

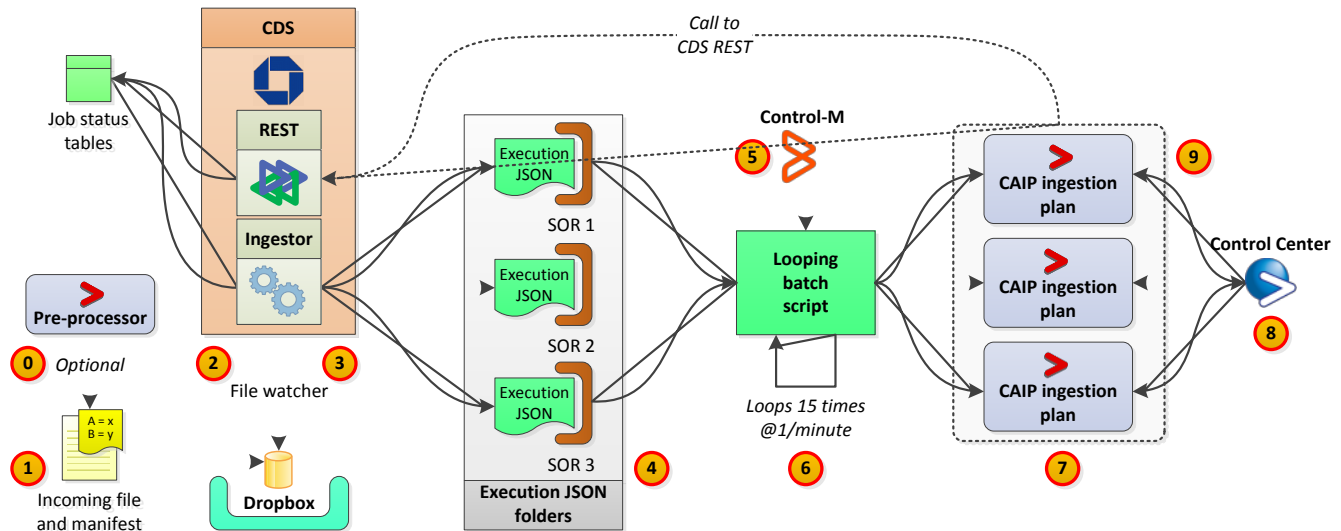
## 1. CAIP Ingestion Overview

The CAIP ingestion workflow provides the means to ingest CDS entities in CAIP. Once the entities have been ingested in CAIP, they can be processed using Ab Initio tools and written to the HDFS conformed zone.

The CDS entities can be ingested from SOR projects from upstream systems and outside vendors. SOR projects comprise the following major components:

- **Input file** — The input data file.
- **Manifest file** — Each input data file must have a corresponding manifest file. A manifest is a standardized control file that conveys transmission completion in the CDS ecosystem, and enables CDS to perform validations and reconciliations on incoming files. For more information, see
- **DML record format** — Each input data file must also have a corresponding DML record format. This record format is an integral part of the CAIP metadata exchange workflow, which is described [TBD].
- **Custom preprocessor graphs** — Custom LOB graphs to perform any required preprocessing on the data prior to ingestion; for example, flattening complex files and performing data quality checks.
- **CAIP execution graph** — Generic graph provided by the CAIP framework to perform encryption, compression, and to write files to the conformed zone.

The following illustration provides an overview of the CAIP ingestion workflow:



In the preceding illustration:

Step	Description
0	One or more optional preprocessing graphs to prepare an input complex raw file for ingestion. Preprocessing is required for incoming files that arrive without a manifest, or which require preprocessing before they are ready for ingestion into the conformed zone.
1	An incoming SOR file and its associated manifest file are uploaded by means of SFTP to the CDS dropbox.
2	The CDS manifest file watcher detects the incoming manifest file, triggers the CDS ingestion process, and updates the relevant job status logging tables.

3	<p>The CDS ingestor writes execution JSON files to SOR-specific JSON folders, as follows:</p> <p><b><code>/apps/cds/abi_meta_ops/execution/&lt;SOR&gt;</code></b></p> <p>Where &lt;SOR&gt; is an SOR-specific folder.</p>
4	<p>The execution JSON folders are read by a looping batch script.</p>
5	<p>The looping batch script is triggered by Control-M. For each loop, the script reads a JSON execution folder, passes the JSON values as input parameters to a corresponding CAIP ingestion plan pset, and then goes to sleep. The command executed by the batch script is as follows:</p> <p><b><code>air sandbox run /Data/abinitio/sandboxes/CDS/sor/caip_&lt;SOR&gt;_pvt/pset/caip_execution_plan.&lt;SOR&gt;.pset -JSON_FILE &lt;execution json&gt;</code></b></p> <p>Note that the script does not check the validity of the JSON file, and does not fail even if the CAIP ingestion plan fails. If the batch script fails for some reason, the error is returned to Control-M.</p>
6	<p>For each Control-M job, the batch script loops fifteen times, once per minute, and then quits.</p>
7	<p>The SOR-specific CAIP ingestion plan psets generate corresponding instances of <code>caip_execution_plan.plan</code>.</p>
8	<p>Control Center is used to monitor runs of the CAIP ingestion plan.</p>
9	<p>The CAIP ingestion plan ingests the given entity into CAIP, and then updates the CDS job status tables by means of CDS REST calls.</p>