Managing Telemetry Definitions on the Fly

Petro Kazmirchuk

Terma B.V.

Schuttersveld 9
2316 XG Leiden, the Netherlands
Email: pkaz@terma.com

ABSTRACT

A parameter pool of a modern spacecraft or instrument may contain tens and even hundreds of thousands of telemetry parameters. Using specific telecommands, AIT engineers can select a subset of these parameters and request an on-board computer to downlink new TM packets that were not originally configured in the on-board software. Unfortunately, these TM packets have to be defined in a spacecraft reference database beforehand, otherwise a central checkout system will not identify these packets. This paper demonstrates a method to create such TM definitions automatically based on analysis of uplinked telecommands. This feature has been recently implemented in the Terma's spacecraft checkout software.

INTRODUCTION

An on-board software data pool (OBSW DP) comprises thousands of telemetry parameters that originate from various spacecraft reference databases (SRDB) and suppliers. This vast amount of data poses a challenge of how to allocate parameters into telemetry packets. This allocation will change and evolve depending on a phase of OBSW development and verification. For example, when integrating an instrument with a satellite bus, telemetry from this instrument and related satellite subsystems should be monitored much closely than the rest of the satellite. Also in case of a malfunction some TM parameters may need to be sampled more often than originally intended, requiring definition of new TM packets.

TM packets can be defined in OBSW on demand using special TC, picking only those TM parameters from an OBSW DP that need to be monitored at the moment. This method is widely used at the flight validation phase, when SRDB is still far from completion. And even in later stages when SRDB becomes more mature, defining static TM packets for all foreseen combinations of parameters is virtually impossible.

The ECSS Packet Utilization Standard (PUS) provides Housekeeping and diagnostic data reporting service that allows commanding of OBSW to define TM packets dynamically [2]. However, central checkout software (CCS) still does not give any special treatment to these TC, and so AIT engineers have to fall back to manually crafted static TM definitions, otherwise the newly defined TM packets coming from an on-board computer (OBC) will not be recognized by the CCS.

HOUSEKEEPING AND DIAGNOSTIC DATA REPORTING SERVICE

This service type provides for the reporting of all TM parameters from OBSW DP that are not reported by more specific services (in particular, types 4 and 5). It defines sub-services dedicated to housekeeping (HK) and diagnostic TM packets. Functionally these sub-services are identical, but they help to distinguish between HK and diagnostic packets for purposes of routing and ground processing. A mission-specific tailored PUS may add more significant differences, for example only diagnostic packets will have supercommutated parameters.

For the purposes of this paper we are interested only in a few sub-services:

- TC(3,1) – Define New Housekeeping Parameter Report
- TC(3,2) – Define New Diagnostic Parameter Report
- TC(3,3) – Clear Housekeeping Parameter Report Definitions
- TC(3,4) – Clear Diagnostic Parameter Report Definitions

- TM(3,25) – Housekeeping Parameter Report
- TM(3,26) – Diagnostic Parameter Report

A tailored PUS may have other sub-services numbered above 127, e.g. to change an existing TM packet definition in OBSW, but implementing support for them becomes trivial after support of the above sub-services is in place, and so we don't focus on non-standard sub-services in this paper.

Structure of application data of TC(3,1) and (3,2) is the same and includes:

- Structure identification (SID) that identifies a TM packet definition.
- Collection interval for this TM packet, expressed in units of a default HK scheduling rate of on-board applications (DIAG_MIN_INTERV).
- List of on-board TM parameter IDs and a few counters that describe their total number and a number of samples of each parameter.

Application data of TC(3,3) and TC(3,4) packets simply contain a list of SID of definitions to be cleared from OBSW.

Finally, source data of TM(3,25) and (3,26) packets contains a SID followed by samples of TM parameters, possibly supercommutated.

SID together with application ID (APID) and sub-service uniquely identifies a TM packet definition.

Therefore, these TC provide enough information to a CCS to create and clear its TM definitions on the fly, so that when newly defined TM packets arrive they can be identified and processed just like TM packets defined in SRDB beforehand.

OVERVIEW OF THE DYNAMIC HOUSEKEEPING IMPLEMENTATION IN CCS5

Terma's modern spacecraft checkout software CCS5 uses a format of SRDB called Mission Information Base (MIB). For automated procedures CCS5 uses Test and Operations Procedure Executive (TOPE) scripting language that is based on Tcl. Both MIB and TOPE are backwards compatible with SCOS2000, and in the same time provide much more functionality.

CCS5 is a distributed system, in which workstations host CCS Client Consoles and connect to a server. The server in turn hosts CCS Server Console, Interface manager (IFMGR) that connects to a specimen under test and special checkout equipment (SCOE), and several indexers that write a test session archive. When a test session is started, CCS5 loads MIB tables into memory where they become accessible to TOPE for reading (all tables) and even writing (some tables and columns). Moreover, CCS5 can load multiple MIBs (e.g. from different equipment suppliers) and merge them in memory, adding a special field "Source" to keep track of a MIB's origin.

The dynamic housekeeping function has been implemented in CCS5 as a dedicated component called DYNHK. The core functionality is performed by a TOPE script running on the CCS Server Console called DynHkMonitor.tcl. The major challenges that occurred on our way were:

- enhancing core C++ modules of our software to allow dynamic definition of TM packets without any disruption to an ongoing monitoring and control process;
- generating TM definitions in the MIB format based on TC parameters standardised by PUS;
- ensuring timely distribution of the new TM definitions across all workstations joined to a test session.

Fig. 1 depicts a simplified use case scenario of dynamic housekeeping:

1. A test operator sends TC(3,1) or TC(3,2) to create a new TM definition in an OBC.
2. OBC replies with TM to verify completion of this TC, e.g. using PUS Service 1.
3. DynHkMonitor derives a new TM packet definition based on analysis of the TC parameters and some configurable settings. A unique SCOS2000 packet ID (SPID) is assigned to this definition.
4. Then DynHkMonitor publishes the new TM definition to the CCS Server, all CCS Client Consoles and IFMGR, that is saved to their in-memory MIB tables. This is done using a TOPE command CCS5::remote that is used throughout the system to communicate between server and client consoles. All MIB manipulation commands are

wrapped in a Tcl library, so that a single invocation of CCS5::remote is sufficient to add or remove a TM definition. Therefore, changes to the MIB tables are done atomically.

5. The operator then sends TC(3,5) Enable Housekeeping Parameter Report Generation.
6. The OBC starts to send a new TM packet (3,25) or (3,26) periodically. It is identified and processed just like a conventional statically defined TM packet.

Similarly, after transmitting and verifying a TC(3,3) or TC(3,4) to delete a packet definition, the DynHkMonitor commands CCS5 to disable the respective TM packet definition in the MIB tables. If, afterwards, such a TM packet arrives, it will be marked by IFMGR as unknown and archived. No further TM processing (e.g. parameter extraction) is performed on it, because it is now an unknown TM packet.

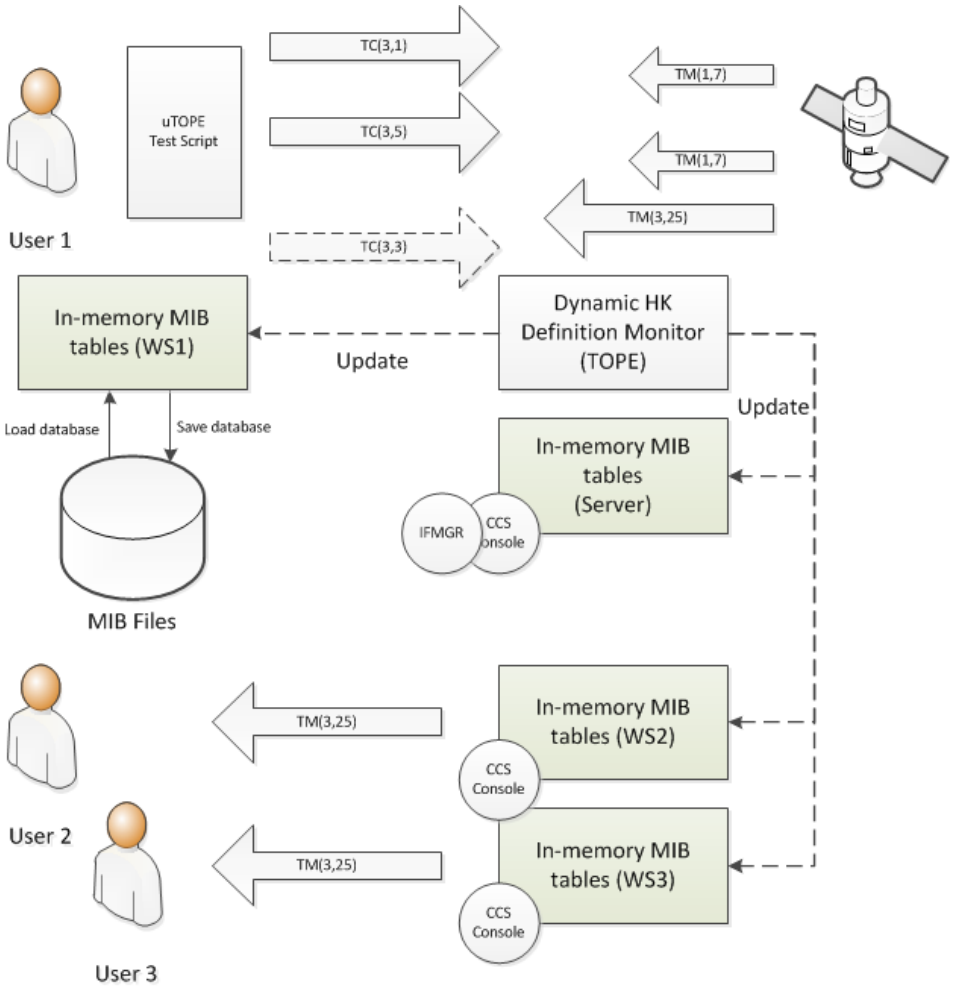

Fig. 1. Dynamic housekeeping workflow

Note that the DYNHK component allows redefinition of a packet using a SID that has already been defined. In such cases, it disables any existing TM packet definition, adds a new definition with a new SPID, and issues a log message. This means that the unique packet identification keys SPID and TPSD are never reused during a session.

Note that changes in the MIB by default stay in memory and thus are volatile and will be lost after the MIB is re-loaded (e.g. when a test session is restarted or switched with the MIB reload option). This leads us to the following considerations.

During checkout we are working with MIB contents that are themselves evolving. The OBSW development cycle may include the addition of new packet definitions to the OBSW image and the MIB. When new entries are created dynamically, we give a distinct value “DYNHK” to the Source field to indicate dynamic HK definition, and hence it is

possible to save the related MIB files separately. This saved MIB could, if desired, be re-loaded and merged during the next test session, effectively working in an incremental way.

But if the satellite has been turned off during this cycle, it may revert to only its static packet definitions. If we are sending the TC(3,x) again, it might be better to throw away the saved MIB, and rebuild the dynamic HK definition again, just as we did the first time. Hence we might make a different choice depending whether the OBSW saves its dynamically defined HK definitions in EEPROM, and whether they survive a reboot or power-off. Therefore, both saving the dynamic HK definitions back to MIB files and further loading previously saved dynamic HK definitions is optional.

MATCHING PUS TO MIB TABLES

CCS5 follows the same steps as in SCOS2000 to identify TM packets:

1. APID is extracted from a CCSDS header.
2. If a data field header is present, service type and subtype are extracted.
3. CCS5 looks up a matching record in the Packets identification criteria table (PIC) and finds location of additional identification fields PI1 and PI2 in the TM packet.
4. CCS5 looks up a matching record in the Packets identification table (PID) and finds out the SPID.
5. Finally this SPID identifies a record in the PLF table (Parameters location in fixed packets) and/or VPD tables (Variable packet definition) that specify what parameters are contained in the packet.

Allocation of TM parameters into packets can be specified using the PLF or VPD table (or both). Note that the TC(3,x) defining new TM packets use lists of parameter ID's without defining their offsets. Therefore, we have chosen to use the VPD table, because, it normally packs parameters directly after each other, which corresponds with the TC application data. This also means that variable-length string types are permitted. An incidental benefit of using VPD is that CCS5 saves all VPD samples of a packet in the session archive. This means that retrieval of the parameters in a specific packet is not dependent on the time when the packet was (re-)defined.

Because of differences between the data model of PUS and MIB, and also due to the fact that the generic PUS often lacks technical details, e.g. exact length of a field in a packet, CCS5 requires some minimal preliminary configuration in order to use DYNHK. In particular, TC(3,1) and TC(3,2) specify a collection interval in units of a default HK scheduling rate, whereas CCS5 needs this value in milliseconds, so that TM parameters can be marked as expired if they are not updated within this interval. So, we provide a user-configurable setting "schedulingRate".

Another issue that we encountered was that DynHkMonitor cannot derive APID of a new TM packet, since it is not necessarily the same as APID of the TC. So, users need to handle this using another dedicated setting that establishes a mapping from TC APID to TM APID.

PIC Table

PIC records for the new TM packets must be defined in advance as well. The reason is that the SID is not directly referenced in MIB ICD. So, CCS5 needs to look in PIC to derive PI1 and PI2 values from the SID. Here is a simplified structure of a HK or diagnostic packet:

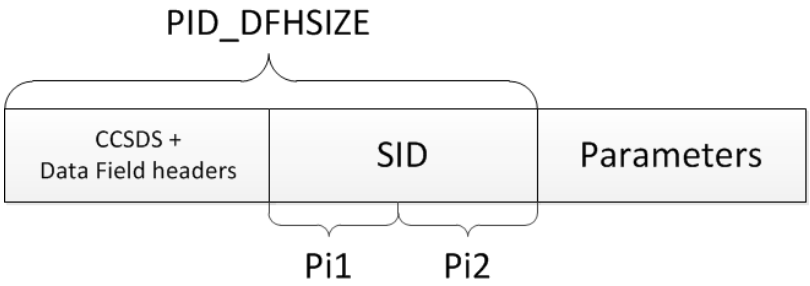

Fig. 2. Location of SID in a TM packet

Depending on a PIC table the SID will be either broken down to PI1 and PI2 values, or it can be taken as PI1, and then PI2 is not used. Also CCS5 uses PIC to calculate PID_DFHSIZE that specifies a start reference to extract variable parameters.

PID Table

Upon a reception of TC(3,1) or TC(3,2) DynHkMonitor populates the following fields in the PID table:

- PID_TYPE – 3
- PID_STYPE – 25 for HK packets or 26 for diagnostic packets
- PID_APID – derived from settings based on TC APID
- PID_PI1_VAL, PID_PI2_VAL – derived from SID
- PID_SPID – unique SPID allocated by DynHkMonitor starting from a user-configurable offset, so that it doesn't overlap with static TM definitions. Each subsequent TC(3,1) or TC(3,2) increments it.
- PID_DESCR – automatically generated packet description, e.g. "Dyn HK <SID value>"
- PID_TPSD – same as SPID
- PID_DFHSIZE – derived from PIC. Specifies a start reference to extract variable parameters.
- PID_TIME – Y (we assume that OBT is present in TM packet headers)
- PID_INTER – calculated from the collection interval
- PID_CHECK – 1 (we assume that PEC is present)

The remaining fields will be blank, and therefore assigned default values as specified by [MIB].

On reception of TC(3,3) or TC(3,4) the respective record will be disabled in the table by setting PID_VALID to false.

VPD Table

Changes to the VPD table depend on whether the TM definition contains supercommutated parameters. In the simple case, when every TM parameter is sampled only once, DynHkMonitor populates only the following fields:

- VPD_TPSD – PID_TPSD
- VPD_POS – from 1 to NPAR (taken from TC parameters)
- VPD_NAME – PCF_NAME of the parameter with PCF_PID – parameter ID from the TC.

The definition of TC(3,1) and (3,2) in [PUS] allows the NFA parameter as an indicator of whether supercommutated parameters follow the specification of single samples. If NFA is 1, then it is followed by a specification of supercommutated parameters. In this case an additional group repeater row needs to be created in the VPD table:

- VPD_NAME – REPEATER
- VPD_GRPSIZE – NPAR2, i.e. the number of parameters in the following group
- VPD_FIXREP – NREP, i.e. the number of samples

This means that in the respective TM packet the following group will contain a number NPAR2 of parameters, repeated NREP times.

CONCLUSIONS AND FUTURE DIRECTIONS

The first and foremost conclusion from this work was that flexible software architecture and data model is essential, when a need arises for such radical changes as manipulating TM definitions without any disruption to ongoing monitoring and control process. Due to the fact that CCS5 was already capable to add or change some MIB tables without a complete reload (e.g. patching calibration curves), required changes to the C++ classes related to PID and VPD tables were minimal. Also, since parameters from variable packets are always saved to a session archive, an operator can access these packets and parameters in MMI and TOPE even after the respective TM definition has been disabled in the in-memory MIB.

The next step to take DYNHK usability even further would be to add a dedicated MMI, where an operator would be able to open a display with all parameters in OBSW data pool, select some of them and request their values. Then CCS would send TC(3,1) or (3,2) to an OBC, defining a TM packet with the selected parameters and then TC(3,5) "Enable Housekeeping Parameter Report Generation", so that the OBC starts generating them, and the new parameters will appear in the TM sheet straightaway. Therefore, all intricacies of writing statements to send a TC will a long list of on-board parameter identifiers would be hidden under the hood.

Ultimately, TM packets and frames do not possess high value by themselves. Since they are only a means to transport TM parameters, all phases of spacecraft design, production and AIT would benefit from reducing volume of hand-crafted TM packets definition, and relying instead on central checkout software to handle these definitions.

A new PUS standard [3] has been published last year, and in particular many new sub-services have been added to Service 3. For example, all sub-services that have mission-specific numbers in our current implementation are now standard in this PUS. Thus, we envision adding support for the new PUS in our DYNHK soon, but it also depends on how fast the industry in general and particularly OBSW developers embrace the new standard.

REFERENCES

[1] SCOS-2000 Team, "SCOS-2000 Database Import ICD," EGOS-MCS-S2K-ICD-0001, version 6.9, *ESA/OPS-GIC*, 2010.

[2] European Cooperation for Space Standardization, "Ground systems and operations – Telemetry and telecommand packet utilization," ECSS-E-70-41A, *ESA-ESTEC Requirements and Standards Division*, 2003.

[3] European Cooperation for Space Standardization, "Telemetry and telecommand packet utilization," ECSS-E-ST-70-41C, *ESA-ESTEC Requirements and Standards Division*, 2016.