
cFE
Software Bus
(SB)
OSK cFS Training

Software Bus (SB) Services - Overview

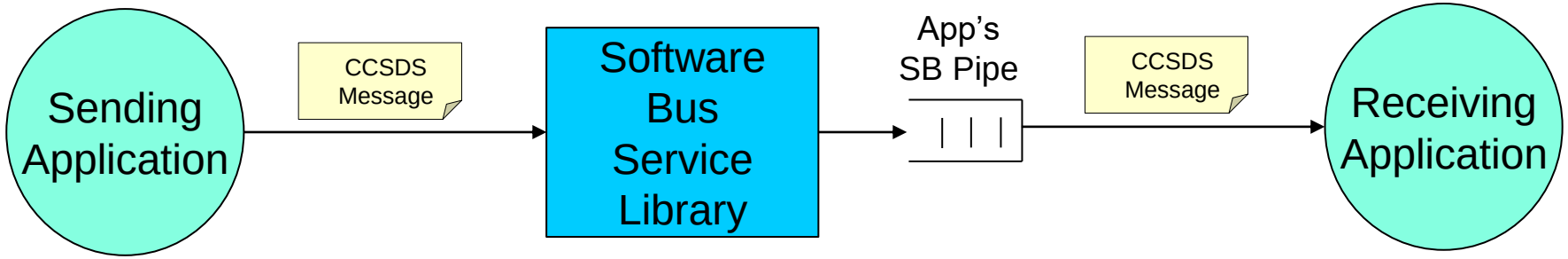
- **Provides an inter-application message service using a publish/subscribe model**

- **Routes messages to all applications that have subscribed to the message (i.e. broadcast model)**
  - Subscriptions are done at application startup
  - Message routing can be added/removed at runtime
  - Sender does not know who subscribes (i.e. connectionless)

- **Reports errors detected during the transferring of messages**

- **Outputs Statistics Packet and the Routing Information when commanded**

Software Bus Application Communication

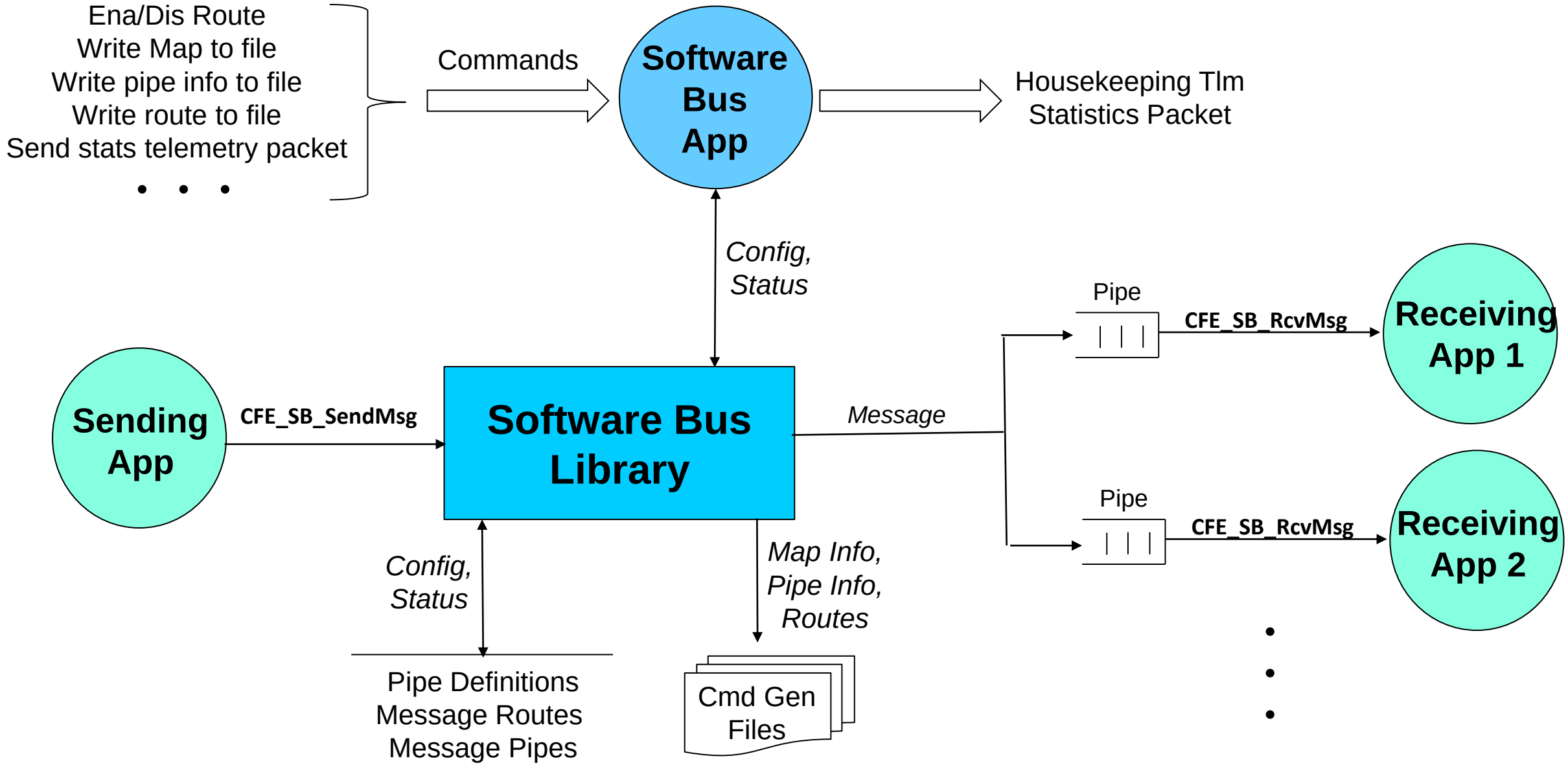


- **Messages defined using the Consultative Committee for Space Data Systems (CCSDS) packet standard**

- **Applications create *SB Pipe (a FIFO queue)* and subscribe to receive messages**
  - Typically performed during application initialization
  - However apps can subscribe and unsubscribe to messages at any time

- ***SB Pipes* used for application data and control flow**
  - Poll and pend for messages

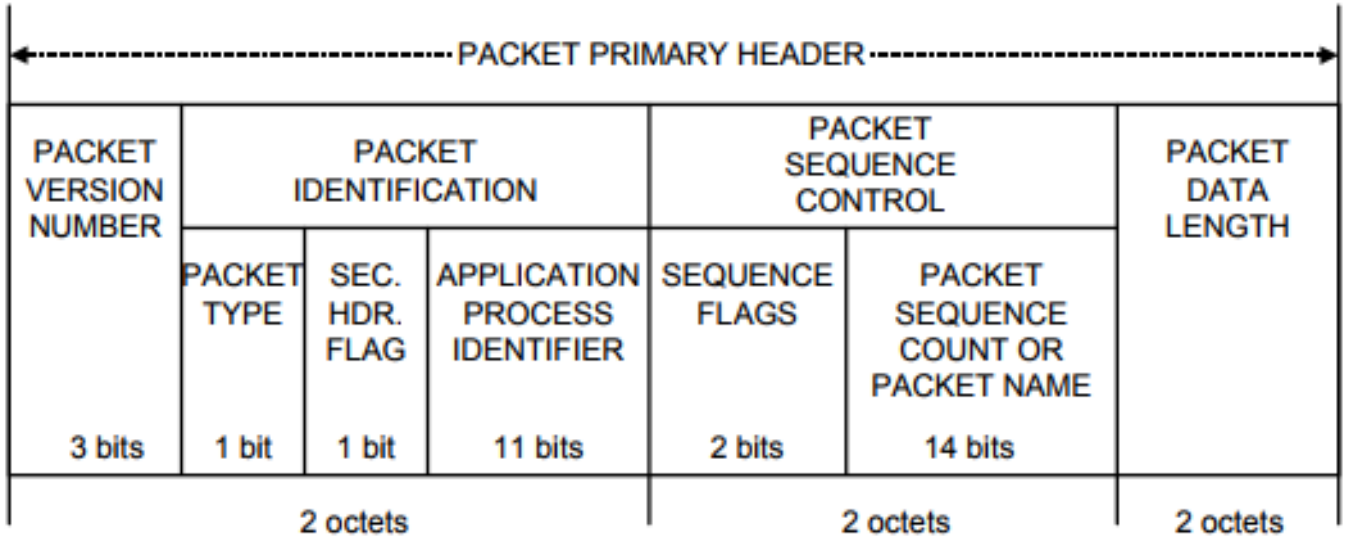
Software Bus Context



SB Messages (1 of 3)

- **By default messages conform to the Consultative Committee for Space Data Systems (CCSDS) space packet standard**
  - In theory other formats could be used but that has not occurred in practice
  - Simplifies data management since CCSDS standards used for flight-ground interfaces

- **CCSDS Packet Primary Header (Always big endian)**



- **Packet Type**
  - 0: Telemetry
  - 1: Command
- **Secondary Header Flag**
  - 1: Secondary Header Present
- **CFE_MISSION_ES_CMD_MSG = 0x1806**
  - Cmd packet with secondary header
  - AppId = 6
- **CFE_MISSION_ES_HK_TLM_MSG = 0x0800**
  - Tlm packet with secondary header
  - AppId = 0

SB Messages (2 of 3)

- **“Packet” often used instead of “message” but not quite synonymous**
  - “Message ID” (first 16-bits) used to uniquely identify a message
  - “App ID” (11-bit) CCSDS packet identifier

- **CCSDS Command Packets**
  - Secondary packet header contains a command function code and checksum
  - cFS apps define a single ground command message and use the function code to identify each command
  - Commands can originate from the ground or from onboard applications

- **CCSDS Telemetry Packets**
  - Secondary packet header contains a time stamp of when the data was produced
  - Telemetry is sent on the software bus by apps and can be ingested by other apps, stored onboard and sent to the ground

SB Messages (3 of 3)

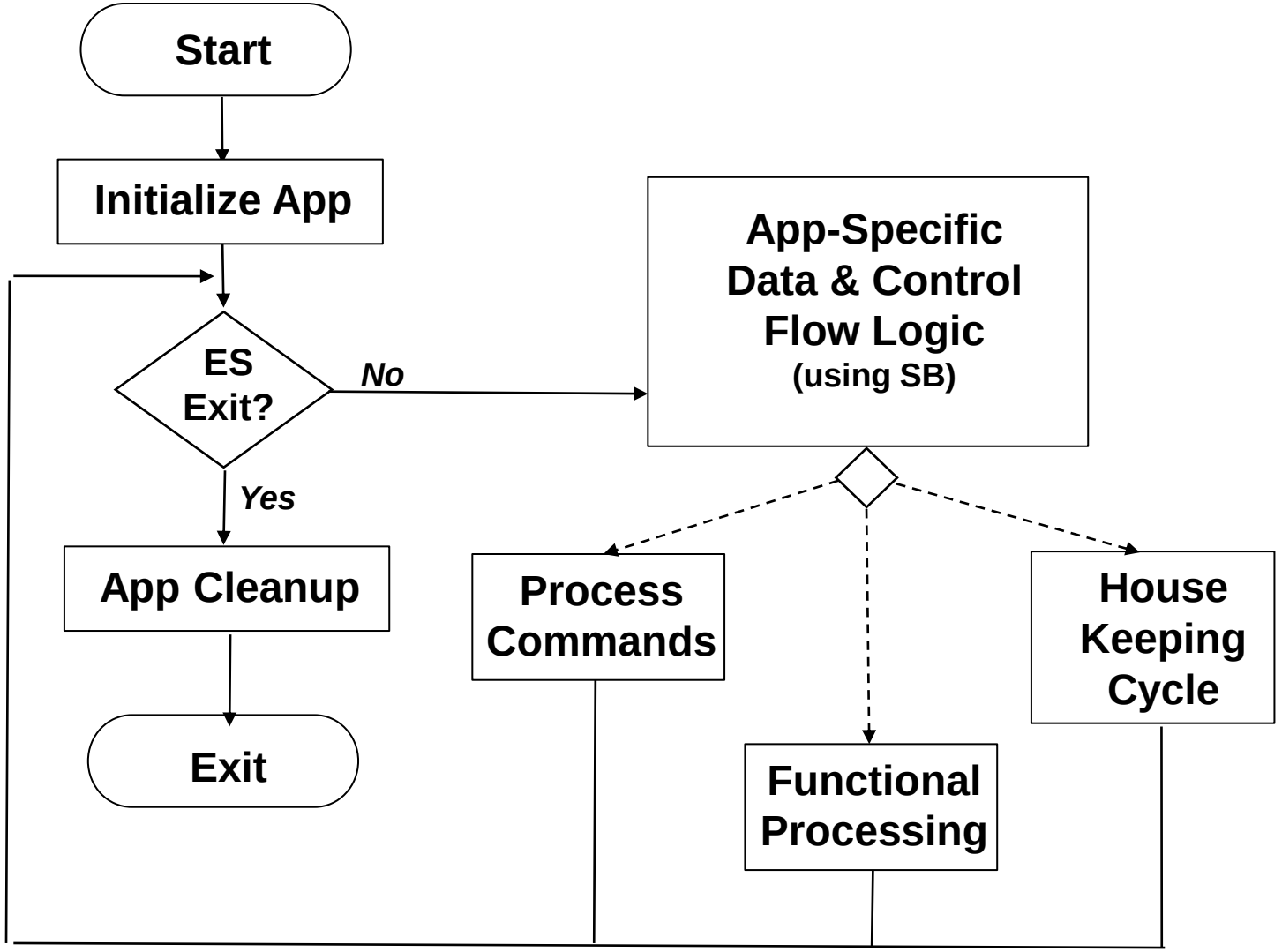
- **Message structure defined in ~/cfs/cfe/fsw/cfe-core/src/inc/ccsds.h**

```
typedef struct{
     CCSDS_PriHdr_t     Pri;
     CCSDS_CmdSecHdr_t  Sec;
} CFE_SB_CmdHdr_t;

typedef struct{
     CCSDS_PriHdr_t     Pri;
     CCSDS_TlmSecHdr_t  Sec;
} CFE_SB_TlmHdr_t;
```

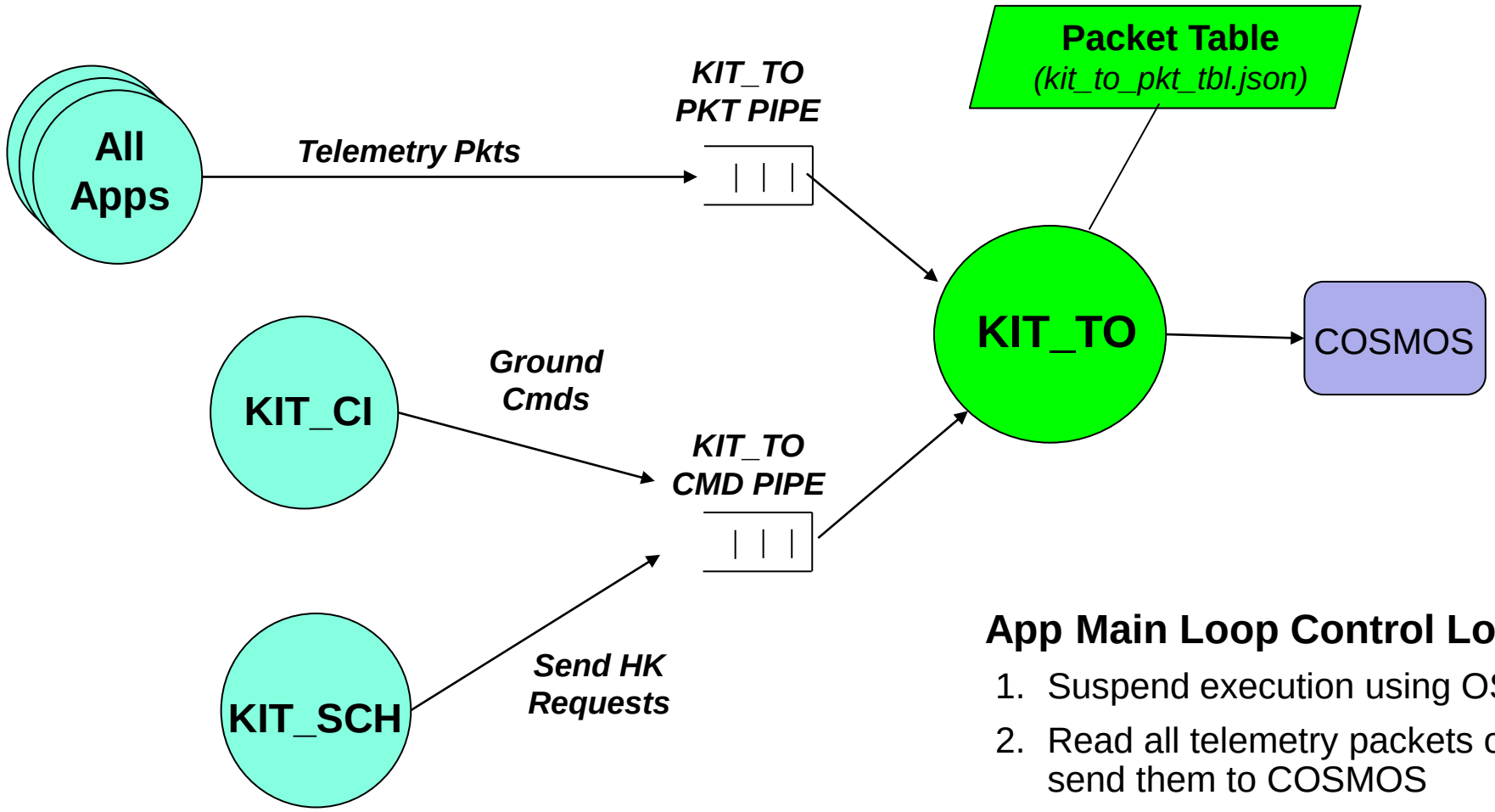
- **SB accessor functions should be used to read/write to message headers**
  - E.g. *CFE_SB_SetCmdCode()*, *CFE_SB_SetMsgTime()*, …
  - User responsible for set telemetry message time
  - Packet sequence count increment/rollover managed by SB send functions
- **cFE 6.6 supports CCSDS extended APID that significantly increases the APID range**

Application SB Service Usage



- **Initialize App**
  - **CFE_SB_CreatePipe ()**
    - **CFE_SB_Subscribe()**
    - **CFE_SB_SubscribeEx()**
  - **CFE_SB_InitMsg()**
- **Command/Functional Processing**
  - **CFE_SB_RecvMsg()**
    - Poll, Pend w/ timeout, Pend indefinitely
  - **CFE_SB_SendMsg()**
    - Both cmd & tlm messages as needed
- **Housekeeping Cycle**
  - **CFE_SB_TimeStampMsg()**
  - **CFE_SB_SendMsg()**
- **App Cleanup**
  - **cFE ES deallocates SB resources**
  - **Special cases exist like releasing zero copy pointers/buffers**

OSK Telemetry Output (KIT_TO) SB Usage Example



App Main Loop Control Logic

1. Suspend execution using OS_TaskDelay
2. Read all telemetry packets on *KIT_TO_PKT_PIPE* and send them to COSMOS
3. Read and process all packets on *KIT_TO_CMD_PIPE*

Application SB Usage Example

- **Kit Telemetry Output (KIT_TO)**
  - Read telemetry packets from SB and send to COSMOS using UDP
  - Telemetry packet table defines which packet should be sent to the ground
  - Initial table defined in *kit_to_pkt_tbl.json*
- **Initialization**
  - Create pipe *KIT_TO_CMD_PIPE* to receive ground commands and send HK requests
    - Use *CFE_SB_Subscribe()* to subscribe to
  - Create pipe *KIT_TO_PKT_PIPE* to receive telemetry packets from apps
    - Use *CFE_SB_SubscribeEx()* to subscribe to packets defined in packet table, specify number of buffers for each message
- **App Control Logic**
  1. Suspend execution using OS_TaskDelay
  2. Read all telemetry packets on *KIT_TO_PKT_PIPE using CFE_SB_RcvMsg (…, CFE_SB_POLL)*
  3. Check for packets on *KIT_TO_CMD_PIPE using CFE_SB_RcvMsg (…, CFE_SB_POLL)*
     - If send HK request then send the KIT_TO's HK packet
     - If load new packet table then unsubscribe from current messages using *CFE_SB_Unsubscribe()* and subscribe to packet defined in the new table using *CFE_SB_SubscribeEx()*

System Considerations

- **Systems must be tuned to optimize memory usage and to prevent dropped messages**
  - Requires a combination of SB configuration parameters, task priorities, and pipe subscription message buffer definitions
- **Helpful telemetry for tuning includes**
  - SB Housekeeping packet counters (No subscribers, send errors, pipe overflows, etc.) and memory stats
  - SB telemetry statistics package (sent upon command) shows pipe high water marks
- **Commands dump SB pipe, message and routing information to files that can be analyzed**
- **“Zero Copy” can be used for high speed transfers**
- **No data is preserved for either a Power-On or Processor Reset**
  - All routing is reestablished as application create pipes and subscribe to messages
  - Any packet in transit at the time of the reset is discarded
  - All packet sequence counters reset to 1

SB Functional API

SB APIs	Purpose
CFE_SB_CreatePipe	API to create a pipe for receiving messages
CFE_SB_DeletePipe	Will unsubscribe to all routes associated with the given pipe id, then remove pipe from the pipe table
CFE_SB_SetPipeOpts	Sets pipe options
CFE_SB_GetPipeOpts	Gets the current pipe options
CFE_SB_SubscribeEx	API to globally subscribe to a message when QOS and MsgLim defaults are insufficient
CFE_SB_SubscribeLocal	CFE Internal API to locally subscribe to a message when QOS and MsgLim defaults are insufficient
CFE_SB_Subscribe	API to locally subscribe to a message when QOS and MsgLim defaults are sufficient
CFE_SB_Unsubscribe	API used to unsubscribe to a message
CFE_SB_UnsubscribeLocal	CFE Internal API used to locally unsubscribe to a message
CFE_SB_SendMsg	API used to send a message on the software bus
CFE_SB_PassMsg	API used to send a message on the software bus
CFE_SB_RcvMsg	API used to receive a message from the software bus
CFE_SB_GetLastSenderId	API used for receiving sender Information of the last message received on the given pipe
CFE_SB_ZeroCopyGetPtr	API used for getting a pointer to a buffer (for zero copy mode only)
CFE_SB_ZeroCopyReleasePtr	API used for releasing a pointer to a buffer (for zero copy mode only)
CFE_SB_ZeroCopySend	API for sending messages in zero copy mode (with telemetry source sequence count incrementing)
CFE_SB_ZeroCopyPass	API for sending messages in zero copy mode (telemetry source sequence count is preserved)

SB Accessor Functions

SB Utility APIs	Purpose
CFE_SB_GetMsgId	Get the message ID of a software bus message
CFE_SB_SetMsgId	Set the message ID of a message in CCSDS header format
CFE_SB_MessageStringGet	Copies a string out of a software bus message
CFE_SB_MessageStringSet	Copies a string into a software bus message
CFE_SB_InitMsg	Initialize the header fields of a message
CFE_SB_MsgHdrSize	Get the size of a message header
CFE_SB_GetUserData	Get a pointer to the user data portion of a message
CFE_SB_GetUserDataLength	Get the length of the user data of a message (total size – header size)
CFE_SB_SetUserDataLength	Set the length field in the primary header
CFE_SB_GetTotalMsgLength	Get the total length of the message which includes the secondary header and the user data field
CFE_SB_SetTotalMsgLength	Set the length field, given the total length of the message
CFE_SB_GetMsgTime	Get the time field from a message
CFE_SB_SetMsgTime	Set the time field from a message
CFE_SB_TimeStampMsg	Set the time field to the current time
CFE_SB_GetCmdCode	Get the opcode field of message
CFE_SB_SetCmdCode	Set the opcode field of message
CFE_SB_GetChecksum	Get the checksum field of message
CFE_SB_GenerateChecksum	Calculate and Set the checksum field of message
CFE_SB_ValidateChecksum	Validate the checksum field of message

Configuration Parameters

Parameter	Purpose
CFE_PLATFORM_SB_MAX_MSG_IDS[1]	Maximum number of unique Message Ids the SB routing table can hold
CFE_PLATFORM_SB_MAX_PIPES[1]	Maximum number of unique Pipes the SB routing table can hold.
CFE_PLATFORM_SB_MAX_DEST_PER_PKT	Maximum number of unique local destinations a single Message Id can have
CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT	Default subscription message buffer limit used by CFE_SB_Subscribe(). CFE_SB_SubscribeEx() accepts a buffer limit for the message
CFE_PLATFORM_SB_BUF_MEMORY_BYTES[2]	Size of the SB buffer memory pool
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_[01-16] [2]	Define SB Memory Pool Block Sizes
CFE_PLATFORM_SB_MAX_BLOCK_SIZE[2]	Defines Max SB Memory Pool Block Size
CFE_PLATFORM_SB_MAX_PIPE_DEPTH	Maximum depth allowed when creating an SB pipe
CFE_PLATFORM_SB_HIGHEST_VALID_MSGID	Highest Valid Message Id. The value of this constant determines the size of the SB message map. The SB message map is a lookup table that provides the routing table index for fast access into the routing table.

1. This constant has a direct affect on the size of SB's tables and arrays. Keeping this count as low as possible will save memory. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'
2. For each message the SB sends, the SB dynamically allocates from this memory pool, the memory needed to process the message.