Utilities

Detailed information on XINA utility applications.

- XINA Connect
- XINA Tunnel
- XINA Import
- XINA Import Helper
- XINA Download
- XINA API Libraries
- XINA Struct Archive
- XINA Struct Mine
- XINA Struct Export
- XINA Struct Trend
- XINA Struct Report
- XINA Struct Report Email

XINA Connect

XINA Connect is a Java application which combines the features of the XINA Tunnel and XINA Import applicatons with a UI. This application is distributed as a Java jar file and requires Java 17 or greater to run. The recommended OpenJDK build is available here.

If you run into security issues with the Windows Portable versions, you can try this workaround:

1. Right click the executable -> Properties -> Attributes
2. Uncheck the checkbox saying that this file came from another computer and might be dangerous
3. Apply
4. Try running the exe again

Latest Version

13.32.1

Windows Portable (no install required)

1. Download
2. Extract to desired location
3. Run `XINA Connect`
4. Create shortcuts as needed

Jar File

```
java -jar xina-connect.jar
```

Changelog

- Add support for "move on error" action. This is now required for `Watch` to prevent `Watch` from getting stuck when it encounters a bad import file.
- Add support for "trash on import" action
- Fix manual import missing the success and error move actions, which was breaking the ability to manually import files
- Fix crash when quitting
- Other UI/UX tweaks

Previous Versions

13.3.0

Jar File (requires Java 21+)

```
java -jar xina-connect.jar
```

1.1.0

Windows Installer

Windows Portable

MacOS Installer

Jar File (requires Java 17+)

```
java -jar xina_connect.jar
```

1.0.0

Windows Installer

MacOS Installer

XINA Tunnel

XINA Tunnel is a command line application required for client applications to communicate with the XINA Server via the XINA API. This application is distributed as a Java jar file and requires Java 21 or greater to run. The recommended OpenJDK build is available here.

Latest Version

XINA Tunnel 14.4.2

Download

Usage

Windows

```
java -jar "path to xina_tunnel.jar" ^
    -host "host URL or IP address" ^
    -keyfile "path to key.json"
```

MacOS, Linux

```
java -jar "path to xina_tunnel.jar" \
    -host "host URL or IP address" \
    -keyfile "path to key.json"
```

Old Versions

XINA Tunnel 9.2.0

Download

Usage

Windows

```
java -jar "path to xina_tunnel.jar" ^
    -host "host URL or IP address" ^
    -keyfile "path to key.json"
```

MacOS, Linux

```
java -jar "path to xina_tunnel.jar" \
    -host "host URL or IP address" \
    -keyfile "path to key.json"
```

XINA Import

XINA Import is a utility for importing XINA API actions as JSON files. This application is distributed as a Java jar file and requires Java 21 or greater to run. The recommended OpenJDK build is available here.

Latest Version

XINA Import 14.4.2

Download

Windows

```
java -jar "path to xina_import.jar" ^
    [additional arguments...]
```

MacOS, Linux

```
java -jar "path to xina_import.jar" \
    [additional arguments...]
```

Argument	Info	Default
`-host <hostname>`	the XINA Tunnel host	`"localhost"`
`-post <port>`	the XINA Tunnel port	`41746`
`-okmove <path>`	if set, move files to path to after successful import	none (files are not moved)
`-oktrash`	if set, move files to OS trash after successful import	`false`
`-okdelete`	if set, permanently delete files after successful import	`false`
`-ermove <path>`	if set, move files to path to after failed import	none (errors stop import)
`-dir <path>`	path to directory containing files to import	
`-watch <path>`	path to directory to watch for files to import	
`-recursive`	if true in `dir` or `watch` mode, searches directory recursively for JSON files	`false`

XINA Import has three modes of operation:

File List

JSON files can be listed explicitly. They will be imported in the specified order.

```
java -jar "path to xina_import.jar" \
    "path to JSON file" \
    "path to JSON file"...
```

Directory

If the `-dir` argument is used, XINA Import will attempt to import all `*.json` files in the specified directory in alphanumeric order.

Note: the directory is now queried after each imported file, so at one of `-okmove`, `-oktrash`, or `-okdelete` must be specified in this mode, or the same file would be imported repeatedly in an infinite loop.

```
java -jar "path to xina_import.jar" \
    -dir "path to directory" \
    -okmove "path to different directory"
```

Watch

If the `-watch` argument is used, XINA Import will attempt to import all `*.json` files in the specified directory in alphanumeric order. Once complete, the directory is watched for any new JSON files, which are imported as they become available. As with directory mode, a `-ok*` argument is required to prevent the same file from being imported repeatedly.

```
java -jar "path to xina_import.jar" \
    -watch "path to directory" \
    -okmove "path to different directory"
```

Previous Versions

XINA Import 10.1.0

Download

Windows

```
java -jar "path to xina_import.jar" ^
    [additional arguments...]
```

MacOS, Linux

```
java -jar "path to xina_import.jar" \
    [additional arguments...]
```

Argument	Info	Default

`-host <hostname>`	the XINA Tunnel host	`"localhost"`
`-post <port>`	the XINA Tunnel port	`41746`
`-okmove <path>`	if set, move files to path to after successful import	none (files are not moved)
`-oktrash`	if set, move files to OS trash after successful import	`false`
`-okdelete`	if set, permanently delete files after successful import	`false`
`-ermove`	if set, move files to path to after failed import	none (errors stop import)
`-dir <path>`	path to directory containing files to import	
`-watch <path>`	path to directory to watch for files to import	
`-recursive`	if true in `dir` or `watch` mode, searches directory recursively for JSON files	`false`

XINA Import has three modes of operation:

File List

JSON files can be listed explicitly. They will be imported in the specified order.

```
java -jar "path to xina_import.jar" \
    "path to JSON file" \
    "path to JSON file"...
```

Directory

If the `-dir` argument is used, XINA Import will attempt to import all `*.json` files in the specified directory in alphanumeric order.

Note: the directory is now queried after each imported file, so at one of `-okmove`, `-oktrash`, or `-okdelete` must be specified in this mode, or the same file would be imported repeatedly in an infinite loop.

```
java -jar "path to xina_import.jar" \
    -dir "path to directory" \
    -okmove "path to different directory"
```

Watch

If the `-watch` argument is used, XINA Import will attempt to import all `*.json` files in the specified directory in alphanumeric order. Once complete, the directory is watched for any new JSON files, which are imported as they become available. As with directory mode, a `-ok*` argument is required to prevent the same file from being imported repeatedly.

```
java -jar "path to xina_import.jar" \
    -watch "path to directory" \
```

```
    -okmove "path to different directory"
```

XINA Import 9.2.0

Download

Usage

Windows

```
java -Dlog4j.configurationFile="path to log4j2.xml" ^
    -jar "path to xina_import.jar" ^
    [additional arguments...]
```

MacOS, Linux

```
java -Dlog4j.configurationFile="path to log4j2.xml" \
    -jar "path to xina_import.jar" \
    [additional arguments...]
```

Argument	Info	Default
`-host <hostname>`	the XINA Tunnel host	`"localhost"`
`-post <port>`	the XINA Tunnel port	`41746`
`-movejson <path>`	directory to move JSON files to after import	none (files are not moved)
`-movefile <path>`	directory to move other files to after import	none (files are not moved)
`-deljson`	if set, permanently delete JSON files after import	`false`
`-delfile`	if set, permanently delete other files after import	`false`
`-dir <path>`	path to directory containing files to import	
`-watch <path>`	path to directory to watch for files to import	
`-recursive`	if true in `dir` or `watch` mode, searches directory recursively for JSON files	`false`

XINA Import has three modes of operation:

File List

JSON files can be listed explicitly. They will be imported in the specified order.

```
java -jar "path to xina_import.jar" \
    "path to JSON file" \
    "path to JSON file"...
```

Directory

If the `-dir` argument is used, XINA Import will attempt to import all `*.json` files in the specified directory in alphabetical order. It is recommended to include `-movejson` or `-deljson` to track progress, in case the import is interrupted.

```
java -jar "path to xina_import.jar" \
    -dir "path to directory" \
    -movejson "path to different directory"
```

Watch

If the `-watch` argument is used, XINA Import will attempt to import all `*.json` files in the specified directory in alphabetical order. Once complete, the directory is watched for any new JSON files, which are imported as they become available. It is required to include `-movejson` or `-deljson`, as otherwise files would be continuously re-imported.

```
java -jar "path to xina_import.jar" \
    -watch "path to directory" \
    -movejson "path to different directory"
```

XINA Import Helper

XINA Import Helper is a utility for generating the XINA API actions as JSON files for an archive directory. The following XINA action files will be generated:

- Cancel any existing task(s) for the archive
- Delete the archive if it already exists in XINA
- Import the file archive
- Run the mining task on the archive to extract the data

The output files can then be imported into XINA using XINA Import. This application is distributed as a Java jar file and requires Java 17 or greater to run. The recommended OpenJDK build is available here.

Latest Version

XINA Import Helper 11.2.3

Download

Windows

```
java -jar "path to xina-import-helper.jar" ^
    [additional arguments...]
```

MacOS, Linux

```
java -jar "path to xina-import-helper.jar" \
    [additional arguments...]
```

XINA Download

XINA Download is a utility for managing batch file downloads with XINA. This application currently works with the SAM and MOMA XINA instances for telemetry file management only.

Latest Version

Windows Installer

xina_download.exe

MacOS Installer

xina_download.dmg

XINA API Libraries

We aim to provide a variety of reference XINA API client implementations for different programming languages and environments.

Python

xina_api_python_0.4.0.zip

Installation

Prerequisites

- Python 3.8 or greater

The `xina` client communicates with XINA via the XINA Tunnel utility. You must have it running for the `xina` client to connect.

Steps

1. Extract the zip file.
2. Open a terminal in the root extracted directory, and execute the following commands:

```
python3 -m pip install .
```

Examples

Client class

```
from xina import XPClient


with XPClient() as x:
    res = x.act({'action': 'version'})


print(res)
# => {'schema':100, 'host':'sandbox.xina.io', 'server':'9.1.4', 'team':100}
```

CLI

```
python -m xina action '{"action":"version"}'
# => {"schema":100,"host":"sandbox.xina.io","server":"9.1.4","team":100}
```

XINA Struct Archive

The XINA Struct Archive utility processes mnemonic buffer files into mnemonic archive files. It is primarily intended to be run as part of the automated XINA Structs data pipeline.

Arguments

Name	Req	Description	Default
task	?	task ID	
conf	?	JSON conf file path	
temp	?	temp directory path	
import	?	import directory path (for file outputs)	
post	?	post directory path (for post-import outputs)	
host		tunnel host	`"localhost"`
port		tunnel port	`41746`

Configuration

Name	Req	Description	Default
origin	?	origin group	
slice		slice length in minutes	`60`
t		slice time barrier (see below)	

Operation

This utility serves three main functions. First, processing raw buffer files of any supported format into well formatted, optimized xbin files. Second, merging buffer file data into time slice separated archive files. Third, processing any changes of flagged buffer files and updating associated archives as needed.

File Processing

Buffer files are typically imported in the `PENDING` state, and must be converted into xbin files, which are indicated with the `PROCESSED` state.

XINA Struct Mine

The XINA Struct Mine utility processes archive files to produce data products for import into XINA. It is primarily intended to be run as part of the XINA Structs data pipeline's Mining Task. There are 2 suppported archive file formats: `xbin` and `xpf`. If the archive file is in the `xpf` format, processing is delegated to a mission specific tool. See the Struct Extract Interface that describes the interface the tool shall implement.

Arguments

Name	Req	Description	Default
task	?	task ID	
conf	?	JSON conf file path	
temp	?	temp directory path	
import	?	import directory path (for file outputs)	
post	?	post directory path (for post-import outputs)	
host		tunnel host	`"localhost"`
port		tunnel port	`41746`

Configuration

Name	Req	Description	Default
pipe	?	pipe group path, or model path if no pipe	
t	?	start time of archive to mine. Can be specified as either a numeric microseconds since unix epoch or an ISO8601 string such as "2025-02-17T00:00:00Z"	
use_pkt_time		If true, use packet time when extracting data. Only applicable for projects that support both ground and packet time.	false
pkt		**Deprecated** alias for `use_pkt_time`	
write_unbinned		If true, write the unbinned (aka delta/full) data files and import actions.	true

Name	Req	Description	Default
write_binned_secs		Array of binned db seconds to write and import. An empty array will not write any binned data. Not including the key will write all binned data. Example: `[60]` will write only the 60 second binned data.	All binned data will be written and imported

Operation

This utility serves a few main functions:

- Extract data from archive files to produce import products such as:
  - Full mnemonic data
  - Delta mnemonic data
  - Time and Interval bin data
  - Filtered mnemonic data
  - Events (Actions?)
  - Limit Report
  - Metrics
  - Mission Specific Data

XINA Struct Export

The XINA Struct Export command line utility processes archive files for a specified time range to produce data products for XINA's Export Tasks. There are 2 suppported archive file formats: `xbin` and `xpf`. If the archive file is in the `xpf` format, processing is delegated to a mission specific tool. See the Struct Extract Interface that describes the interface the tool must implement. Available data products include:

- Full resolution mnemonic data with a configurable format
- Mnemonic data statistics
- Configurable plots of data
- Limit Report
- Events
- Log file

Arguments

Name	Req	Description	Default
cache		Cache directory path. Used for any persistent caching across tasks for optimization purposes.	
cancellation		Path to the cancellation file for detecting cancel requests.	
conf	?	File path to the JSON config containing the export parameters.	
env	?	File path to the JSON environment file which defines environment parameters such as project specific paths.	
extract		Overrides the path provided in the environment file to the struct_extract app. Only used if the archive format is `xpf`.	
help		Print available options.	
host		XINA Tunnel host.	`"localhost"`
port		XINA Tunnel port.	`41746`
import	?	Import directory path. Files that need to be imported will be placed here. `struct_export` is responsbile for generating the corresponding import action file but `xina-run` will perform the actual importing.	
log	?	File path that log events will be written to.	

Name	Req	Description	Default
out	?	Ouptut directory path. Files placed here will be uploaded and accessible directly from the Task interface (e.g. notifications panel).	
pkt_models		Models to use packet time by default. **[TBD Remove]**	
plot	?	Overrides the path provided in the environment file to the app that generates plot PDFs.	
post	?	Post directory path (for post-import outputs) e.g. uploading log files to the Task record.	
python	?	Overrides the path to the Python executable used by the plot app provided in the environment file.	
task	?	File path to the JSON file containing info about the Task.	
temp	?	Temp directory path for storing temporary files during execution. For example, zipped archives are unzipped to this directory for processing.	

conf

The below table describes the available fields in the `conf` JSON file and is used to control how data should be exported.

Name	Type	Req	Description	Default
model	`utf8text`	?	Path of model to export data from e.g. `moma.fm`	
start	`instant(us)`	?	Start time of data to export. May be provided as a Unix timestamp in microseconds (e.g. `1738781947000000`) or an ISO 8601 formatted string (e.g. `2024-06-12T00:00:00Z`).	
end	`instant(us)`	?	End time of data to export. May be provided as a Unix timestamp in microseconds (e.g. `1738781947000000`) or an ISO 8601 formatted string (e.g. `2024-06-12T00:00:00Z`).	

Name	Type	Req	Description	Default
ueid	`UUID`		Event UUID of interval if the export was requested for an interval. The event's start and end time will be used instead of `start` and `end`.	
disable_filter	`boolean`		If `true`, does not apply the filters defined in `data_conf`. This is useful if you want to export an existing profile that has filters, but don't want to apply the filters.	false
copy	`boolean`		If true, the zip file will be uploaded and available from the Task UI.	false
label	`utf8text`		The text that will be used to name the files and final zip file. The format will be like `2024_06_12_00_00_00_2024_06_12_00_20_00_profile_label`	
profile	`utf8text`		The name of the Profile if the export was generated from one. If provided, then the Profile's `plot_conf` and `data_conf` will be used.	
auto_conf	`struct_auto_conf`		The `struct_auto_conf` that triggered the export. Only provided if the export was auto generated.	
plot_conf	`struct_plot_conf`		The plot configuration used to generate the plot PDF. See plot format. An empty object i.e. `{}` can be provided to generate the default plot configuration. If not provided, then plots will not be generated.	One mnemonic per plot, one plot per page, sorted by `mn_id` in ascending order.
data_conf	`struct_data_conf`		See struct_data_conf	
multi	`boolean`		**[TBD REMOVE - Not currently supported]** Has extract process all archives at once. This was added for performance reasons since loading the mnem def is slow.	

Example conf:

Note: The plot_conf was truncated for brevity.

This example performs an export with fields provided directly in the `conf` object.


```
{
  "model": "oci.fm",
  "label": "test",
  "start": 1718150400000000,
  "end": "2024-06-12T00:20:00Z",
  "ueid": null,
  "copy": true,
  "plot_conf": {
    "trend_series": [
      "avg"
    ],
    "pages": [
      {
        "plots": [
          {
            "title": "OCI Pri Power (15A)",
            "mnemonics": [
              "PSE.OM1.OCI_PRI_CURR"
            ]
          },
          {
            "series": [
              {
                "mnemonic": "oci.dau.ddc.FPGA.CcdOpMode",
                "plot_options": {
                  "color": "k"
                }
              }
            ]
          }
        ]
      }
    ]
  },
  "data_conf": {
    "limit": false,
    "ids": "@#[42316,42318-42322,43255,44243,45140,45187-45210,45233-
45236,45238,45240,45241,45245,45247,45248,45259,45260,45265-
```

```
45268,45270,45272,45273,45277,45279,45280,45291,45292,45442,45444,45446,45448,45450,45452,45454,45460,45462,45472,45538,45540,45542,45544,45546,45548,45550,45556,45558,45568,51908,51910,64097,64109,64111,64113,64115,66333,66341,66349,66357,66365,66373,66381,67536,67537,69019,69024-69026,69031,69032,69729,77534,77570,77696]sci",
    "dis": false,
    "fill": false,
    "columns": {
      "ts_utc_iso": true
    },
    "join": true,
    "pkt": true
  }
}
```


This example performs an export for an existing Profile Definition. The `data_conf` and `plot_conf` objects will be retrieved from the `OPS_FLT_OCI_SDS` profile.


```
{
  "model": "oci.fm",
  "profile": "OPS_FLT_OCI_SDS",
  "label": "test",
  "start": 1718150400000000,
  "end": "2024-06-12T00:20:00Z",
  "ueid": null,
  "copy": true,
}
```

XINA Struct Trend

The XINA Struct Trend command line utility generates trend data products for a specified time range or Event Intervals. It is used by the XINA Trend Task. Generated data products include:

- Binned trend data
- Configurable trend plots
- Log file

Arguments

Name	Req	Description	Default
cache		Cache directory path. Used for any persistent caching across tasks for optimization purposes.	
conf	?	File path to the JSON config containing the trend parameters.	
env	?	File path to the JSON environment file which defines environment parameters such as project specific paths.	
help		Print available options.	
host		XINA Tunnel host.	`"localhost"`
port		XINA Tunnel port.	`41746`
import	?	Import directory path. Files that need to be imported will be placed here. `struct_trend` is responsbile for generating the corresponding import action file but `xina-run` will perform the actual importing.	
log	?	File path that log events will be written to.	
out	?	Ouptut directory path. Files placed here will be uploaded and accessible directly from the Task interface (e.g. notifications panel).	
post	?	Post directory path (for post-import outputs) e.g. uploading log files to the Task record.	
task	?	File path to the JSON file containing info about the Task.	
temp	?	Temp directory path for storing temporary files during execution.	

conf

The below table describes the available fields in the `conf` JSON file and is used to control how data should be trended.

Name	Type	Req	Description	Default
model	`utf8text`	?	Path of model to trend data from e.g. `moma.fm`	
copy	`boolean`		If true, the zip file will be uploaded and available from the Task interface.	false
trend	`utf8text`		The name of the Trend Definition to generate the trend for. If provided, then the `profiles`, `trend_conf`, and `plot_conf` will be loaded from it.	
label	`utf8text`		The text that will be used to name the files and final zip file. The format will be like `2024_06_12_00_00_00_2024_06_12_00_20_00_profile_label`	
profiles	`set(utf8string)`		Name of profile(s) to include in trend	
trend_conf	`struct_trend_conf`		See struct_trend_conf	
plot_conf	`jsonobject`		See trend plot conf	

Example conf:

This example performs a trend export with fields provided directly in the conf object.

```
{
  "label": "Test_Trend_From_Conf",
  "desc": "This is a test trend. The Trend parameters are provided in the conf.",
  "model": "model.path",
  "copy": true,
  "profiles": [
     "TEST_PROFILE"
  ],
  "trend_conf": {
     "t": [
        {
           "start": "2024-11-01T00:00:00Z",
           "end": "2024-11-01T00:10:00Z"
```

```
        }
      ],
      "bin_minutes": 1,
      "disable_filter": false,
      "intervals": null
    },
    "plot_conf": {}
}
```

In this example, the trend will be generated from an existing Trend Definition named `TEST_TREND`.

```
{
  "trend": "TEST_TREND",
  "model": "model.path",
  "copy": true
}
```

XINA Struct Report

The XINA Struct Report command line utility generates Export and Trend tasks for Profiles and Trend Definitions that are configured to auto generate. It can also optionally generate the XINA Report Email Task. The Export tasks will export the a days worth of data and the Trend tasks will use the time ranges or intervals as configured in the Trend Definition. By default, it will generate reports for the day before the task's creation time. The most common use case is to create the XINA Report Task (xina_struct_report) using XINA's internal cron system once per day.

Arguments

Name	Req	Description	Default
cache		Cache directory path. Used for any persistent caching across tasks for optimization purposes.	
conf	?	File path to the JSON config containing the report parameters.	
env	?	File path to the JSON environment file which defines environment parameters such as project specific paths.	
help		Print available options.	
host		XINA Tunnel host.	`"localhost"`
port		XINA Tunnel port.	`41746`
import	?	Import directory path. Files that need to be imported will be placed here.	
log	?	File path that log events will be written to.	
out	?	Ouptut directory path. Files placed here will be uploaded and accessible directly from the Task interface (e.g. notifications panel).	
post	?	Post directory path (for post-import outputs) e.g. uploading log files to the Task record.	
task	?	File path to the JSON file containing info about the Task.	
temp	?	Temp directory path for storing temporary files during execution.	

conf

The below table describes the available fields in the `conf` JSON file.

Name	Type	Req	Description	Default
model	utf8text	?	Path of model to generate Export and Trend tasks for e.g. "moma.fm"	
email	boolean		If true, the report email task will be generated	true
date	utf8text		Generates the report for the given date/day. Supports 2 formats: YYYY-MM-DD (e.g. 2020-10-25) and YYYY-DOY (e.g. 2025-300).	Generates report for the day before the task creation time.

Example conf:

```
{
  "model": "pace.ops_pace",
  "email": true,
  "date": "2024-01-30"
}
```

XINA Struct Report Email

The XINA Struct Report Email command line utility will find daily Export and Trend packages created for the given day, determine which users are registered to be emailed, and then create an email containing direct links to the Export and Trend packages. Note that this utility creates an email task for each user; the actual emailing will be handled by an AWS Lambda task.

Arguments

Name	Req	Description	Default
cache		Cache directory path. Used for any persistent caching across tasks for optimization purposes.	
conf	?	File path to the JSON config containing the report parameters.	
env	?	File path to the JSON environment file which defines environment parameters such as project specific paths.	
help		Print available options.	
host		XINA Tunnel host.	`"localhost"`
port		XINA Tunnel port.	`41746`
import	?	Import directory path. Files that need to be imported will be placed here.	
log	?	File path that log events will be written to.	
out	?	Ouptut directory path. Files placed here will be uploaded and accessible directly from the Task interface (e.g. notifications panel).	
post	?	Post directory path (for post-import outputs) e.g. uploading log files to the Task record.	
task	?	File path to the JSON file containing info about the Task.	
temp	?	Temp directory path for storing temporary files during execution.	

conf

The below table describes the available fields in the `conf` JSON file.

Name	Type	Req	Description	Default
model	`utf8text`	?	Path of model to generate emails for e.g. "moma.fm"	
hostname	`utf8text`	?	The hostname that will be used for creating the URLs to the Export and Trend packages in the email contents e.g. "moma.xina.io"	
date	`utf8text`		Generates the report for the given date/day. Supports 2 formats: YYYY-MM-DD (e.g. 2020-10-25) and YYYY-DOY (e.g. 2025-300).	Generates report for the day before the task creation time.

Example conf:

```
{
  "model": "pace.ops_pace",
  "hostname": "https://pace.xina.io",
  "date": "2024-01-30"
}
```