# Generate UUID

**Description**
This app generates a random RFC 4122 version 4 UUID and adds it to the incoming job as a private data variable. The UUID is generated using node.js' crypto package which includes a cryptographic pseudorandom number generator.
RFC 4122: https://datatracker.ietf.org/doc/html/rfc4122

Typical output of different UUIDs:
```
6948DF80-14BD-4E04-8842-7668D9C001F5
4B8302DA-21AD-401F-AF45-1DFD956B80B5
8628FE7C-A4E9-4056-91BD-FD6AA7817E39
10929DF8-15C5-472B-9398-7158AB89A0A6
ED280816-E404-444A-A2D9-FFD2D171F928
D952EB9F-7AD2-4B1B-B3CE-386735205990
3F897E85-62CE-4B2C-A957-FCF0CCE649FD
8E7C2F0A-6BB8-485C-917E-6B605A0DDF29
1AD2F3EF-87C8-46B4-BD1D-94C174C278EE
AA97B177-9383-4934-8543-0F91A7A02836
```

**Example Use Cases**
1. **Version Control**
   - Generate a new UUID for each version of a file as it passes through different processing stages. This ensures that intermediate versions can be uniquely identified and accessed if needed.
   - Example: Differentiate between the original file and its processed variants, such as one with color reduction and another with resolution changes.
2. **Database Integration**
   - Use the generated UUID as a primary key or reference ID for storing metadata about the file in a database.
   - Example: Save file processing logs, timestamps, and statuses in a database and use the UUID to retrieve this data later.
3. **Cross-System Synchronization**
   - When transferring files between different systems, include the UUID as a consistent identifier to avoid name conflicts or processing errors.
   - Example: Sync files between a local hotfolder system and a cloud-based processing service, ensuring the same UUID is used in both environments.

**Compatibility**
Switch Fall 2023 or later.

**Connections**
The app has an incoming connection and a single output connection. In the event of an error, the job fails.

**Flow elements properties**
- **Private Data Key:** Set the private data key for storing the UUID.