# developer.skatelescope.org Documentation

***Release 0.12.3***

**Marco Bartolini**

**Mar 13, 2024**

# CONTENTS

The processing controller (PC) is the SDP service responsible for the controlling the execution of processing blocks (PBs).

CHAPTER

ONE

# PROCESSING CONTROLLER

The processing controller (PC) is the SDP service responsible for controlling the execution of processing blocks (PBs).

Each execution block (EB) that SDP is configured to execute contains a number of PBs, either real-time or batch. The real-time PBs run simultaneously for the duration of the EB. Batch PBs run after the EB is finished, and they may have dependencies on other PBs, both real-time and batch.

The SDP architecture requires the PC to use a model of the available resources to determine if a PB can be executed. This has not been implemented yet, so real-time PBs are always executed immediately, and batch processing ones when their dependencies are finished.

## 1.1 Processing block and its state

A PB and its state are located at the following paths in the configuration database:

```
/pb/[pb_id]
/pb/[pb_id]/state
```

The PB is created by the subarray Tango device when starting an EB. Once it is created it does not change. The state is created by the PC when deploying the processing script, and it is subsequently updated by the PC and the script.

The entries in the PB state relevant to the PC are `status` and `resources_available`, for example:

```
{
    "status": "WAITING",
    "resources_available": false
}
```

`status` is a string indicating the status of the script. Possible values are:

- `STARTING`: set by the PC when it deploys the script, hereafter the script is responsible for setting `status`
- `WAITING`: script has started, but is waiting for resources to be available to execute its processing
- `RUNNING`: script is executing its processing
- `FINISHED`: script has finished its processing
- `FAILED`: set by the PC if it fails to deploy the script, or by the script in the case of a non-recoverable error

`resources_available` is a boolean set by the PC to inform the script whether it has the resources available to start its processing. Although the resource model is not implemented yet, this is used to control when batch PBs with dependencies start.

## 1.2 Behaviour

The behaviour of the PC is summarised as follows:

1. If a PB is new, the PC will create the processing script deployment for it. A PB is deemed to be new if the PB state does not exist. The PC reads the script definition from the configuration DB to discover which OCI container image to deploy. It creates the state and sets `status` to `STARTING` and `resources_available` to `false`. If the script definition is not found in the configuration DB, the PC still creates the state, but sets `status` to `FAILED`.
2. If a PB's dependencies are all `FINISHED`, the PC sets `resources_available` to `true` to allow it to start executing. Real-time PBs do not have dependencies, so they start executing immediately.
3. The PC removes processing deployments (scripts and execution engines) not associated with any existing PB. This is used to clean up if a PB is deleted from the configuration DB.
4. Clean up the Configuration Database:
    - Delete PBs that are `FINISHED` and do not have any associated execution blocks. If there is an associated EB, check if that exists, and if it doesn't, delete the PB.
    - Delete EBs without any associated PBs. If there are associated PBs, check if they exist, and if they don't, delete the EB.
    - Iterate through EBs with PBs. If the EB status is `FINISHED`, Check if all of the associated PBs are also `FINISHED` and have been in the database in a finished state for at least an hour. If these conditions apply, delete all of the PBs and the EB. If there is at least one PB that cannot be deleted, do not delete any of the other PBs or the EB, and wait until all can be deleted together.

## 1.3 Implementation

The above explained behaviour of the PC is implemented using the Configuration Library's Config().watcher() method. For more information on watchers take a look at the Watchers section of the Configuration Library documentation.

CHAPTER

TWO

# INDICES AND TABLES

- genindex
- modindex