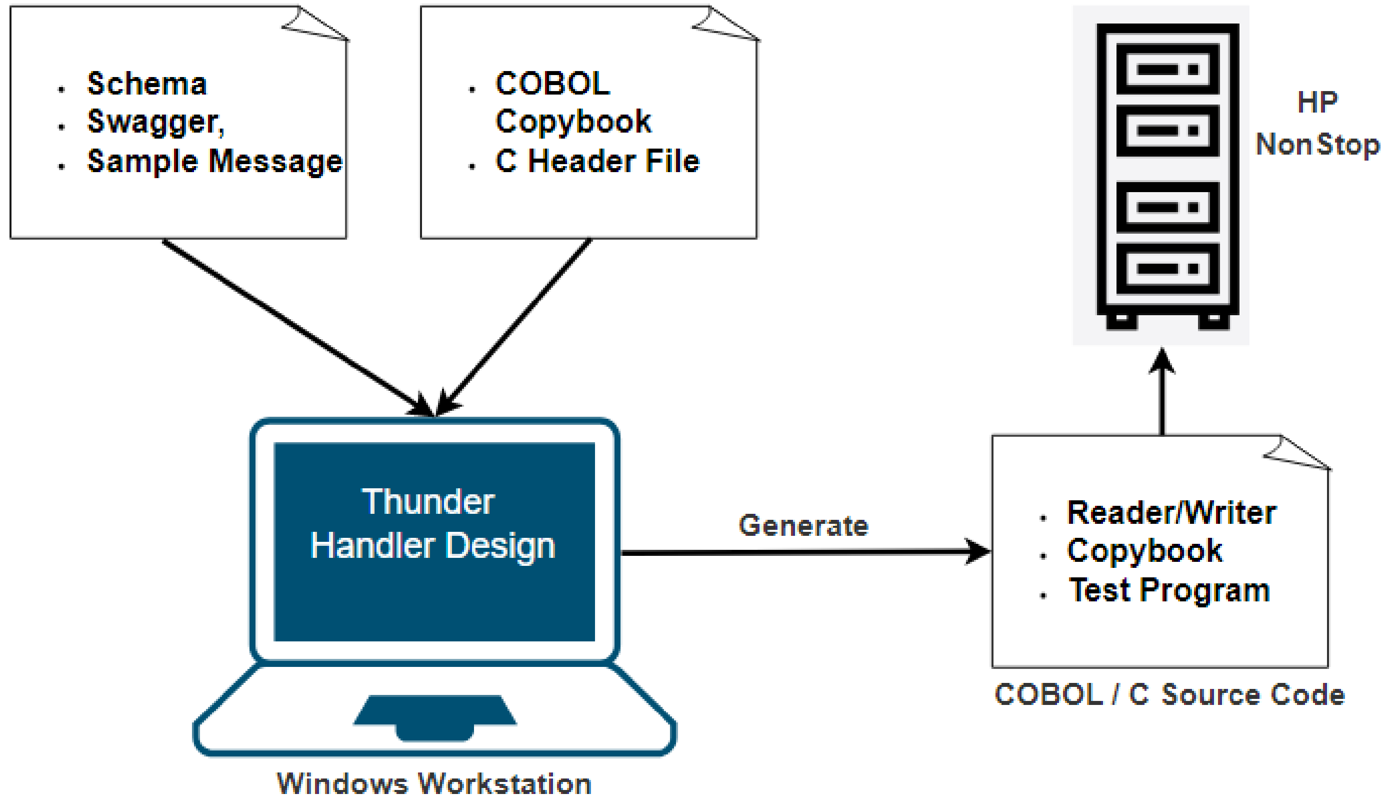
TIC TICSOFTWARE.COM

Canam Software

JSON Thunder Product Overview

Using the Thunder Toolset

Developer Workflow



JSON Thunder enables COBOL programs to read and write JSON documents so they can:

- exchange data in JSON format with other applications and companies.
- use JSON messages to call web services or interface with third-party packages and non-COBOL applications.
- create JSON APIs for COBOL programs - enabling them to provide web services or be called by non-COBOL applications.

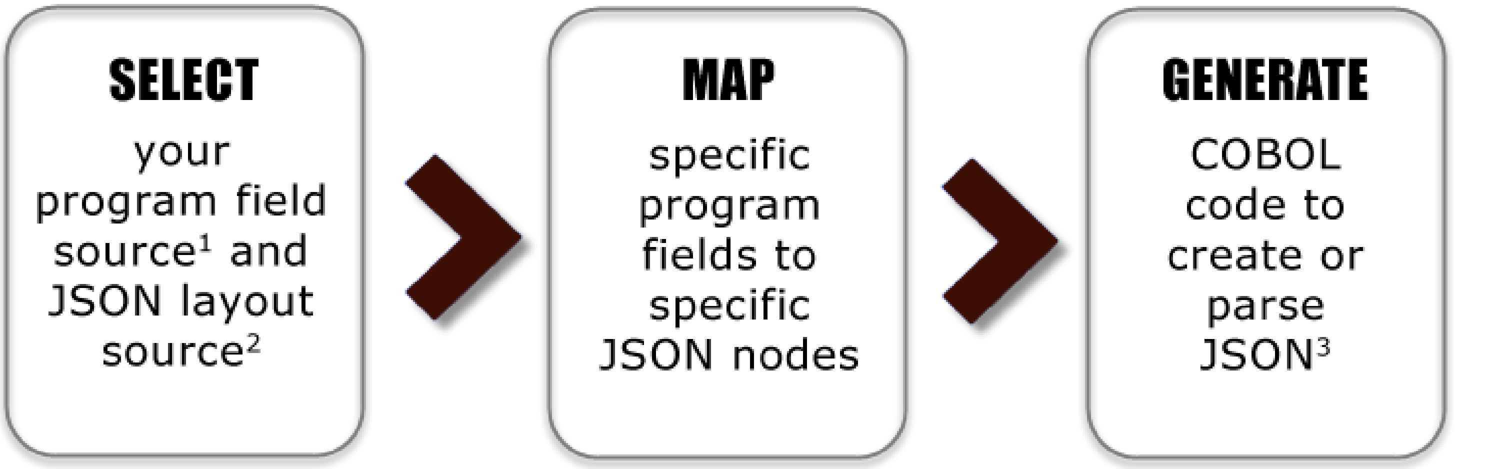
Read and Write JSON Messages in COBOL

JSON Thunder generates all the COBOL source code needed to:

- read JSON messages into COBOL structures.
- create JSON messages from COBOL structures.
- validate JSON messages to ensure only well-formed and valid documents are processed.

The code is generated as a COBOL sub-program that can be called from existing COBOL programs.A sample calling program is provided to assist with developing new programs.

How it Works



SELECT your program field source[1] and JSON layout source[2]

MAP specific program fields to specific JSON nodes

GENERATE COBOL code to create or parse JSON[3]

[1] Program field source can be COBOL copybook or derived from the JSON layout.

[2] JSON source can be JSON Schema, sample JSON message or derived from program data fields.

[3] The final output is a customized, callable program that can be used in existing applications. If you don't have an existing application, Thunder also provides a sample main (or calling) program that can serve as the starting point to developing it.

Model-Driven Development

JSON Thunder uses a model-driven development approach enabling a faster time to market and the ability to respond to changes quickly.The Thunder toolset provides:

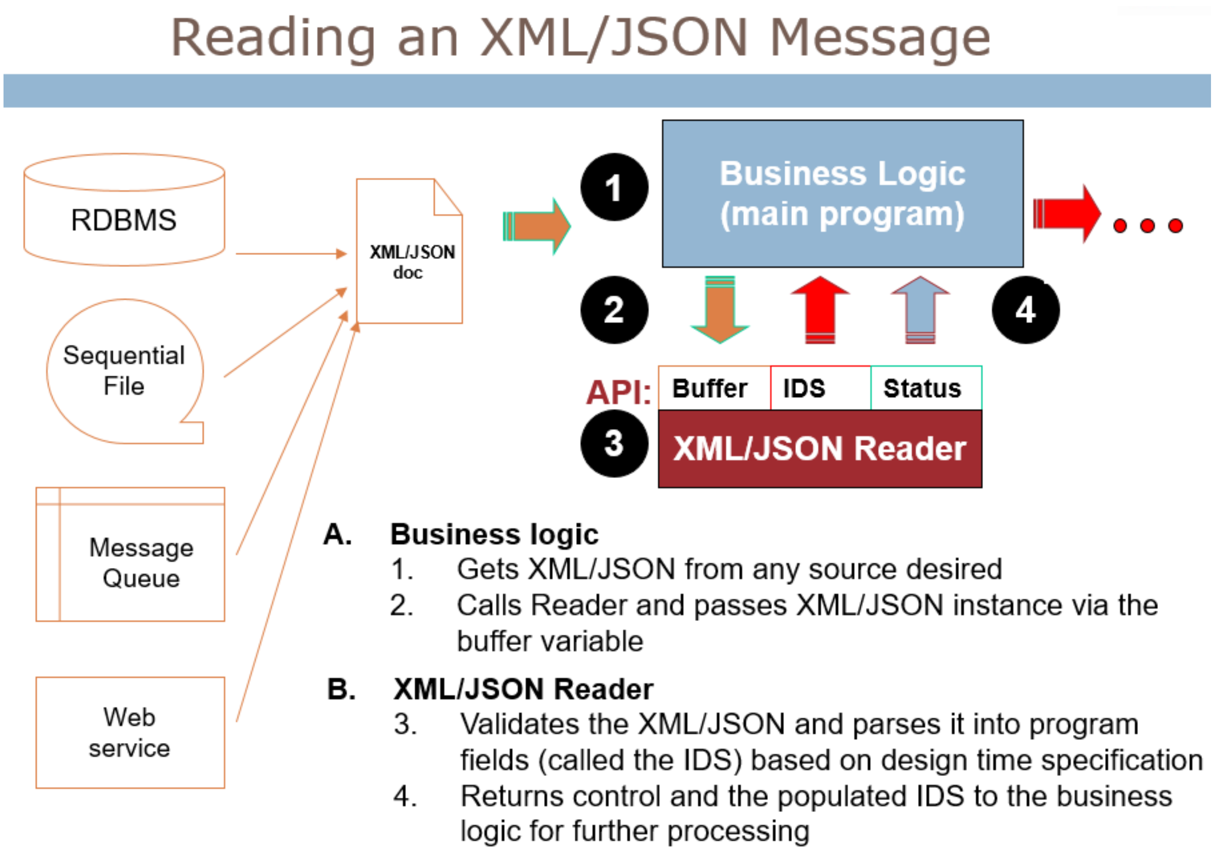
- A Visual Designer showing the program field layout, the JSON message layout and the mapping between the two.
- A full life cycle solution that is used in both development and maintenance.
- Automated source code generation.

Features

- High performance (not a generic parser!).
- Addresses the complexities of JSON (e.g. anyOf, oneOf, allOf, etc.).
- Model-driven approach reducing development time and enabling developers to respond to changes quickly.
- Create Handler Designs from JSON schemas, Swagger definitions, sample JSON messages and COBOL/C data structures.
- Architecture and platform independent solution.
- Easy integration with existing programs

How the Generated Code is Used at Runtime.

Reading an XML Message

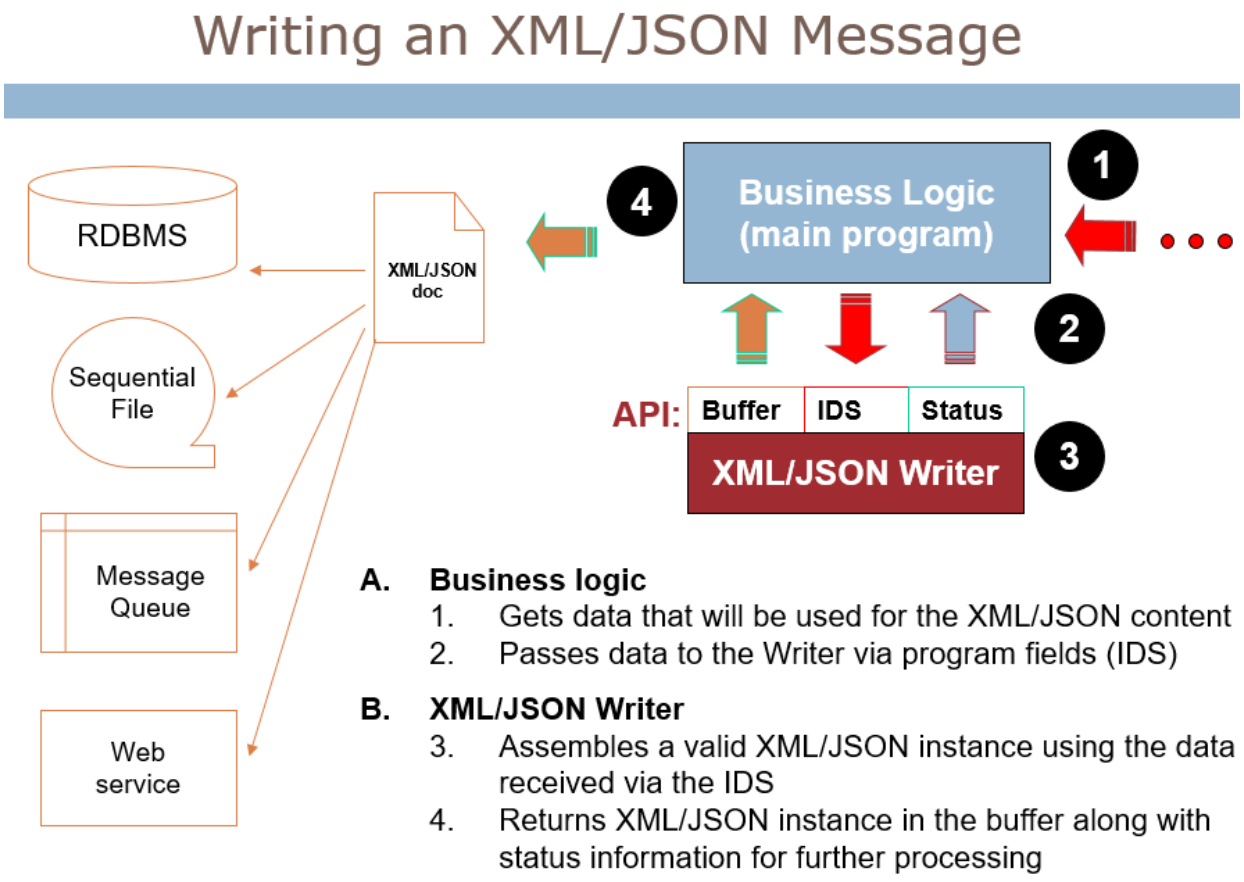


Reading an XML/JSON Message

A. **Business logic**
1. Gets XML/JSON from any source desired
2. Calls Reader and passes XML/JSON instance via the buffer variable

B. **XML/JSON Reader**
3. Validates the XML/JSON and parses it into program fields (called the IDS) based on design time specification
4. Returns control and the populated IDS to the business logic for further processing

Writing an XML Message



Writing an XML/JSON Message

A. **Business logic**
1. Gets data that will be used for the XML/JSON content
2. Passes data to the Writer via program fields (IDS)

B. **XML/JSON Writer**
3. Assembles a valid XML/JSON instance using the data received via the IDS
4. Returns XML/JSON instance in the buffer along with status information for further processing