Northfield ACES

The Northfield Asset Coverage Expert System

The ACES Team

Northfield Information Services

Table of contents

1. Welcome to Northfield ACES

The Northfield Asset Coverage Expert System (ACES) makes it easy for your organization to understand its financial risk by using Northfield's Risk Models, Optimizer Software, and Data Services from a unified web services platform. This site provides login access, comprehensive developer documentation for webservice and application software developers, and architecture and technical information about the Northfield ACES Software Platform.

You can download ACES documentation as a PDF here. For more information about Northfield, Northfield's research, and sales enquiries, please visit the Northfield home page.

Single Sign On Service Access

ACES Model Desktop - for model data downloads.

ACES Developer Desktop - for optimizer and web service developers.

Section	Description
About Northfield ACES	An overview of the Northfield ACES webservice and software platform and its capabilities.
ACES Software Platform	Documentation about the Northfield ACES Software Platform that powers https://aces.northinfo.com. The Northfield ACES Software Platform is also available for enterprise customers who want onsite deployment, or who want to integrate Northfield components into their own application servers.
API Documentation	ACES Services API Documentation.

2. About Northfield ACES

The Northfield Asset Coverage Expert System (ACES) makes it easy for your organization to understand its financial risk by using Northfield's Risk Models, Software and Data Services from a unified web services platform.

The Key Features of the Northfield ACES platform are:

- The Model Service
- Analytics Services
- Data Services
- Programmer Tools
- Enterprise Software Deployment

2.1 The Model Service

The Northfield ACES Model Service allows you to access Northfield Risk Model data while using an expert system to automatically solving complex asset data matching problems such as:

- Multiple identifier type lookups (Eg. CUSIP, SEDOL, TICKER).
- Identifier correction (Eg. Check digit provided/not provided).
- Identifier changes from corporate actions.
- Proxy generation for missing records .
- Custom record creation for the Northfield EE Model.

2.2 Optimizer Service

The Northfield ACES Online Optimizer Service lets you:

- Optimize your portfolios using the Northfield Optimizer.
- Perform risk and performance calculations using our Risk Analytics Toolkit.

2.3 Data Access

The Northfield ACES Data Access Service allow you to access Northfield and partner data:

- Download Northfield’s risk model data files.
- Access portfolio data held at custodians.
- Access reports created by us for you by using our RAMP protocol that uses our ACES platform. These reports are held by our secure data provider who is qualified and certified for this purpose.

2.4 Programmer Tools

All Northfield ACES Services are available for use in your own applications. You can:

- Use the Northfield ACES Java API to securely access the Northfield ACES platform from within your own java programs.
- Integrate any application using JSON based web services.

All components use the same documented object model which has comprehensive API documentation. The full javadoc for the java API is available here.

2.5 Enterprise Software Deployment

Northfield ACES is a deployment of the Northfield ACES Software Platform; a comprehensive application server developed by Northfield to integrates all our software and data services into an enterprise ready software stack. The Northfield ACES Software Platform is also available for onsite deployment for organizations that wish to host and manage their own environments.

3. Python/Webservice/Java Programmers

The Northfield ACES Software Platform is accessible to Python, Webservice and Java programmers.

3.1 NISACES Java API

The Java components and webservice model are documented in the "ACES Software Platform" section of this website. The full javadoc for the java API is available here.

3.2 NISACES Python API

NISACES provides a Python library that allows you to access Northfield ACES from your own Python programs. The NISACES library automatically manages secure connections to the Northfield servers and exports three modules that you can use in your own applications:

- nisopt - constructs input JSON for the NISOPT optimizer webservice API and provides helper functions to parse output JSON into useful reports.
- nisaces - constructs input JSON for the NISMDL model data webservice API. It makes model data available for your own analytics, and also manages automatic integrattion with the NISOPT optimizer webservice and the NISRAT Risk Analytics Toolkit webservices.
- nishelperfunctions - provides useful functions to interact with projects and files constructed by the Northfield Windows desktop optimizer application.

3.2.1 NISACES Python Installation

The NISACES Python library is installed using "pip" so that all dependencies and the operating environment are setup automatically for you. We recommend installing NISACES in a virtual Python environment in the following manner.

```
python3 -m venv .venv/
source .venv/bin/activate
python3 -m pip install --upgrade pip
python3 -m pip install https://saas.northinfo.com/wui/nisaces-5.0.0.tar.gz
python3 <your program>.py
deactivate
```

Once installed the "nisaces" library will be on your python path and available to be imported into your programs.

3.2.2 NISACES Python Usage

The following example demonstrates how you would use NISACES in your own Python program.

```
from nisaces import nisaces
from nisaces import nisopt
from nisaces import nishelperfunctions

# Establish a connection
auth_code = "<your CCS connection token>"
optimizer = nisopt.Northfield_Optimizer_API()
optimizer.set_auth_key(auth_code)

# Load data from a desktop project
portfolio_data = nishelperfunctions.read_csv_into_list_of_lists("cash.csv")
benchmark_data = nishelperfunctions.read_csv_into_list_of_lists("stock.hld")
buylist_data = nishelperfunctions.read_csv_into_list_of_lists("buylist.hld")
alpha_data = nishelperfunctions.read_csv_into_list_of_lists("alpha.alp")
attributes_data = nishelperfunctions.read_attributes_from_file("attributes.csv")
asset_universe = nishelperfunctions.get_asset_universe(portfolio_data,benchmark_data,buylist_data)

# Load the risk model data from the ACES NISMDL web service
risk_model_data = nisaces.run_ACES(auth_code,"nis-mdl-02m-usd","20240930",asset_universe)
mdl_aces_data = risk_model_data["data"]["mdl-factors"]
cor_aces_data = risk_model_data["data"]["mdl-correlations"]
exp_aces_data = risk_model_data["data"]["mdl-exposures"]

# Setup the optimiser with file and NISMDL data.
```

```
opt_request = nisopt.Optimizer_Request()
opt_request.add_risk_model_data(mdl_aces_data,cor_aces_data,exp_aces_data,headers=True)
opt_request.add_holdings_data(portfolio_data,benchmark_data)
opt_request.add_security_selection_data(buy_list=buylist_data,alphas=alpha_data)
opt_request.add_security_constraints_data(security_maximum_default_weight=5,position_threshold=.05)
opt_request.add_portfolio_constraints(maximum_number_of_assets=50)
opt_request.add_attributes_data(attributes_data)
optimizer.set_input_json(opt_request.to_json())
optimizer.run_optimization()

# Save data for local use.
optimizer.write_risk_decomposition_report("report_riskDecompOptimal.csv",True)
optimizer.write_risk_decomposition_report("report_riskDecompInitial.csv",False)
```

3.3 Access and Support

The Northfield team provides personal support to Application programmers. We will help you establish access to the ACES Platform, develop a business scenario with you and your stakeholders, and provide you with a working model of your scenario in your language of choice that you can then use in your own programs.

For more information please contact our programmer support team.

4. ACES Software Platform

4.1 The Northfield ACES Software Platform

4.1.1 Overview

The Northfield ACES Software Platform is a suite of software tools developed by Northfield so it can provide a “Software As A Service” solution to clients that want to access Northfield data and analytics services via the cloud. The Northfield ACES Software Platform is the software that powers the services accessible from https://aces.northinfo.com.

Northfield has a diverse range of clients with diverse technology requirements. Some clients want full onsite software based solutions, some want a fully hosted cloud based solution, and some want to mix and match both cloud and onsite services. For example, a client may want to use a cloud query for risk model data, but run their portfolios with the Northfield Optimizer within a private network. Another client may have their own end user software and simply want to make web service calls. To achieve these requirements the Northfield ACES Software Platform uses a unique peer to peer architecture that allows clients to securely mix and match the services they use, where they use them, and where the data processing is performed.

The purpose of this document is to provide software Architects and Programmers with the information they need to understand and use the Northfield ACES Software Platform software.

4.1.2 Architecture

The Northfield ACES Software Platform is implemented as a suite of java software components where each provides a public API via a java interface that allows programmers to configure and assemble the components to meet the specific needs of their applications. The components follow the services layer pattern and are 1 referred to as “Services”. The Services provided by the Northfield ACES Software Platform are shown in Table 1.

Table 1: Services

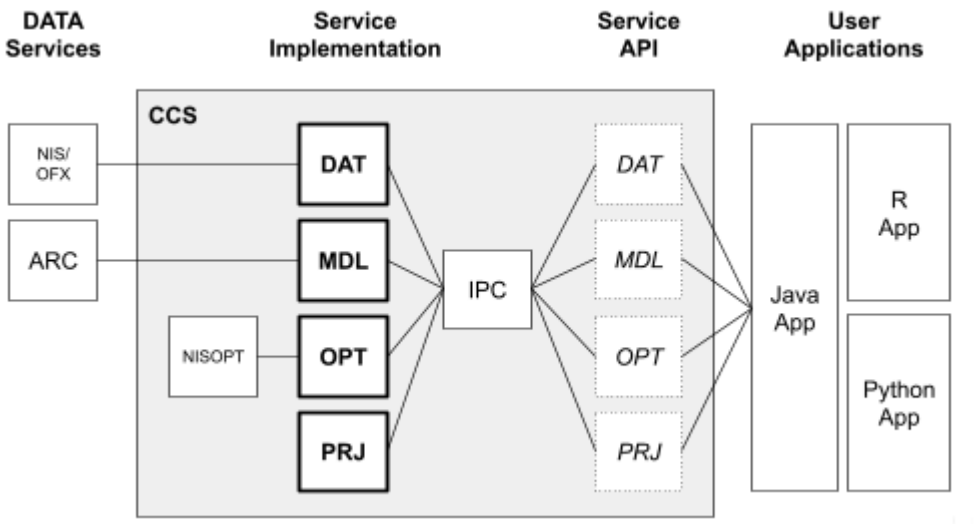
Service	Code	Description
Archive Service	ARC	Creates and distributes Northfield Risk Model data as archive files suitable for on-demand download and use by the SDK model component.
Connection Control Service	CCS	The application management system that configures the business components for use in user applications.
Data Service	DAT	Access to Northfield and 3rd party data sources for benchmark and portfolio data.
Model Service	MDL	Provides access to risk model data and associated services such as security identifier matching and custom exposure record calculations.
Optimizer Service	OPT	Enables the creation of optimizer projects, the invocation of the optimizer service, and the management of optimizer results and report data. The optimizer service integrates via the java native interface (JNI) to the Northfield Optimizer engine that is written in C++.
Project Service	PRJ	A framework for integrating configuration and process execution and data across the data, model & optimizer services. This allows for all server side processing so as to limit data shipping over the network.

The Northfield ACES Software Platform implements the IOC container pattern where a containing application is responsible for the composition and configuration of the Services at runtime. This feature is managed by the Connection Control Service (CCS) and allows Northfield ACES Software Platform applications to be configured according to a user selected mode.

CCS also contains a configurable inter process communication (IPC) mechanism1 which allows Services to be implemented locally, in a client/server configuration, or in a peer-to-peer model where the user chooses where a process is run based on their cost, performance and security requirements. External data services are managed by TLS connections initiated by the CCS container.

Figure 1 illustrates how a User Application integrates to the Service API that communicates through the IPC mechanism to the Service Implementation objects, which also have access to external data services. R & Python User Applications can gain access to the CCS Service API via bridging software.

Figure 1: Architecture



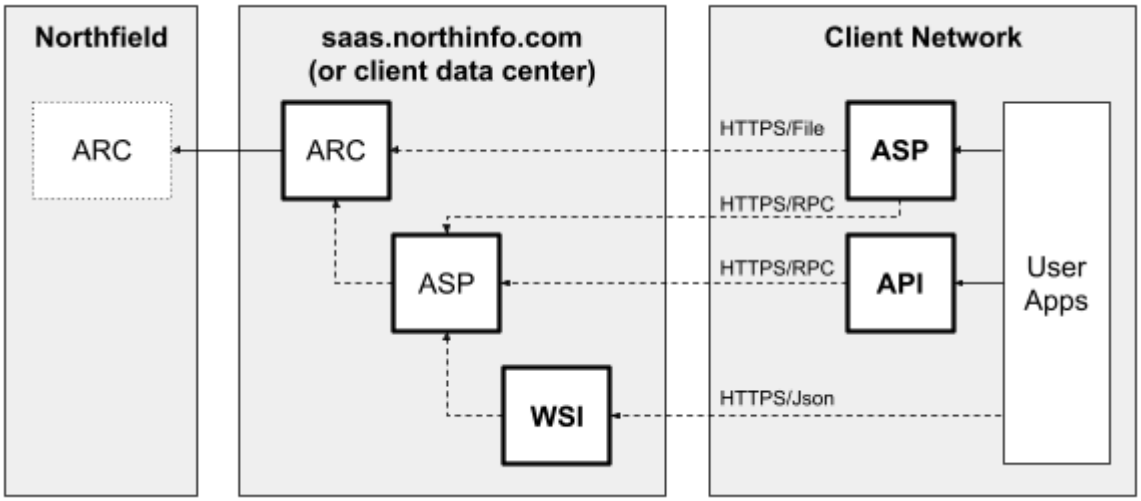
4.1.3 Deployment

The Northfield ACES Software Platform architecture enables the one code base to be packaged into different configurations that can be mixed and matched to allow different deployment models. The key deployable packages are:

- ARC - The risk model archive server that synchronizes with the Northfield model database and serves model files in ARC file format via HTTP/HTTPS.
- ASP - The Analytics Services Platform server that handles authorised Service API calls over a web RPC mechanism.
- API - The API jar file that can only talk to an ASP server.
- WSI - The Web Programmer Interface server which provides a JSON web services API translation for the Northfield ACES Software Platform Services.

Figure 2 illustrates how clients can configure the Northfield ACES Software Platform in their networks. User Applications have a choice of integrating via the Java SDK with all local processing or a combination of both, the Java API talking to saas.northinfo.com for purely remote processing, and the Web Services Interface (WSI) for access to the Services from any application type using JSON web service calls over HTTPS.
Any or all of these can be configured in combination of onsite or from Northfield's cloud servers at saas.northinfo.com

Figure 2: Deployment



4.1.4 System Requirements

The Northfield ACES Software Platform requires the Java Development Kit (JDK) and Java Runtime Environment (JRE) to be installed on the host computer and available on the system path of the user account running the software. The Northfield ACES Software Platform is available in two packages:

- Northfield ACES Software Platform Software (NIS-ASP) - contains all binary resources and demonstration code suitable for developing applications that perform local and/or remote analytics processes.
- Northfield ACES Software Platform Application Programmer Interface (NIS-API) - contains a single jar file and demonstration code suitable for developing applications that perform analytics using remote processing.

Both packages are made available to licensed users for download from saas.northinfo.com. Licensed users will also be provided with a “ccs.auth” code (authority code) which is used by the services at https://aces.northinfo.com for RPC call authentication and authorization.

4.2 Archive Service (ARC)

4.2.1 ARC Overview

The Model Archive Service (ARC) provides data to instances of the Northfield ACES Software Platform configured to provide a Model Service. For example, the Northfield ACES Software Platform Server running on https://aces.northinfo.com/ implements the Model Service API, and it pulls its model data from the ARC server running in the same network. The Model Service is configured to request the appropriate archive file from the Archive Service when a model query is requested that the Model Service has not performed before. The file is downloaded, opened, indexed, and cached for the first and any subsequent calls.

The Archive Service provides the master set of files for a model on a date in a single archive. It will contain whatever files the model query requires on the requested date including:

- factor files
- correlation files
- exposure files (Ticker, Cusip, Sedol, Munis, Funds)
- ID cross indexes.
- Factor adjustment data
- exposure risk adjustment data
- A reference to the model date that the daily model adjustes
- Daily asset ID change data

The ARC Service allows User access to approved Archive Data Sets and lets the Model Service control User access to approved model configurations.

4.2.2 ARC Operation

The Archive Service is available from the ARC tab on the SaaS web application, and archives can also be downloaded using the Web Service Interface.

4.2.3 ARC File Formats

There are two ARC file formats:

- EOM files contain the factor, correlation and exposure sets as at the end of the month.
- EOD files contain the factor and exposure adjustment files, and a “mth” file which contains the end of month date to which the EOD files are related.

Some archives will contain additional information. For example, some sets include multiple exposure files indexed with different identifier types (Eg. CUSIP, Ticker, SEDOL), different asset types (Bonds, Funds etc), and additional data such as cross index files and return files.

4.3 Connection Service (CCS)

4.3.1 CCS Overview

The Connection Service is the Northfield ACES Software Platform assembly and configuration service. Its role is to provide a simple interface over the complex requirements of creating and instantiating both client and server configurations of the other Northfield ACES Software Platform services. The ConnectionService: * Provides a unified way to assemble Northfield ACES Software Platform components and manage their life-cycle requirements. * Enables flexible configuration and licensing for different customer arrangements. * Supports the modification and tuning of parameters within a client project.

4.3.2 CCS Operation

User programs instantiate the NisConnect object and access any other Northfield ACES Software Platform services via the ConnectionService instance. This is shown in the following code from the samples.

```
Properties info = new Properties();
info.setProperty(NisConnect.CCS_BASE, base);
info.setProperty(NisConnect.CCS_MODE, mode);
ConnectionService ccs = new NisConnect(info);
try {
    ccs.open();
    ... use connection ...
} finally {
    ccs.close();
}
```

4.3.3 CCS Configuration

The NisConnect constructor accepts a standard java Properties object that contains connection setting information:

- ccs.mode - NisConnect manages user program construction and assembly based on preconfigured modes. The mode is set by using one of the modes defined in nis.app.asp.NisConnect. These are explained in Table 4b.
- ccs.base - NisConnect expects a pre configured connection service directory that contains any binary and connection assembly resources. The base directory in the Northfield ACES Software Platform archive provides a standard CCS_BASE for use by all client installations.
- ccs.logs - To enable logs in the connection service in any mode set this property as “true”.
- ccs.auth - Modes requiring online access require this property to be set with the ccs.auth data provided to you by Northfield support.

As discussed in the Architecture section, the Northfield ACES Software Platform architecture enables the one code base to be packaged into different configurations that can be mixed and matched to allow different deployment models. The standard package comes pre-configured with 4 standard connection modes and these are described in Table 4. Note that custom configurations can be created to meet clients specific needs in terms of balancing onsite and offsite processing with data access and security requirements.

NisConnect CCS_MODE Options Table

Mode	Download	Features	Notes
CCS_M01	NIS-SDK	OPT API ONLY configuration with single threaded calls to a single NISOPT JNI instance running in the same JVM. Requires the SDK to be installed and base/nis-api.jar on the program classpath. Designed to provide direct access to the optimizer in a single threaded, single user desktop environment.	
CCS_M02	NIS-SDK	OPT API ONLY configuration that calls to an ASP server process that is started in the background when the ConnectionService is created. The background ASP server process runs the NISOPT JNI library in a different address space to the client. Requires the SDK to be installed and base/nis-api.jar on the program classpath. This was designed for use in web server environments where a JNI crash might bring down the web server JVM.	
CCS_M03	NIS-API	API Mode - Pure Java library that makes all calls to a hosted Northfield ACES Software Platform server (Eg. aces.northinfo.com). Only requires the JAR file on the program classpath and server credentials.	
CCS_M04	NIS-SDK	SDK Mode - Binary installation with optimizer libraries that can mix MDL & ARC data access to a hosted Northfield ACES Software Platform server with the optimizer running in the same JVM as the user program. Requires the SDK to be installed and base/nis-api.jar on the program classpath. Requires server credentials	

Access Control and Product Codes

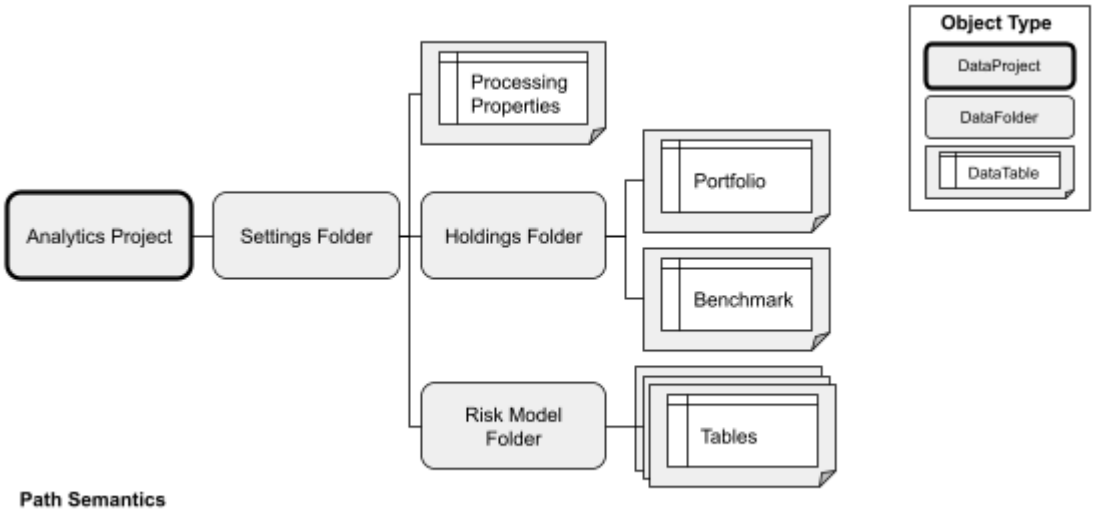
Northfield ACES Software Platform uses an attribute based security model that uses product codes combined with access control list rules to determine what Services and sub categories of services a User may access in an online environment. The use of these codes can be seen in the SDK samples, and the full list of product codes are available from the CCS page of the ACES Developer Desktop web application.

4.4 Data Service (DAT)

4.4.1 DAT Overview

The Data Service allows a User Program to make, save, and load DataProject objects. The DataProject model can be used to create a folder structure of tables that can be accessed by path semantics in the same manner as a file system. This is illustrated in Figure 3.

Optimizer Project DataService Diagram



The benefits of the DataService are:

- A DataProject can be serialized and sent to another node for processing where it can be deserialized and processed. The process can add its own data and pass it on to another node, or return it to the original sender.
- The DataProject is able to be serialized into JSON form and is used to encode and decode data in the Web Services Interface.
- External data sources can be added to the DataService to provide access to table based data in User projects.
- CSV Files can be accessed and loaded by the User program into the DataService.

4.4.2 DAT Operation

User Programs will normally use the DataService for accessing local and external data sources. External Data Sources are secured and identified by Data Project Codes. An example of data codes are shown in Table 5.

Sample Data Codes and Information Table

Code	Arguments	Source	DataProject Contents
nis-dat-001	None	Northfield Sample Data	Sample “benchmark” and “portfolio” data from the Northfield ACES Software Platform server.
nis-dat-218	OFX Portfolio ID	Open Finance	Asset list and asset information from an OFX positions report.

A full list of data codes and the data they provide is available on request. Access to data codes also needs to be granted and associated with your ccs.auth code provided to you by Northfield.

The Data API is best understood by reviewing the SDK demo files and the JavaDoc.

4.4.3 DAT Configuration

The DataService can be configured to read local CSV files. You can set the following properties in the ccs.properties file:

- csv.file.path: absolute path to root directory
- csv.root.name: name

This instructs the DataService to read the csv.file.path directory and all sub directories as DataFolder objects with the directory name as the folder name, and all CSV files as DataTable objects with the table name being the file name. The path root name is set by default to “local”, but can be any string value.

4.5 Model Service (MDL)

4.5.1 MDL Overview

The Model Service enables user programs to source and blend Northfield risk model data. This data can be used in the optimiser or in risk reporting and performance tools. The key features of the Model Service are:

- Online access to monthly and daily model data in compressed files that are organised and cross linked for easy retrieval and use in servers environments.
- Server based indexing, multiple data set searches, and automatic identifier cross referencing.
- Creating data sets for specific models and sets of securities.
- Blending risk models across different horizons. For example, blending annual and bi weekly horizons using monthly and daily risk model data.
- Custom exposure record proxy generation.
- Information about how an exposure record was matched.
- An online status report for model information and date ranges.

4.5.2 MDL Operation

Model Product Codes Risk Models are secured and identified by Model Product Codes. An example of the model codes for the Australian model are shown in the following table.

Australian Model Codes & Information

ARC Code	Model Code	Name	Currency	Update Period
010	nis-mdl-011	Australia Model - Monthly	AUD	EOM
010	nis-mdl-012	Australia Model - Bi Weekly	AUD	EOD

Access to model codes also needs to be granted and associated with your ccs.auth code provided to you by Northfield.

4.5.3 MDL Configuration

There are no local configuration options for the model service.

4.6 Optimizer Service (OPT)

4.6.1 Overview

The Optimizer Service provides a way for java programs to access the same software engine used in the Northfield Optimizer Desktop software. It is designed to provide an object model that follows the same tree structure as that used by the desktop software. This enables analysts and programmers to share a common model for defining and implementing analytics projects. Technically, the OptimizerService provides a java API on top of a system that performs the required mappings and transfer to and from the underlying C++ engine via the java native interface (JNI). The C++ engine is contained in a system library that is included in the CCS_BASE area.

4.6.2 OPT Operation

Scenario Development

The Northfield Optimizer is designed as a multi purpose tool that supports user scenarios starting with simple risk analyses, portfolio optimizations based on user provided parameters, through to complex multiple optimizations that can estimate parameters. As a multi purpose tool with complex settings that does not constrain the ability of the user to explore their data, there is a high risk of user error in the configuration of inputs and interpretation of outputs. In order to minimize and manage errors, we recommend your user interface:

- Implements clearly defined scenarios that are prototyped in the desktop tool.
- Ensures datasets are present, complete and correct for the User scenario
- Ensures input values and parameters are within the ranges you have prototyped
- Only presents the User with the specific join/run function required for your scenario
- Produces output reports and tables that are relevant to the scenario.

It is beyond the scope of this document to describe the operations of the OptimizerService in detail, and the best way to determine the correct settings for an optimizer project is to create a desktop project and then map the fields you wish to set by referencing the GUI form names and mapping them to the same objects in the API Javadoc. Note that the Java API does not provide access to any of the file based methods of the underlying C++ API.

The Northfield support team is available to help you develop and test scenario prototypes, and the sample programs demonstrate how to organise and start an optimizer project.

Error Handling

Clients need to be informed as to what occurred inside an optimization process. The OptimizerService provides access to both optimizer messages and error flags with typed error codes.

ERROR MESSAGES

The optimizer messages are the text messages that appear in the window at the bottom of the Windows GUI application. Java programs can access these messages using the following methods:

```
// After a successful project run:
String[] messages = project.reports().messages();
// After an OptimizerException is thrown:
String[] messages = exception.messages();
```

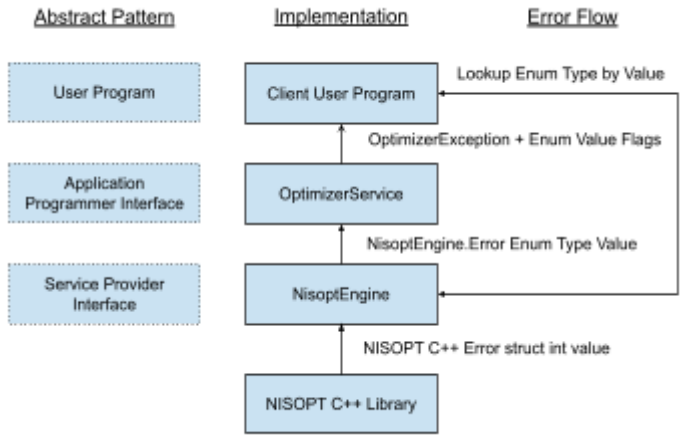
ERROR FLAGS

The service layer pattern used by the Northfield ACES Software Platform defines an abstract programmer model that isolates the client user program from the underlying implementation model. Java programs that interface to libraries via the JNI have a mismatch between the information free integer error reporting in the native library called in the JNI library and the typed exception model used in java APIs. The Northfield ACES Software Platform implements a hybrid model to resolve the mismatch that occurs between the OptimizerService that calls the C++ optimizer library via the JNI. The hybrid model implementation maps the integer constants defined in the C++ error structures to integer values mapped to Java enumerated types. This mapping is performed in the NisoptEngine plugin that plugs the C++ library into the OptimizerService.

The flow of errors codes is:

1. The C++ library returns one or more flags to the JNI program defined in C structs.
2. The JNI program converts C struct values to integer values mapped in the NisoptEngine.Error table.
3. The JNI program returns the integer values to the OptimizerService which attaches the codes to the OptimizerException returned to the User Program via the method: int[] OptimizerException.flags(); The exception is marked by the JNI library using an Exception type defined by the OptimizerException.Type.
4. The User Program queries the NisoptEngine.Error table for the enumerated type that corresponds to the flag values.

This process is shown in the following diagram.



-

The following sample shows how to map exception flags to NisoptEngine codes.

```
   try {
   ...
         OptimizerService opt = ccs.get(OptimizerService.class);
         Project project = opt.create();
         opt.run(project, ProjectTask.RUN);
       } catch (OptimizerException ex) {
         System.out.println("Service Exception:");
         Type type = ex.type();
         System.out.println("  " + type.name() + ":" + ex.getMessage());
         System.out.println("Engine Flags:");
         for (int flag : ex.flags()) {
             NisoptEngine.Error err = NisoptEngine.Error.get(flag);
             System.out.println("  " + flag + ":" + err.name());
         }
         System.out.println("Engine Messages:");
         Tools.printLog(ex.messages(), "  ");
       } ...
```

The output of this program after a failed join is:

```
==================================
    Optimizer Error Tests
==================================
Service Exception:
    |DATA_ERROR:join failed
Engine Flags:
    617:ERR_JOIN_MODEL
Engine Messages:
    Not enough data for JOIN
    Join failure
```

ERROR FLAG TYPES

The OptimizerException.Flag is an error category flag that can be accessed from:

```
OptimizerException.Flag OptimizerException.flag();
```

The type flags correspond to the JNI process of:

- copying data over (|DATA_ERROR),
- validation before processing (SETTINGS_ERROR),
- errors that occur during processing (PROCESS_ERROR), and
- an internal failure (ENGINE_ERROR).

Exception Type	Code	Text
USAGE_ERROR	100	API usage exception
	DATA_ERROR	200
SETTINGS_ERROR	300	settings/configuration error
PROCESS_ERROR	400	process error. Eg. unable find solution
ENGINE_ERROR	500	internal error

ERROR FLAG VALUES

The NisoptEngine.Error table is a lookup table that can be used to find an enumerated type that corresponds to an integer flag value returned from:

```
int[] OptimizerException.flags();
```

For the NisoptEngine, more than one flag of PROCESS_ERROR type can be returned, for all other types only one flag is expected to be returned.

The enumerated type name is derived from the C++ struct name used in the JNI library.

Exception Type	Enumerated Type Name	Code	Text
DATA_ERROR	ERR_JOIN_ZEROPRICEFOUND	601	
DATA_ERROR	ERR_JOIN_ZEROPORTVALUE	602	
DATA_ERROR	ERR_JOIN_EMPTYBENCHMARK	603	
DATA_ERROR	ERR_JOIN_COMPASS	604	
DATA_ERROR	ERR_JOIN_PORTFILE	605	
DATA_ERROR	ERR_JOIN_BUYFILE	606	
DATA_ERROR	ERR_JOIN_BENCHFILE	607	
DATA_ERROR	ERR_JOIN_DBFILE	608	
DATA_ERROR	ERR_JOIN_ALPHAFILE	609	
DATA_ERROR	ERR_JOIN_MINFILE	610	
DATA_ERROR	ERR_JOIN_MAXFILE	611	
DATA_ERROR	ERR_JOIN_MINTRADESIZEFILE	612	
DATA_ERROR	ERR_JOIN_ROUNDBASEFILE	613	
DATA_ERROR	ERR_JOIN_TRANSBUY	614	
DATA_ERROR	ERR_JOIN_TRANSSELL	615	
DATA_ERROR	ERR_JOIN_PRICEFILE	616	
DATA_ERROR	ERR_JOIN_MODEL	617	
DATA_ERROR	ERR_JOIN_CORR	618	
DATA_ERROR	ERR_JOIN_QPEN	619	
DATA_ERROR	ERR_JOIN_MODELB	620	
DATA_ERROR	ERR_JOIN_CORRB	621	
DATA_ERROR	ERR_JOIN_IND	622	
DATA_ERROR	ERR_JOIN_SECT	623	
DATA_ERROR	ERR_JOIN_MTBL	624	
DATA_ERROR	ERR_JOIN_COMPASSETS	625	
DATA_ERROR	ERR_JOIN_ACC	626	
DATA_ERROR	ERR_JOIN_XML_SOURCE	627	
DATA_ERROR	ERR_JOIN_INDMAP	628	
DATA_ERROR	ERR_JOIN_NLTC_CORR	629	
DATA_ERROR	ERR_JOIN_REFPORT	630	
DATA_ERROR	ERR_JOIN_NLTC_SYS	631	
DATA_ERROR	ERR_JOIN_RESIDUAL	632	
DATA_ERROR	ERR_JOIN_THRESHOLD	633	
DATA_ERROR	ERR_JOIN_UCITS	634	

Exception Type	Enumerated Type Name	Code	Text
SETTINGS_ERROR	ERR_JOIN	635	error during JOIN
SETTINGS_ERROR	ERR_RUN	636	error during RUN
SETTINGS_ERROR	ERR_RAPZERO	637	RAP is 0
SETTINGS_ERROR	ERR_NEGATIVEVAL	638	negative variance in the factor table
SETTINGS_ERROR	ERR_WRONGMINMAX	639	wrong min max constraints
SETTINGS_ERROR	MSG_MAXITERREACHED	640	maximum number of iterations reached
SETTINGS_ERROR	MSG_MAXTURNOVERREACHED	641	maximum turnover value is reached
SETTINGS_ERROR	MSG_MAXTURNOVERREACHED1	642	maximum turnover value is reached
SETTINGS_ERROR	MSG_CANTFINDABYBESTSWAP	643	cannot find any best stock to buy or sell
SETTINGS_ERROR	MSG_MAXPRESREACHED	644	maximum precision is reached
SETTINGS_ERROR	MSG_THRESHOLDERROR	645	cannot find any swap to solve pairing problem
SETTINGS_ERROR	MSG_MAXGAPGAINREACHED	646	maximum capital gain value is reached
PROCESS_ERROR	OPT_VIOLATIONS_SECURITY	647	violation of min/max constraints for stock
PROCESS_ERROR	OPT_VIOLATIONS_GROUP	648	industry constraints are violated
PROCESS_ERROR	OPT_VIOLATIONS_SECTOR	649	sector constraints are violated
PROCESS_ERROR	OPT_VIOLATIONS_FACTOR	650	factor constraints are violated
PROCESS_ERROR	OPT_VIOLATIONS_THRESHOLD	651	threshold constraints are violated
PROCESS_ERROR	OPT_VIOLATIONS_MINTRADESIZE	652	minimum trade size constraints are violated
PROCESS_ERROR	OPT_VIOLATIONS_MAXASSETS	653	max assets constraints are violated
PROCESS_ERROR	OPT_VIOLATIONS_LONGSHORT	654	long/short constraints are violated
PROCESS_ERROR	OPT_VIOLATIONS_MAXTRACKINGERROR	655	maximum tracking error constraints are violated
ENGINE_ERROR	ERR_WRONGINPUTSTR	656	
ENGINE_ERROR	ERR_COMPLETECALC	657	
ENGINE_ERROR	ERR_MEMALLOC	658	
ENGINE_ERROR	ERR_TWOSPACE1	659	
ENGINE_ERROR	ERR_TWOSPACE2	660	

Exception Type	Enumerated Type Name	Code	Text
ENGINE_ERROR	ERR_TWOSPACE3	661	
ENGINE_ERROR	ERR_TWOSPACE4	662	
ENGINE_ERROR	ERR_UNDEFSTEPCODE	663	
ENGINE_ERROR	MSG_WRONGVARVALUE	664	
ENGINE_ERROR	ERR_TWOSPACE1_5	665	
ENGINE_ERROR	MSG_CANCELEDBYUSER	666	
ENGINE_ERROR	MSG_INITPROBLEMFAULT	667	
ENGINE_ERROR	ERR_USERBEFOREOPT	668	
ENGINE_ERROR	ERR_STOPEED_BYCALLBACK_AFTER_EACH_ITER	669	
ENGINE_ERROR	ERR_UNDEFINED	670	

4.6.3 OPT Configuration

All configuration is normally managed by ConnectionService mode defaults.

4.6.4 Notes

- Percentage Values: Asset Weights and percentages are the same as the desktop and should be set as values between 0 - 100 where 100 = 100%. File Operations
- All file and GUI based operations are disabled in the OptimizerService. This includes the file methods in the C++ API. The reason for this is that the OptimizerService component is designed to be clustered and file based operations bind an instance to a specific machine configuration.