

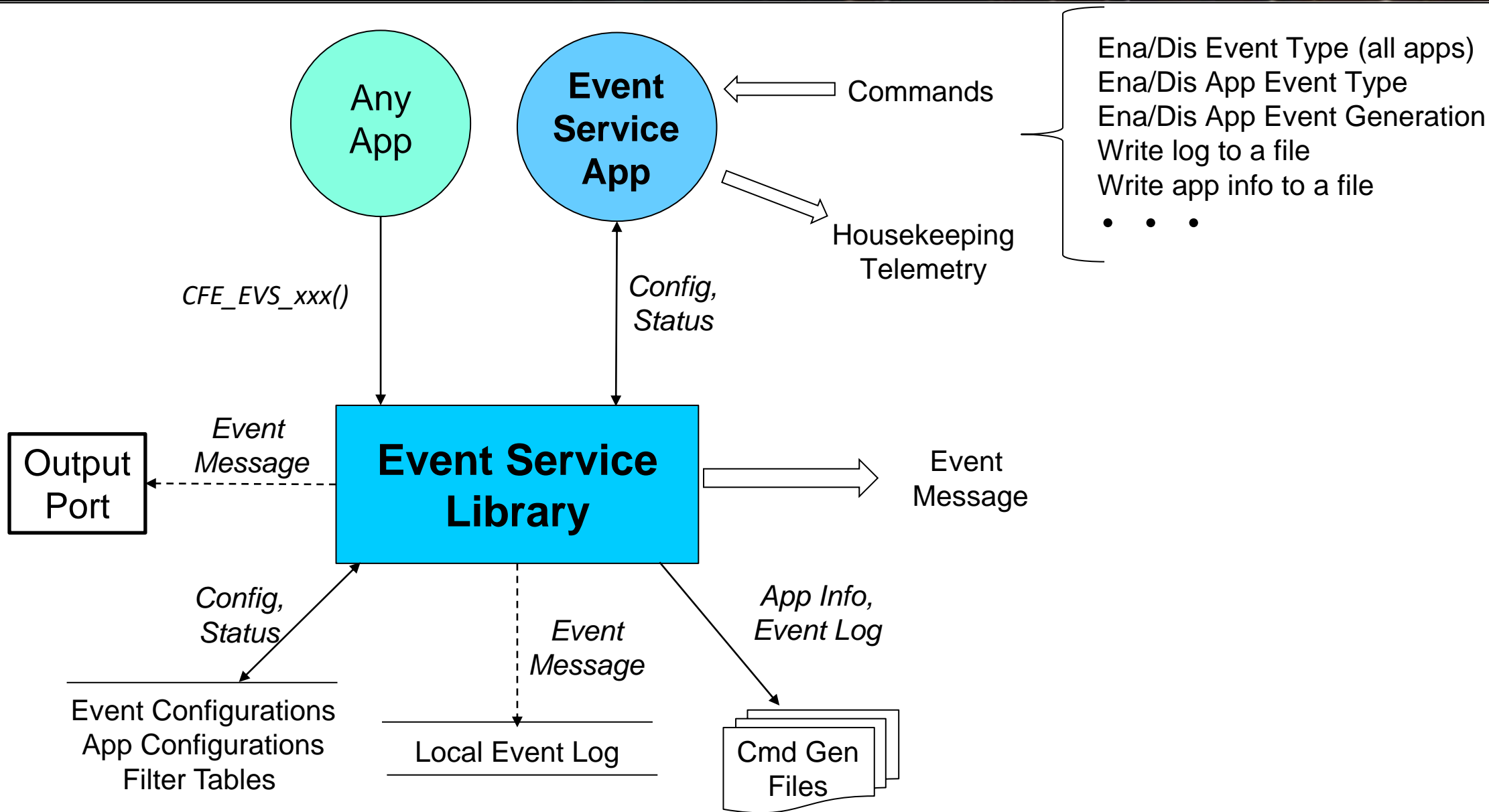


cFE Event Service (EVS)

OSK cFS Training
6/11/20

- **Provides an interface for sending time-stamped text messages on the software bus**
 - Considered asynchronous because they are not part of telemetry periodically generated by an application
 - Processor unique identifier
 - Optionally logged to a local event log
 - Optionally output to a hardware port
- **Four event types defined**
 - Debug, Informational, Error, Critical
- **Event message control**
 - Apps can filter individual messages based on identifier
 - Enable/disable event types at the processor and application scope

Event Service Context



Event message example:

14:14:40.500 ERROR CPU=CPU3 APPNAME=CFE_TBL EVENT ID=57 Unable to locate 'TST_TBL.invalid_tbl_02' in Table Registry

Time Event Type processor Application Event ID Event text



- **Spacecraft time via CFE_TIME_GetTime()**
- **Spacecraft ID (not shown) defined in cfe_mission_cfg.h**
- **Processor ID defined in cfe_platform_cfg.h**
- **Event ID is unique within an application**
- **Event Text is created using printf() format options**
- **“Short Format” platform option allows messages to be sent without text portion**

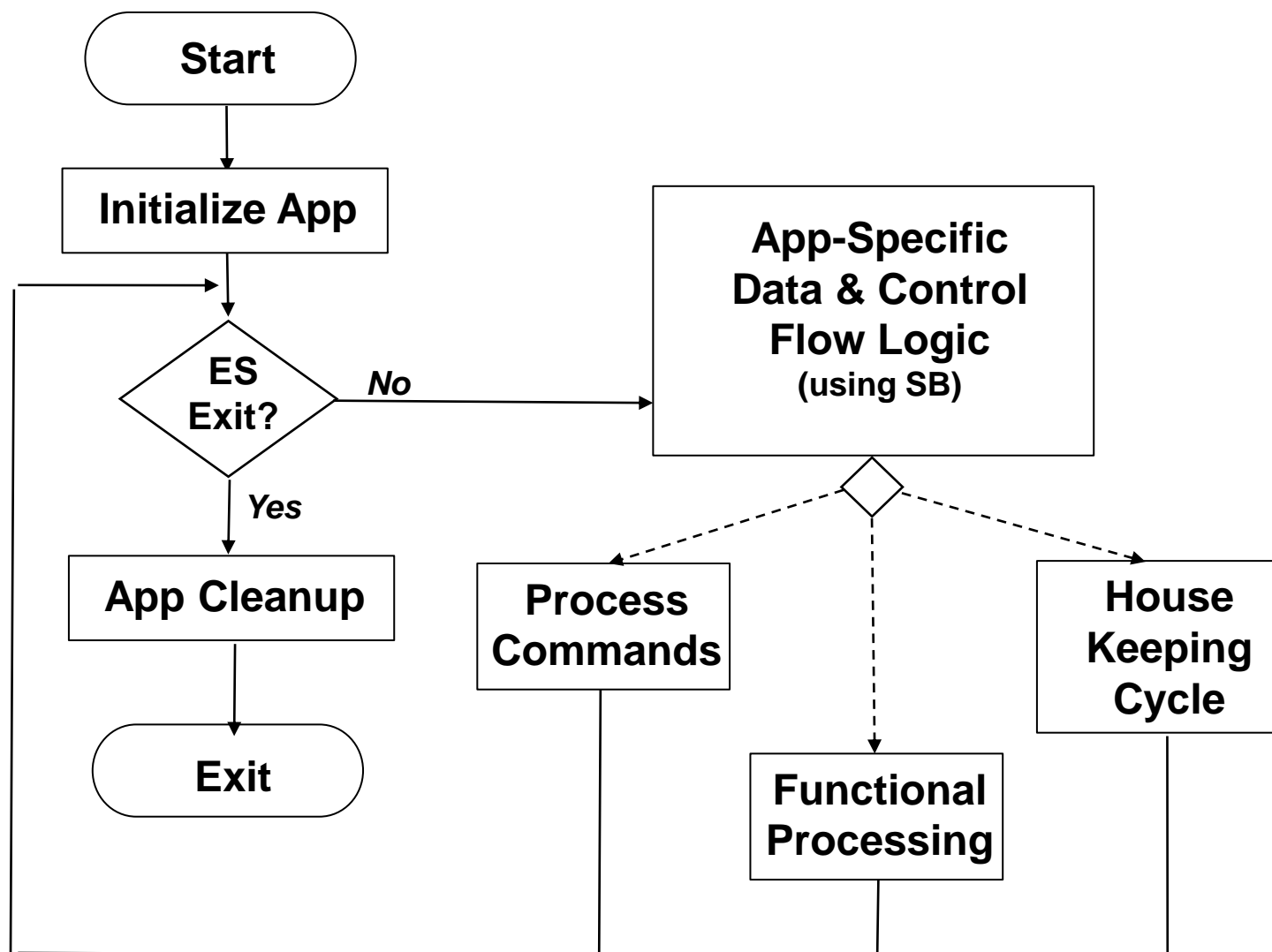
- **“Filter Mask”**
 - Bit-wise Boolean AND performed on event ID message counter, if result is zero then the event is sent
 - Mask applied before the sent counter is incremented
 - 0x0000 => Every message sent
 - 0x0003 => Every 4th message sent
 - 0xFFFE => Only first two messages sent
- **Reset filter**
 - Filters can be reset from an application or by command
- **Event filtering example**
 - Software Bus ‘No Subscriber’ event message, Event ID 14
 - See *cfe_platform_cfg.h* CFE_SB_FILTERED_EVENT1
 - Default configuration is to only send the first 4 events
 - Filter Mask = 0xFFFC
- **CFE_EVS_MAX_FILTER_COUNT (cfe_evs_task.h) defines maximum count for a filtered event ID**
 - Once reached event becomes locked
 - Prevents erratic filtering behavior with counter rollover
 - Ground can unlock filter by resetting or deleting the filter



Event Message Control



- **Processor scope**
 - Enable/disable event messages based on type
 - Debug, Information, Error, Critical
- **Application scope**
 - Enable/disable all events
 - Enable/disable based on type
- **Event message scope**
 - During initialization apps can register events for filtering for up to CFE_EVS_MAX_EVENT_FILTERS defined in *cfe_platform_cfg.h*
 - Ops can add/remove events from an app's filter



- **Initialize App**

- CFE_EVS_Register()
 - Optional filter parameter

- **Command/Functional Processing**

- CFE_EVS_SendEvent() as needed

- **Housekeeping Cycle**

- CFE_EVS_ResetFilter()
 - Rare and typically use in interface apps where the event situation may be corrected

- **App Cleanup**

- CFE_EVS_Unregister()
 - Recommended but ES does it for an app, so not mandatory

Application Functions	Purpose
CFE_EVS_Register	Register the application with event services. All Applications must register with EVS
CFE_EVS_Unregister	Cleanup internal structures used by the event manager
CFE_EVS_SendEvent	Request to generate a software event. Event message will be generated based on filter settings
CFE_EVS_SendEventWithAppID	Generate a software event as though it came from the specified cFE Application
CFE_EVS_SendTimedEvent	Generate a software event with a specific time tag
CFE_EVS_ResetFilter	Resets the calling application's event filter for a single event ID
CFE_EVS_ResetAllFilters	Resets all of the calling application's event filters

- **Applications should register with EVS immediately after ES app registration to allows events to be used rather than syslog writes for noteworthy events**
- **Local event log**
 - Suitable for multi-processor architectures
 - Serves as backup to onboard-recorder during initialization or error scenarios
 - Preserved across a processor reset
- **Event message guidelines**
 - Don't desensitize operators with too many events
 - Be judicious and consistent with when informational events are used
 - Consider whether routine telemetry is a better option for certain state knowledge
 - Balance testing and operational needs
 - Is the event a convenience for testing or does it help operations?
 - Be cognizant that other apps can monitor events and take corrective action based on events



Reset Behavior



- **Power-on Reset**
 - No data preserved
 - Application initialization routines register with the service
 - If configured local event log enabled
- **Processor Reset**
 - If configured with an event log, preserves
 - Messages
 - Mode: Discard or Overwrite
 - Log Full and Overflow status



Retrieving Onboard State



- **Housekeeping Telemetry**
 - Log Enabled, Overflow, Full, Enabled
 - For each App: AppID, Events Sent Count, Enabled
- **Write application data to file. For each app**
 - Active flag – Are events enabled
 - Event Count
 - For each filtered event
 - Event ID
 - Filter Mask
 - Event Count – Number of times Event ID has been issued
- **Local event log**
 - If enabled events are written to a local buffer
 - Log “mode” can be set to over write or discard
 - Serves as backup to onboard-recorder during initialization or error scenarios
 - Suitable for multi-processor architectures
 - Command to write log to file

Parameter	Purpose	Scope	Notes
CFE_EVS_LOG_ON	cFE core platform message IDs	Platform	Defines the message IDs the cFE core will use on that Platform(CPU)
CFE_EVS_LOG_MAX	OSAL platform configuration	Platform	
CFE_EVS_DEFAULT_LOG_MODE	cFE core platform configuration	Platform	Most cFE parameters are here
XX_platform_cfg.h	Application platform wide configuration	Platform	
CFE_EVS_DEFAULT_TYPE_FLAG	Application message IDs	Platform	
CFE_EVS_MAX_EVENT_FILTERS	Define Maximum Number of Event Filters per Application	Platform	