicator) is applied on a NinjaTrader Chart. As soon as the Indicator is applied (on chart), the AP shall launch and a bi-directional communication channel shall be established between the AP and NinjaTrader, allowing parameter exchange. Finally, the application shall read .ntd file (NinjaTrader custom database file) for the use of OHLCV data at the AP (for instance a custom chart display besides NinjaTrader can be displayed simultaneously and automatically, if needed). From this point on, any complex market study can be done at the custom AP space and the results can be passed to NinjaTrader for display purposes via annotations and/or further manipulation for real time trading and historical back testing.

Solution Libraries: NTCL(NinjaTrader Custom Class Library), ANCL(Application-Ninja Class Library), CACL (Custom Application Class Library)

Library References:

- NTCL project references ANCL
- NTCL project references CACL (This is declared as a class library as well-(refer to Note1)
- Custom Application Class Library CACL references ANCL
- NTCL project references NinjaTrader. Core library (refer to Note2)

Description(detail): NTCL is the class library, where the NinjaTrader indicator class (Ninja Script) resides. At OnStartUp event handler it Instantiates NTCL class “ClsNinjaPresenter” by passing core (NinjaTrader. Core) parameters (InstrumentName, periodType, marketDataType) via abstract layer interface INinjaView. (These parameters are required to read the .ntd NinjaTrader database file). In return, ClsNinjaPresenter class creates an instance of “clsLibNinja” class of ANCL Class library (at constructor). It then passes the class object reference clsLibNinja to clsApPlatPresenterMain class of CACL Class Library via the CACL interface “IApPlat” (while creating an instance of clsApPlatPresenterMain at the same

time). Now, both class libraries ANCL and CACL are pointing at the same instance of "clsLibNinja" class. From this point on, the CACL can pass process data to NinjaTrader and read back data from NinjaTrader, as a piping is established between the Application Platform (AP) and NinjaTrader.

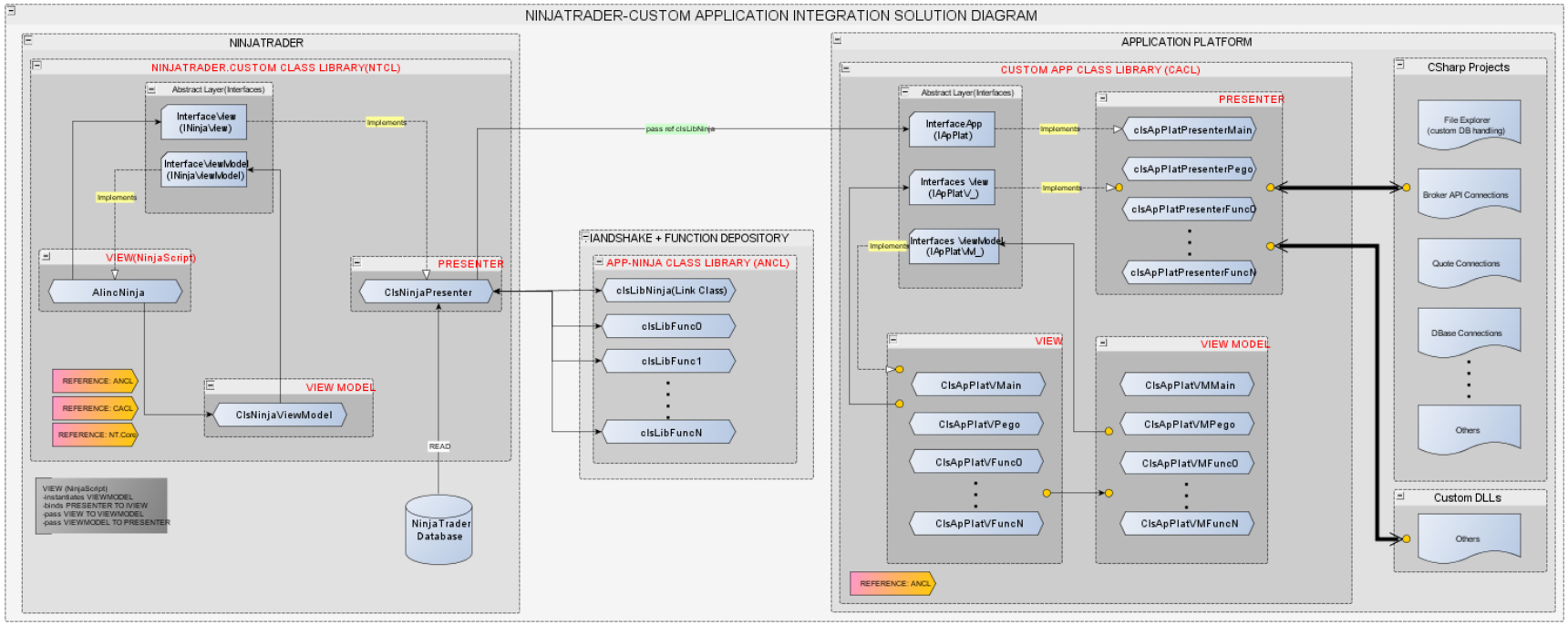
Reading of DB Files: NinjaTrader uses custom .ntd database files for its charts. These files can be read from the binary file, decoding hexadecimal sequences (refer to excellent work done by mrjoe, @gomi ,@dalebru and nicolas11 of BigMike forum. Thanks again. https://www.bigmiketrading.com/ninjatrader-programming/7396-ntd-file-specification.html). This function is accomplished at ClsNinjaPresenter.

Note1: In a classic MVVM (ModelView-ViewModel) architecture, View Model is usually a Windows Form Application. However, here the Application Platform (AP) output type shall be declared as Class Library, for the simple reason, that the NinjaTrader script only binds DLLs as reference. Therefore, the View Model above is a class library with forms (views) added.

Note2: NinjaTrader. Core DLL shall be referenced ONLY by the NTCL *project. Other projects in the solution shall NOT reference this DLL, as it causes conflicts and eventual crash of entire NinjaTrader platform (possible cause may be DLL synchronization issues).*

Note3: ANCL and CACL *Class libraries shall be built into directory "Documents\NinjaTrader 7\bin\Custom". Otherwise, the libraries do not link properly (reason unknown but possibly has to do with the internals of the NinjaTrader script engine design).*

Note4: All the class libraries in the solution shall be based on framework .NET version 3.5, if NinjaTrader 7 is used.


NINJATRADER-CUSTOM APPLICATION INTEGRATION SOLUTION DIAGRAM
NINJATRADER
NINJATRADER.CUSTOM CLASS LIBRARY(NTCL)
Abstract Layer(Interfaces)
Interface\View (INinja\View)
Interface\ViewModel (INinja\ViewModel)
Implements
VIEW(NinjaScript)
AIincNinja
PRESENTER
ClsNinjaPresenter
REFERENCE: ANCL
REFERENCE: CACL
REFERENCE: NT.Core
VIEW MODEL
ClsNinjaViewModel
READ
VIEW (NinjaScript)
-instantiates VIEWMODEL
-binds PRESENTER TO IVIEW
-pass VIEW TO VIEWMODEL
-pass VIEWMODEL TO PRESENTER
NinjaTrader Database
pass ref clsLibNinja
IANDSHAKE + FUNCTION DEPOSITORY
APP-NINJA CLASS LIBRARY (ANCL)
clsLibNinja(Link Class)
clsLibFunc0
clsLibFunc1
clsLibFuncN
APPLICATION PLATFORM
CUSTOM APP CLASS LIBRARY (CACL)
Abstract Layer(Interfaces)
InterfaceApp (IApPlat)
Interfaces \View (IApPlatV_)
Interfaces \ViewModel (IApPlat\VM_)
PRESENTER
clsApPlatPresenterMain
clsApPlatPresenterPego
clsApPlatPresenterFunc0
clsApPlatPresenterFuncN
VIEW
ClsApPlatVMain
ClsApPlatVPego
ClsApPlatVFunc0
ClsApPlatVFuncN
VIEW MODEL
ClsApPlatVMMain
ClsApPlatVMPego
ClsApPlatVMFunc0
ClsApPlatVMFuncN
REFERENCE: ANCL
CSharp Projects
File Explorer (custom DB handling)
Broker API Connections
Quote Connections
DBase Connections
Others
Custom DLLs
Others

regregregergrege