ADVANCED CAPABILITIES OF STAR ALGORITHM PROCESSING FRAMEWORK (SAPF) VERSION 2.0

Alexander Ken[2], Arthur Russakoff[2], Brian Helgans[3], Thomas King[2], Walter Wolf[1]

[1]NOAA/NESDIS/STAR, College Park, Maryland, 20740; [2]IMSG, Rockville, Maryland, 20852; [3]GAMA-1 Technologies, Greenbelt, MD, 20770

INTRODUCTION

- The STAR Algorithm Processing Framework (**SAPF**) software is developed by the **NOAA STAR** Algorithm Scientific Software Integration and System Transition Team (**ASSISTT**) to facilitate and support transition of satellite remote sensing algorithms from research to operations.
- The SAPF is a single software framework with common interfaces for running and managing the precedence of large sets of enterprise algorithms.
- The SAPF is currently run within NOAA Data Exploitation (**NDE**) to support the operational **JPSS S-NPP** and **NOAA-20 VIIRS** cloud, cryosphere, aerosol, and land products. In addition, it supports all **GOES-16** baseline and enterprise algorithms as well as production generation for foreign satellites.
- The ASSISTT Process Lifecycle and Algorithm Integration Team (**PLAIT**) integrates **Level 2 product algorithms** into the SAPF for both **testing** and **operational** implementation.
- **SAPF** version 1.0 is being replaced by **version 2.0** in early 2019. Version 2.0 contains significant updates to support greater resource efficiency, scalability, improved run time performance, simplified configuration for users, as well as additional features such as parallel processing.

SOFTWARE TO SUPPORT ASSISTT R2O

- Multi-mission and Enterprise Research-To-Operations (R2O) support requires software infrastructure to be **versatile**, **feature-rich**, and easily **extensible**.
- Intense schedules for algorithm integration and validation require SAPF to be a **time and cost efficient** software integration platform.
- A growing number of supported algorithms requires a **scalable** architecture.
- Closely supporting research with frequent algorithm updates and change requests requires the framework to be **agile** and **flexible**.
- When used in an operational environment such as NDE, SAPF needs to be **reliable**, **performance and resource efficient**.
- SAPF 2.0 is designed to answer the challenges of ASSISTT R2O.

INTEGRATION API

- The **fully-featured, native FORTRAN and C/C++** Application Programming Interface (**API**) allows for seamless and easy integration of algorithms and other general software components with SAPF in a **multi-language programming environment**.

CONFIGURATION SYSTEM

- The **XML based** configuration allows efficient and user-friendly construction of multi-attribute data keys and effectively parameterizes and **controls all data flows** in the system.
- Supports and largely automates construction of any **algorithm execution graphs**, allowing for **multi-satellite**, **multi-temporal**, and otherwise varying configurations.
- **Type-safe** design for use with type-safe programming languages.
- A nearly **layout-free** design allows integrators to construct algorithm configurations in a **flexible** and user-friendly way.
- Supports **XPath** at the configuration file and at the API levels. Any value added in configuration files can be directly accessed from algorithms via API.
- Supports XML **variable substitution** and access to **environment variables**, making the configuration **highly parameterizable**.
- **Scoped lookup** of variables and XML class elements achieves **condensed structured data storage** and very **high usability** of the configuration files.
- Advanced XML parser processing directives, such as **include**, **template**, **if**, and **for** are helping to make configuration **reusable**, **standardized**, and **compact**.

In general Run Configuration

```
<SATELLITE>NPP</SATELLITE>
<INSTRUMENT>VIIRS</INSTRUMENT>
...
```

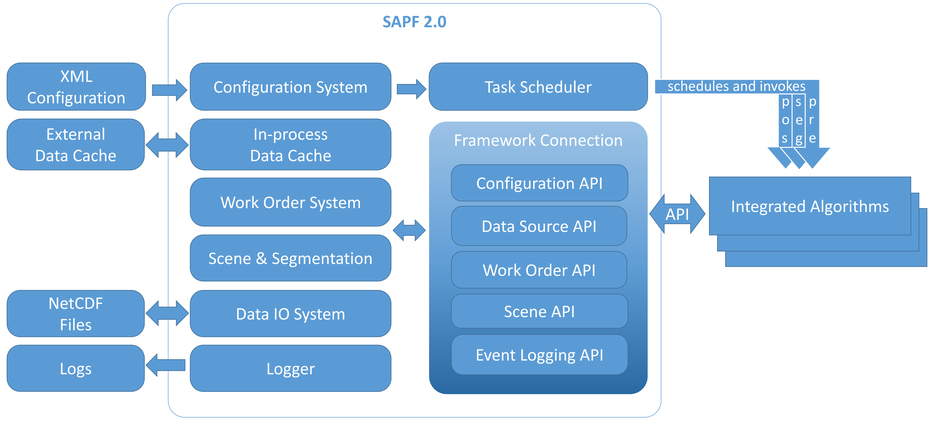
In Cloud Mask Configuration

```
<!-- include file for instrument-dependent variables -->
<__include>
    Config/Algorithms/CLOUD_MASK_EN/${INSTRUMENT}-CLOUD_MASK_EN.xml
</__include>
```

Include

Variable

SAPF 2.0 SYSTEM COMPONENTS



DATA ARCHITECTURE

- Data Cache operates with **multi-attribute key-value records**

- Currently used attributes: *Satellite, Instrument, Resolution, Channel, (data) Name, Segment, Time Offset, Component (algorithm), Task*.
- The data architecture is highly **scalable and flexible**. It is universal with regard to ASSISTT requirements.
- **2-tier Data Cache** implementation *efficiently* supports the **in-process and multi-processor data flows**.
- Data Cache is designed for **easy integration** with various **COTS data bases** and **data cache** technologies.
- Data entries can be presented in a form of **JSON** or **XML** headers and standardized value types and designed for an **easy data exchange with other information systems** and for use in **service-oriented architectures** (Microservices, SOA).

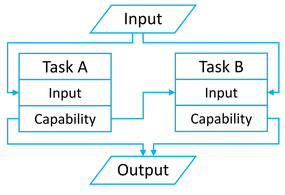
```
{
    "record": {
        "id": {
            "component": "SAT_VIIRS",
            "satellite": "NPP",
            "instrument": "VIIRS",
            "name": "WAVENUMBER"
        },
        "data": {
            "type": "float-array",
            "shape": "21",
            "file": "binary_array_file_path.nc"
        }
    }
}
```

Key

Value

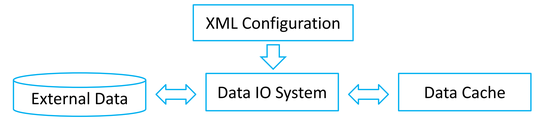
TASK SCHEDULING AND EXECUTION

- **Tasks** are **algorithm configuration instances** which are scheduled and executed via SAPF.
- **Task scheduler** builds an **execution graph** by analyzing and connecting **inputs** and **capabilities** of **configured tasks**, while utilizing the **pull strategy**.
- **System inputs** and **outputs** are also used as graph nodes.
- For example, an algorithm can be executed multiple times with different time-dependent parameters and inputs in a single temporal configuration.
- Data between Tasks is passed via Data Cache.



DATA IO SYSTEM

- **Transfers data** between Data Cache and external data sources
  - **Automatic**: no programming required.
  - **Flexible**: configured as part of the overall SAPF XML configuration.
- Currently implemented: **NetCDF** input and output
  - Transfers data between NetCDF files and the SAPF 2.0 data cache.
  - Configuration uses **data bindings**.
  - Supports NETCDF 3 and 4 standards (including groups and arrays of strings).
  - Can be used for 3-rd party data import, as a file transform tool, and for Data Cache data exporting.
  - Writes outputs in **parallel** when SAPF is used in parallel processing mode.
- Adding other data storage facilities and formats is being considered: HDF, BUFR.



WORK ORDER SYSTEM

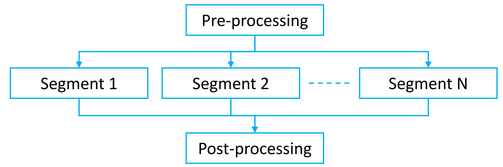
- Allows for **selective, resource-efficient computation**.
- Informs algorithms via API about requested outputs.
- Work Orders are **automatically generated** based on the configured algorithm inputs and capabilities, as well as on the Data IO System data bindings.
- Only tasks that have work orders are scheduled for execution.

SCENE AND DATA SEGMENTATION SYSTEM

- Supports **padded**, **multi-dimensional**, and **multi-resolution** data input and segmentation.
- Currently used dimensions: COLUMN, ROW, PLANE, TIME.
- API utilizes **dimension superposition** approach.
- Allows to choose input ROI and desired numbers of segments.
- Padding can be **automatically calculated** based on configured algorithm needs.

PARALLEL PROCESSING

- Allows to process data segments in parallel using **multiple processes** on the same computer or using **multiple compute nodes** in a cluster.
- Processes are dispatched in the following order: **pre-processing**, **parallel segment processing**, **post-processing**.



EVENT LOGGING

- Seven-level, **customizable** per algorithm and per SAPF subsystem logging is developed to be a **powerful troubleshooting tool**.
- **Efficient** and **convenient** API utilizes delayed message construction and conditional compilation.

FUTURE PLANS

- Support of an **Enterprise Data Management** infrastructure for Satellite and Environmental data.
- Integration with **Microservices** (**SOA**) based architecture, utilizing Docker and Kubernetes.