SAND2016-5915
Unlimited Release
Printed June 2016

Contingency Contractor Optimization Phase 3 Sustainment, Developer's Guide Contingency Contractor Optimization Tool – Prototype

Justin D. Durfee, Christopher R. Frazier, Alisa Bandlow

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: http://www.ntis.gov/search

Unlimited Release

Contingency Contractor Optimization Phase 3 Sustainment, Developer's Guide Contingency Contractor Optimization Tool – Prototype

Justin D. Durfee, Christopher R. Frazier, Alisa Bandlow
Operations Research and Knowledge Systems
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-MS1138

Abstract

This document provides background and instructions for developing and building the Contingency Contractor Optimization Tool – Prototype (CCOT-P) application.

Contents

1. INTRODUCTION

This document provides background and instructions for developing and building the Contingency Contractor Optimization Tool – Prototype (CCOT-P) application.

2. ENVIRONMENT

The following are parts of the environment for developing and building the CCOT-P application:

2.1. Operating System

CCOT-P requires Windows 7 or newer for development.

2.2. Java

The Java Development Kit (JDK), at least version 1.7, is required. A corresponding Java Runtime Environment (JRE) may be required for testing and running the application.

2.3. Eclipse

(https://eclipse.org)

Eclipse, while not specifically required, provides plugins for Google and Hibernate which facilitate development and test. At the time of the last development, Eclipse IDE for Java Developers, Luna Service Release 2 (4.4.2), was used.

2.4. Gradle

(http://gradle.org)

Gradle was chosen for the build due to its robust dependency management and concise, hierarchical logic. At the time of the last development, Grade v2.5 was used. To make variations to the build procedures, it would be best to review the Gradle documentation at the website above.

3. LIBRARIES

The following libraries are required for developing and building the CCOT-P application:

3.1. GWT

(http://www.gwtproject.org/)

GWT (formerly Google Web Toolkit) provides a set of Java APIs and Widgets to simplify web application development, permitting a developer to write completely in Java, without having to be an expert in browsers and JavaScript. Prior to deployment, Java class files are translated into corresponding HTML and JavaScript.

CCOT-P uses GWT 2.5.1. Currently, this version *must* be used, as the corresponding GXT libraries depend on it; GXT is not compatible with subsequent versions.

3.2. Sencha GXT

(https://www.sencha.com/products/gxt)

GXT is a comprehensive Java framework for building web applications. It extends GWT to provide high-performance customizable UI components. CCOT-P requires both GXT v2.2.3 and GXT v3.0.7. Note that GXT requires a license for development other than open source. Consult the website for details.

3.3. GWTUpload

(https://code.google.com/archive/p/gwt-upload/)

GWTUpload provides an API allowing a browser client to upload files to the web application. This is used for uploading TPFDDs. GWTUpload v1.0.1 was used for the last CCOT-P build, though it is possible that subsequent versions may be used.

3.4. Hibernate ORM

(http://hibernate.org/orm/)

Hibernate provides an intuitive Java persistence framework for working with databases, allowing developers to map plain old Java objects (POJOs) directly to database table entries. At the time of the last CCOT-P build, Hibernate v4.2.17 was used, though it is possible that subsequent versions may be used.

3.5. Apache POI

(https://poi.apache.org/)

Apache POI provides Java APIs for reading and writing Microsoft Office files. CCOT-P uses this to read and write Microsoft Excel files. At the time of the last CCOT-P build, POI v3.11 was used, though it is possible that subsequent versions may be used.

3.6. Microsoft SQL JDBC

(https://www.microsoft.com/en-us/download/details.aspx?id=11774)

Hibernate ORM requires the Microsoft SQL Server JDBC library to access the SQL Server instance used by CCOT-P. At the time of that last CCOT-P build, SQL JDBC v4.1 was used, though it is possible that subsequent versions may be used.

3.7. Apache Commons

(https://commons.apache.org/)

The following components from the Apache Commons project and their respect versions are as follows:

- BeanUtils v1.8.3
- Collections v3.1
- FileUpload v1.2.2
- Logging v1.1.1

3.8. CPLEX

(http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud)

CCOT-P uses the IBM ILOG CPLEX commercial solver to execute its optimization models. It utilizes Java API through the cplex.jar library. After installing CPLEX, the developer will need to provide a link to this library to build the web application. CCOT-P uses CPLEX Optimization Studio v12.4.

4. PROJECTS

CCOT-P consists of three projects.

4.1. CCOT (Web Application)

The bulk of CCOT-P exists in the web application project. There are client and server packages which contain their respective classes. Classes in the server package will form the Java back end servlet, while the client classes (associated with the UI) will be translated by the GWT compiler into HTML and Javascript that will be served to the client upon browser invocation. The final product is a Java .war file which needs to be hosted by a Java server such as Tomcat, Jetty, or JBOSS. Review the GWT documentation (http://www.gwtproject.org/) for more information.

4.2. CCOT Solver

The CCOT Solver project provides encapsulated access to CPLEX to solve the CCOT-P optimization, so that the web application and solver do not need to reside on the same machine. The CCOT Solver creates a Java Remote Method Invocation (RMI) server called by the web application to solve a particular application. The final product of the build is a Java .jar file that can either be invoked manually, or installed as a Windows service using Apache Procrun (http://commons.apache.org/proper/commons-daemon/procrun.html). The CCOT Solver and CPLEX must be installed on the same machine.

4.3. CCOT Poller (optional)

The CCOT Poller project is an optional diagnostic tool which periodically polls the web application, solver, and database, and provides a simple web page of the results. The final product of this build is also a Java .jar file that can either invoked manually or installed as a Windows service.

5. CONFIGURATION

Following are steps necessary to configure your development environment for CCOT-P. This assumes the developer is using the environment described above (Eclipse).

1. Install the Google and Hibernate plugins in Eclipse.
2. If required, download GWT-2.5.1 and install the gwt-2.5.1 SDK in Eclipse. (The Google plugin will likely have a newer version.)
3. If required, add jre7 to Eclipse (Windows->Preferences->Java->Installed JREs).
4. If not already specified, set the JAVA_HOME environment variable to the JDK installation directory (i.e. "C:\Program Files\Java\jdk7"). Gradle requires this to function.

   NOTE: While either Java 7 or 8 may be used for compiling and running CCOT and its components, you should use the same version for all components. Otherwise, you may get version errors when you try to run.

5. CPLEX will most likely need to be installed on both the system you are using for development as well as the system which is running the solver. (These may be the same system.) On the development system, it may be easiest to provide a symbolic link to the CPLEX Java library as follows:
   a. Open a command prompt as an Administrator.
   b. Switch the to the "lib" directory under the CcotSolver project.
   c. Use "mklink" to create a symbolic link to your installation of CPLEX, substituting the appropriate directories below:

   ```
   mklink cplex.jar "C:\Program
   Files\IBM\ILOG\CPLEX_Studio124\cplex\lib\cplex.jar"
   ```

6. You may be able to quickly run and debug CCOT using GWT's devmode ("Run As -> Web Application (GWT Dev Mode)"), which will start up an internal Jetty instance.
7. Since Gradle is the tool used for building CCOT, you will likely want to create either a symbolic link to "gradle" or "gradle.bat" (in the gradle\bin directory) or add it to your path.

6. BUILD PROCEDURES

Gradle provides several build tasks, the most significant of which are described below. Type "gradle tasks" from the command line to list all available tasks.

6.1. Distribution Task ('gradle dist')

This is the task a developer will likely run most often. From the top Java project directory (which contains the Ccot, CcotSolver, and CcotPoller directories), issue the "gradle dist" command from the command line.

This will build the three components of CCOT (web application, solver, and poller) and copy the documentation and third party libraries into a 'dist' directory. From there, one should follow the *CCOT-P Installation Instructions* to deploy.

Note that you may see warnings about Java versions if you are using Java 8 but compiling for Java 7 (typical). You may also see warnings during the 'compileGwt' task as GWT tries to match its configurations to specific browser versions.

6.2. Clean Task ('gradle clean')

The clean task removes all of the built artifacts. This may be run either from the top Java project directory to clean all projects, or from within any of the project directories to clean the individual projects.

6.3. Build Task ('gradle build')

This task builds the project binary files. It may also be called from either the top Java project directory or from within any of the project directories to build only a specific project. *Note that this task builds the project(s) but does not copy over any dependencies, documentation, etc.* A project's binary build will be put into the respective project's build/libs directory. For the CCOT web project, this is a .war file. For the CcotSolver and CcotPoller projects, these are .jar files.

DISTRIBUTION

1 MS0899 Technical Library 9536


Sandia National Laboratories