

## 1.0 INTRODUCTION

This package contains a set of Windows™ executable applications based on components of the JTNC DoD IR tool suite. The DoD IR tool suite is a collection of Python, C#, Java and other software and scripting services used to automatically scan technical data and other artifacts presented to the JTNC organization for DoD community reuse.

The full DoD IR tool suite contains a number of tools that require a specialized PC workstation setup to operate the complete set of tool chain services. While many of these tools address specific JTNC processing requirements, JTNC community members and other organizations have found a number of the tools to be useful.






The PC workstation environment dependencies limit distribution of the services to stakeholders. To address this issue, the DoD IR team publishes specific toolbox services as ‘standalone’ publications that can typically be installed on PC workstations without requiring administrative rights or the installation of specific applications and support infrastructures. Check with your IT department to ensure the installation is conformant with your organization’s policies.

This package contains the standalone instance of the JTNC Software Communications Architecture (SCA) Application Program Interface (API) scan tooling and simplified report generator. These tools enable the user to perform analysis of a code base to determine usage of Standard JTNC SCA and API services as published to the DoD Information Technology Standards Registry (DISR). The JTNC API scanner is provided in two formats.

- The first is limited to ‘DoD Distribution Statement D’ and is also restricted by the Arms Export Control and Export Administration Acts.
- The second is approved for public release and is governed by ‘DoD Distribution Statement A.’

## 2.0 PACKAGE ORGANIZATION

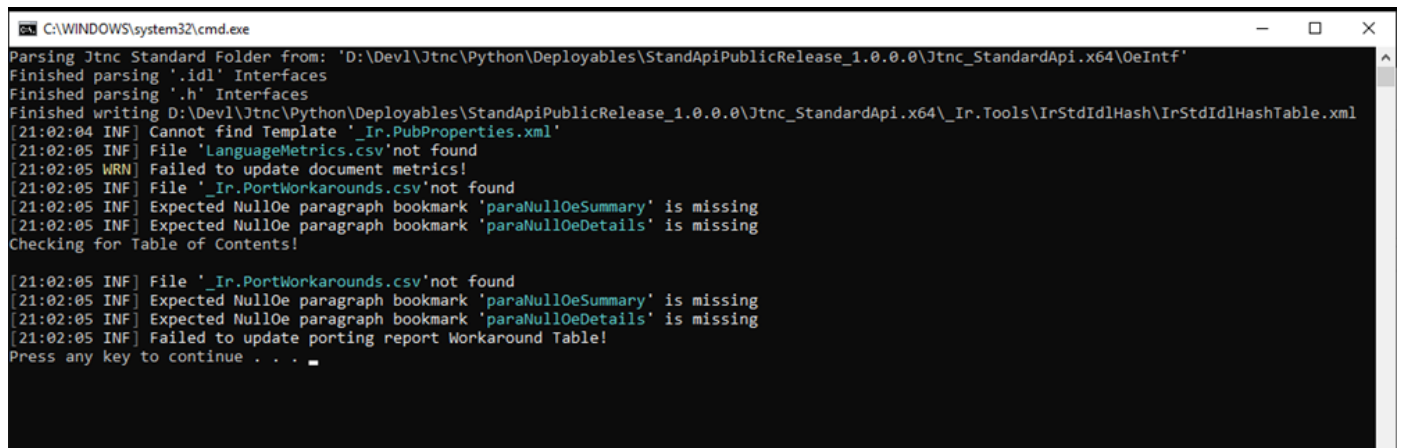
The JTNC Standard API scanner is delivered in the following folder structure:

- ▼  **Jtnc\_StandardApi.x64** • *Jtnc\_StandardApi.x64* – indicating this is a 64 bit version (.x86 are 32 bit)
- >  **\_Ir.Tools** • *\_Ir.Tools* – Report generator executable & support .dll’s
- >  **bin** • *bin* – executables created from Python scripting tools
- >  **OeIntf** • *OeIntf* – JTNC API sources (Public or Distribution D Release)
- >  **Sample** • *Sample* – A simple sample to validate operations

The root of the Jtnc\_StandardApi.x64 package contains simple windows ‘batch’ scripts that allow the user to verify the operation of the system after installation. The system requires the installation of .Net version 4.7 or later to operate. The user should verify a compatible installation is available. In most instances, the system will notify you if a compatible version of .Net is unavailable. Microsoft Word version 2013 or later must also be installed to support report generation.

The batch scripts are the following:

- **SampleRun.bat** – Clicking on this should produce a console window similar to the one below.



The logger entries marked ‘INF’ are informational only and can be ignored.

```
[21:02:04 INF] Cannot find Template '_Ir.PubProperties.xml'
[21:02:05 INF] File 'LanguageMetrics.csv' not found
[21:02:05 WRN] Failed to update document metrics!
[21:02:05 INF] File '_Ir.PortWorkarounds.csv' not found
[21:02:05 INF] Expected NullOe paragraph bookmark 'paraNullOeSummary' is missing
[21:02:05 INF] Expected NullOe paragraph bookmark 'paraNullOeDetails' is missing
Checking for Table of Contents!
[21:02:05 INF] File '_Ir.PortWorkarounds.csv' not found
[21:02:05 INF] Expected NullOe paragraph bookmark 'paraNullOeSummary' is missing
[21:02:05 INF] Expected NullOe paragraph bookmark 'paraNullOeDetails' is missing
[21:02:05 INF] Failed to update porting report Workaround Table!
Press any key to continue . . .
```

Other error or warning messages may provide sufficient information to allow users to resolve. If you are unable to correct the problems, please notify the JTNC support team at [DoD\\_IR@navy.mil](mailto:DoD_IR@navy.mil).

While the process is running, you should see a word document open which is then populated with the results from the scan performed on the ‘Sample’ data provided. Navigate to the ‘Sample’ folder and open the document ‘\_Ir.BldRpt.docx’ to see the results. The document is generated based on the material found in the StdIdlCheckResults.xml & StdIdlCheckResults.csv files also found in the Sample folder.

- **SetEnv.bat** – contains the scripts needed to set the PATH environment to support the tool execution.

### 3.0 BIN & \_IR.TOOLS FOLDERS

These folders contain the executables and supporting utilities required for the application. Each of the tools provide command line help using ‘--help or -h’ as an argument. We should note that although the tooling help will display the original ‘python’ filename of the executable, the required python support engines are included in the bin directory to avoid collision with existing python workstation installations or the need for the user to install the python services. The principal executables provided are documented below.

#### 3.1 GenCodeHashTableStdIdl.exe: Generate table of MD5 hashes for standard API files

This utility generates the hash tables for the standard API sources files that is compared against provided IDL (see: OeIntf folder). This hash is produced by removing all comments and whitespace from the source file and generating a hash value based on the executable/compliable content only. This process only needs to be run when changes are made to the OeIntf folder contents.

Command line help displays:

```
usage: GenCodeHashTableStdIdl.py [-h | --help] [-i oeintf_folder] [-o outfile.xml]
  oeintf_folder:  path to standard IDL (defaults $IR_ROOT/OeIntf/Jtnc/OeIntf)
  outfile.xml:XML results written to the file name provided
                  (defaults $IR_ROOT/_Ir.Tools/IrStdIdlHash/IrStdIdlHashTable.xml)
  -h:             this!
```

All of the arguments are optional.

-i identifies the ‘input’ API folder tree and defaults to ‘OeIntf’ for the current installation  
-o identifies to ‘output’ location for xml hash data generated and defaults to Ir.Tools/IrStdIdlHash/IrStdIdlHashTable.xml

It is highly recommended that you do not deviate from the defaults.

#### 3.2 GenStdIdlCheck.exe: Generate XML log for the results of comparing API files against the standards

This utility locates and generates hash tables for a body of IDL code that is then compared against the standard reference API data produced by the GenCodeHashTableStdIdl.exe source files. The tool then creates an xml file with the results found for each of the IDL source files found within the code collection.

Command line help displays:

```
usage: GenStdIdlCheck [-d directory] [-i standard_idl.xml] [-o outputfile.xml] [-h]
```

directory is the folder containing IDL to compare with the standard (defaults to '.')  
standard\_idl.xml: name of the file containing the results of GenCodeHashTableStdIdl.py  
defaults to \${IR\_ROOT}/\_Ir.Tools/IrStdIdlHash/IrStdIdlHashTable.xml  
if IR\_ROOT is not defined, it defaults to '/Jtnc' in the path to standard\_file\_name.xml  
outputfile.xml: XML results written to the file name provided, defaults stdout  
-h display this help

All of the arguments are optional.

- d directory Identifies the folder path to search for IDL and candidate header files for comparison with JTNC Standards. The directory tree is recursively searched. The default is the current directory.
- i standard\_idl.xml Identifies the xml hash data produce for the JTNC Standard API's produced by GenCodeHashTableStdIdl and defaults to Ir.Tools/IrStdIdlHash/IrStdIdlHashTable.xml which is the default location for this metadata.
- o output.xml By default, the xml output produced by the tool is written to the standard output. The batch script provided will write the xml data to the cleverly named file 'StdIdlCheckResults.xml' and can be found in the 'Sample' folder.

### 3.3 GenStdIdlCheckCsv.exe: Post-processes XML log file output from GenStdIdlCheck.py into a CSV file

The current version of the report generator is dependent on comma-separated-values (CSV) formatted files for table generation purposes. Future editions of the tooling will use XML data sets. Currently an additional step is required to convert the XML content produced by GenStdIdlCheck to CSV format for the current report generator.

Command line help displays:

```
usage: GenStdIdlCheckCsv.py [-i standard_file.xml] [-o outputfile] [-h]
-i standard_file.xml where input is XML log file generated by GenStdIdlCheck.py, defaults to stdin
-o outputfile XML results written to the file name provided, defaults stdout
-h is print help
CSV results file is output
```

All of the arguments are optional.

- i standard\_file.xml An XML metadata file produced by GenStdIdlCheck or reads from standard input if the argument is unused. The batch scripting provided uses the file name 'StdIdlCheckResults.xml' which can be found in the folder 'Sample'.
- o outputfile The CSV file used by the report generator to produce the standard API's findings. The batch scripting provided creates the file name StdIdlCheckResults.csv that can be found in the 'Sample' folder.

### 3.4 RptGenerator.exe: Report Generator

This utility is a C# based Windows .Net dependent application used to create a simple report providing the results of the JTNC Standard API comparisons with the code base under evaluation. The example provided will produce two tables, with one identifying the use or potential misuse of JTNC standard API's. This utility is currently under revision to provide additional flexibility in supporting user-specific extensions to the tooling.

Command line help displays:

-g, --generic	(Default: False) Generic Report you must provide a template - icons & merge document are optional
-w, --waveform	(Default: False) Select Waveform Report
-o, --operating-environment	(Default: False) Select Operating Environment Report
-n, --network-manager	(Default: False) Select a Network Manager Report
-r, --radio-service	(Default: False) Select Radio Service Report
-v, --visible	(Default: False) Enable word visibility (debugging)
-l, --log-level	(Default: Warning) Set the logger level (Verbose, Debug, Information, Warning, Error, Fatal)
-t, --template	(Default: ) Provide the full-path name to the generator template document (will search 'exePath!/Templates)
-m, --merge-doc	(Default: ) Provide the full-path name to the generator merge overlay document (will select latest numbered version)
-b, --build-report	(Default: False) Select a NullOe Engineering Build Report
-d, --draft-report-name	(Default: _Ir.BldRpt.docx) Provide the full-path name of the Generator 'Draft Output'
-i, --icon-files	Comma separated list of branding icons
--help	Display this help screen

This utility produces documents by referencing a user-provided 'Template' with bookmarks and 'boiler-plate' wording used in the production of a standardized report document. The template output can also be altered by the addition of a 'merge' document to provide instance specific reporting.

The utility was originally developed to automate the reports generated by the JTNC and legacy JTRS repository processing. As a result, the 'waveform, operating-environment, network-manager, radio-service and build-report' arguments pertain to the selection of the standardized document templates and merge element specific to the JTNC organization. The 'generic' argument provides the user with the ability to specify unique template and merge inputs to the

generator. The batch script does this through the execution of the 'Sample/RptGen.bat' file. This batch script executes the command line:

```
RptGenerator - gv -t StdApi.dotx
```

This causes a 'generic' run using the file 'StdApi.dotx' as the template file with no merge documentation. The 'visible' argument causes the generation of the document to be displayed by the word application as it is created. The report template name '\_Ir.BldRpt.docx' is used by default.

#### **4.0 USAGE RECOMMENDATIONS**

Users may wish to modify the provided batch files to support reporting requirements. Future updates to this tooling will follow a similar deployment pattern resulting in the creation of version-specific folder trees to reduce the possibility of accidental overwrite of existing content. Although the tooling should not modify any of the software artifacts found in the code base under review, the pattern of placing a copy of the code for testing in a subfolder within the deployed API service folder tree and navigating to this folder for the generation of the reporting material should limit exposure to any unknown processing issues.

Please provide comments and recommendations to [DoD IR@navy.mil](mailto:DoD_IR@navy.mil).

## APPENDIX A SAMPLE REPORT

The following is an example of a report generated from the ‘Sample’ provided.

### A.1 STANDARD API CHECK

Table 1 identifies the standard JTNC APIs provided with the product. It is important to understand that the IDL provided by the developer may represent platform implementation details and do not imply a requirement for use by this specific waveform implementation. Actual IDL interface usage is documented in the NullOe build report for the waveform.

**Table A-1. JTNC Standard APIs Provided**

API	Type	API File	Version	Notes
<b>Legacy Branch or Deprecated API</b>				
	Base	LogService.idl	None	-2.2
<b>Packet API</b>				
	Base	Packet.idl	2.0.2	(2.0.1 +9)
	Extension	PacketEmptySignals.idl	2.0.2	(2.0.1 +9)
	Extension	PacketFlowControl.idl	2.0.2	(2.0.1 +9)
	Base	RenamePacket.idl	2.0.2	(2.0.1 +9) name mismatch: sb:Packet.idl
<b>SCA Core Framework</b>				
	Standard	CF.idl	None	-2.2
	Standard	PortTypes.idl	None	-2.2
	Standard	StandardEvent.idl	None	-2.2
	Standard	CF.idl	2.2.2	(2.2.2)
	Standard	PortTypes.idl	2.2.2	(2.2.2)
	Standard	StandardEvent.idl	2.2.2	(2.2.2)
	Standard	CF.idl	4.1	-4.1
	Standard	porttypes.idl	None	(2.2) name mismatch: sb:PortTypes.idl
	Standard	Renamed_CF.idl	None	(2.2) name mismatch: sb:CF.idl
	Standard	cf.idl	2.2.2	(2.2.2) name mismatch: sb:CF.idl
	Standard	RenamedStandardEvent.idl	2.2.2	(2.2.2) name mismatch: sb:StandardEvent.idl
	Standard	cfAggregatedevice.idl	4.1	(4.1) name mismatch: sb:CFAggregateDevice.idl
	Standard	CFDomainManager.idl	4.1	-4.1
	Standard	Renamed_CFAggregateDeviceAttributes.idl	4.1	(4.1) name mismatch: sb:CFAggregateDeviceAttributes.idl

While Table A-1 shows the standard JTNC APIs provided with the product, a comprehensive summary of all IDL files (JTNC and non-JTNC) from the delivery can be found in file

A-1

‘StdIdlCheckSummary.csv’ located in the product distribution ‘Ir.Notes’ folder. The comprehensive summary categorizes the IDL files as:

- Software Communications Architecture (SCA) Standard
- JTNC Standard
- Object Management Group (OMG)
- Application Specific

## A.2 APPLICATION APIs PROVIDED

Table A-2 identifies the collection of CORBA-based APIs provided with the waveform application. As mentioned earlier, the additional IDL provided by the developer should only define interfaces required between waveform application components. The realization and implementation of these interfaces by the applications’ components must be complete for the waveform to be considered SCA conformant. Any disparity between the source code ‘as provided’ in the waveform application and any platform-specific implementation dependencies are identified in the NullOe build report.

**Table A-2. Other APIs Provided**

API	API File	Notes
<b>Application Unique Interface</b>		
	JtrsCorbaTypes.idl	
	LogService.idl	
	PortTypes.idl	
	CF.idl	
	CFAdministratableInterface.idl	
	jtrsCorbaTypes.idl	
	PacketEmptySignals.idl	
	CosNaming.idl	
	CSI.idl	
	CSIIOP.idl	
	Dynamic.idl	
	DynamicAny.idl	
	FT.idl	
	GIOP.idl	
	GSSUP.idl	
	IIOP.idl	
	IOP.idl	