EUROCONTROL

ATNP/WG3/WP/___

26th February 1997

AERONAUTICAL TELECOMMUNICATION NETWORK PANEL

WORKING GROUP 3 (APPLICATIONS AND UPPER LAYERS)

Phuket, Thailand, 4 - 6 March 1997

Implementation of Eurocontrol CNS/ATM-1 Trials End System (TES)

Prepared by: Tony Kerr and Danny Van Roosbroek

Presented by: Danny Van Roosbroek

SUMMARY

This information paper describes the approach which has been adopted for the implementation of ATN applications conforming to the CNS/ATM-1 Package SARPs. The objective is share the experience gained in this work, and to demonstrate that the SARPs presented at ATNP/2 form a sound basis for implementation.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Scope

This paper describes the approach which has been adopted for the implementation of ATN applications conforming to the CNS/ATM-1 Package SARPs. The objective is share the experience gained in this work, and to demonstrate that the SARPs presented at ATNP/2 form a sound basis for implementation.

The objective of the Trials End System (TES) prototype is the validation of the ATN draft SARPs for air-ground applications through the development of prototype software that implements these SARPs.

The TES software implements :

- The air-ground functionality of the Automatic Dependant Surveillance (ADS) application
- The air-ground functionality of the Controller Pilot DataLink Communication (CPDLC) application.
- Both the air-ground and ground-ground functionality of the Context Management (CM) application.
- The ATN Upper Layers (efficiency enhancement option session, efficiency enhancement option presentation, ACSE edition 2 and Control Function)

Since the objective of TES is not to validate the ATN internet, the TES software relies on the transport services provided by the standard, X/Open XTI compliant, Hewlett Packard OTS 9000 stack. It also interfaces to a full ATN Internet stack.

Standard tools are used for the purpose of controlling and monitoring the TES software (starting and stopping processes, setting up traces, etc.) and for the purpose of developing and executing validation scenarios.

This TES prototype will be used by Eurocontrol for further SARPs validation through the execution of scenarios and inter-operability trials with other implementations.

1.2. Background

The Eurocontrol Trials End System (TES) project is involved in a number of activities in support of the validation of the draft ICAO Air-Ground SARPs and supporting ATN Upper Layers.

A previous paper (WG3/WP4-13 “Approach to Validation of CNS/ATM-1 Package SARPs”) outlined a methodical approach to SARPs validation. Amongst the concepts in that paper were:

- Prototype Implementation. The SARPs are validated for completeness and consistency by ensuring that they can be implemented. The TES Prototyping contract is producing implementations of the functionality specified in ADS, CM, CPDLC and Upper Layer SARPs.
- API specification. This process helped to reveal a number of inconsistencies in the early draft SARPs by considering information flows at a number of conceptual interfaces. As part of the specification work for the TES Prototyping Contract, a number of strategic end system application programming interfaces (APIs) have been specified.
- Interoperability Testing. The highest level of confidence in the draft SARPs is obtained by performing interoperability tests between independent implementations of the applications specified in the SARPs. This provides confidence that the SARPs

are written unambiguously. Scenarios are being developed to support validation by means of inter-operating independent implementations of the Air-Ground SARPs.

Fundamental to the implementation approach is the definition of a set of programming interfaces enabling a modular approach to be taken. There is no requirement for such interfaces to be standardised as SARPs, but it could be beneficial to share the interface definitions with other States and Organisations, to encourage the development of portable applications and therefore potentially decrease costs by maximising the market relevance of products developed to work within the global ATN environment.

It is emphasised that although the TES development is referred to as "prototype", this is in the sense of being amongst the pioneering implementations of CNS/ATM SARPs, and is in no sense a "throw-away" piece of experimentation. The development approach is based on sound engineering principles, leading to pre-operational quality software, rather than a quick-and-dirty rapid prototyping approach that might have been envisaged.

2. SARPS BASELINE

The Trials End System (TES) software implements version 3.0 of the air-ground applications SARPs and version 4.0 of the Upper Layer Communication Service SARPs. These were the initial baseline SARPs which were placed under configuration control after the Munich meeting of ATNP/WG3 in June 1996.

The TES philosophy has been to adhere as closely as possible to this baseline. However, certain defects reported in these versions of the draft SARPs and resolved by the WG3 subgroups have been corrected in the TES system, based on an assessment by the TES Configuration Control Board (TES-CCB) as to whether:

a) they present a blocking problem to the implementors (categorised as "BASELINE"), or

b) they affect the "bits on the wire" and hence the potential for interoperability (categorised as "INTEROP").

An implementation which stuck strictly to the baseline SARPs would not work. Indeed, the SARPs documentation is being validated by various methods (paper analysis, simulation, modelling, prototyping) resulting in a non-negligible number of defects which have been detected and reported to the SARPs editors. Some of these address serious problems in the specification of the communication protocols. A new version of the SARPs was released in November 1996 incorporating resolutions to defects reported at that time (ICAO version 1.0).

However, development of SARPs compliant systems started several months before the release of this stable version. In addition to the baseline version, SARPs-compliant systems implementors have therefore to implement the changes approved by the ATNP WG3 subgroups to correct the errors. Various criteria (such as the selected baseline version, the technical flexibility to implement a modification, the impact on the code already developed and on the test phase, the financial and human resources required to support the modification) will lead each implementor to select a subset of defects to be implemented. It is very likely that the sets of defects implemented in each SARPs implementation will be different and the risk to get non inter-operable systems is quite high.

It is therefore necessary to identify unambiguously the version of the Upper Layers and air-ground Applications and the associated defects implemented in the interworking implementations.

For each implementation, a statement of defect resolutions along with a completed Protocol Implementation Conformance Statement (PICS) should be used to study the degree of interoperability possible.

The following subsections specify the baseline for the TES implementations.

2.1. Context Management (CM) Application

2.1.1. Functional Baseline

The TES implementation of the CM application supports the full functionality as defined in the CM SARPs. Thus, the CM-ground-ASE conforms to Configuration XXVIII (i.e. CM/ground + CO-FU + UP-FU + MA-FU + FO-FU + FO-IN) and the CM-air-ASE conforms to Configuration I (i.e. CM/air + CO-FU + UP-FU + MA-FU) as defined in section 2.1.8 “Subsetting Rules” of the CM SARPs. This means that the TES system implements the Logon service, the Update service and the Contact service, and the ground CM ASE supports the ground forwarding service. The “maintain dialogue” option is provided to the CM-ground-users.

2.1.2. CM SARPs Baseline

Version 3.0 of the draft CM SARPs were released as the output version of the Munich WG3 meeting on 21st June 1996.

Since this meeting, two new versions have been produced:

1. Version 3.1 was presented as a working paper at the Alexandria WG3 meeting. This version includes the modifications agreed by SG2 at its Silver Spring meeting in late August 1996, based on defect reports CM-011 to CM-034 submitted to the CM editor. Once converted into Word Perfect format, this version is identified as ICAO Version 1.0 (ATNP/2 output version).
2. Since ATNP/2, more defects have been reported to the CM editor, CM-035 to CM-039. The resolution of these defects in ICAO version 1.0 will form the proposed ICAO version 2.0.

The implementation of the following defect resolutions has been formally approved by the TES-CCB:

ICAO reference	TES reference	Reason for implementing this DR
CM-012	PCR-42	BASELINE
CM-025	PCR-24	INTEROP
CM-013	EXTERNAL PCR-81	INTEROP
CM-037	PCR-65	INTEROP

2.1.3. Open Issues

There is no open issue related to the CM application. The CM SARPs are quite stable and the processed defect reports are not essential. Interoperability of the TES CM Application with other implementations should be easily realised having in mind the few defects impacting the protocol and the 'bits on the line' not implemented in the TES system.

2.2. Controller Pilot Data Link Communication Application

2.2.1. Functional Baseline

The TES implementation of the air CPDLC ASE supports the full functionality as defined in the CPDLC SARPs for an airborne system, i.e. “Core” CPDLC service (start, message, end and abort service) and the Downstream Clearance (DSC) services.

The TES implementation of the ground CPDLC ASE supports a subset of the full functionality as defined in the CPDLC SARPs for a ground system. Core CPDLC and DSC services are supported ; the ground CPDLC forwarding service is not supported.

Thus, the CPDLC-ground-ASE conforms to Configuration II (i.e. Core, DSC, receive/reject Forward) and the CPDLC-air-ASE conforms to Configuration II (i.e. Core plus DSC) as defined in section 2.3.8 “Subsetting Rules” of the CPDLC SARPs.

2.2.2. CPDLC SARPs Baseline

Version 3.0 of the draft CPDLC SARPs was released as the output version of the Munich WG3 meeting on 21st June 1996.

Since this meeting, two new versions have been produced:

1. Version 3.1 was presented as a working paper at the Alexandria WG3 meeting. This version includes the modifications agreed by SG2 at its Silver Spring meeting in late August 1996, based on defect reports V3-1 to V3-38 submitted to the CPDLC editor. Once converted into Word Perfect format, this version is identified as ICAO Version 1.0 (ATNP/2 output version).
2. Since ATNP/2, more defects have been reported to the CPDLC editor. The resolution of these defects in ICAO version 1.0 will form the proposed ICAO version 2.0.

The implementation of the following defect resolutions has been formally approved by the TES-CCB.

ICAO reference	TES reference	Reason for implementing this DR
V3-7	PCR-17	INTEROP
V3-12	EXTERNAL PCR-82	BASELINE
V3-15	PCR-16	INTEROP
V3-16	PCR-21	INTEROP
V3-17	PCR-43	BASELINE
V3-18	EXTERNAL PCR-83 (items 9, 16 and 17 only)	ASN.1
V3-34	EXTERNAL PCR-84	INTEROP

2.2.3. Open Issues

A lot of defects on version 3.0 of the CPDLC SARPs have been processed by the CPDLC editor causing a non-negligible modification of the baseline document. In particular, the ASN.1 description in chapter 4 has been drastically modified. Interoperability can not be achieved if the ASN.1 descriptions differ from one implementation to another.

2.3. Automatic Dependent Surveillance Application

2.3.1. Functional Baseline

The TES implementation of the air ADS ASE supports the full functionality as defined in the ADS SARPs for an airborne system, i.e. demand, event, periodic and emergency contracts.

The TES implementation of the ground ADS ASE supports the full functionality as defined in the ADS SARPs for a ground system, i.e. demand, event, periodic and emergency contracts.

The ADS Report Forwarding application (ARF ASE as defined in SARPs section 2.2.2) is not implemented in TES.

Thus, the ADS-ground-ASE conforms to Configuration VII (i.e. ADS/ground + DC-FU, EC-FU, PC-FU, EM-FU) and the ADS-air-ASE conforms to Configuration I (i.e. ADS/air +

DC-FU, EC-FU, PC-FU, EM-FU) as defined in section 2.2.1.8 “Subsetting Rules” of the ADS SARPs.

2.3.2. ADS SARPs Baseline

Version 3.0 of the draft ADS SARPs was released as the output version of the Munich WG3 meeting on 21st June 1996.

Since this meeting, two new versions have been produced:

1. Version 3.1 was presented as a working paper at the Alexandria WG3 meeting. This version includes the modifications agreed by SG2 at its Silver Spring meeting in late August 1996, based on defect reports ADS-0031 to ADS-0057 submitted to the ADS editor. Once converted into Word Perfect format, this version is identified as ICAO Version 1.0 (ATNP/2 output version).
2. Since ATNP/2, more defects have been reported to the ADS editor. The resolution of these defects in ICAO version 1.0 will form the proposed ICAO version 2.0.

The implementation of the following defect resolutions has been formally approved by the TES-CCB.

ICAO reference	TES reference	Reason for implementing this DR
ADS-0031	EXTERNAL PCR-76	BASELINE
ADS-0036	PCR-44	INTEROP
ADS-0048	PCR-12	INTEROP
ADS-0056	EXTERNAL PCR-73	INTEROP
ADS-0059	PCR-63	INTEROP
ADS-????	PCR-67	INTEROP
ADS-????	PCR-69	BASELINE
ADS-????	PCR-71	INTEROP
ADS-????	PCR-72	INTEROP
ADS-????	PCR-74	INTEROP
ADS-????	PCR-75	INTEROP
ADS-????	PCR-77	INTEROP

2.3.3. Open Issues

The ADS application runs a very complex protocol due to the variety of ADS contracts set up in parallel and to the multiplexing of these contracts over a single dialogue. As a consequence, the number of defects reported on the protocol itself is higher than for the other air-ground applications. Most of these defect resolutions have to be implemented in the first SARPs application implementations to guarantee a satisfactory level of interoperability.

2.4. Upper Layers Communication Services

2.4.1. Functional Baseline

Unlike the air-ground application SARPs, the ULCS SARPs do not define any functional subsets. The TES implementation of the ULCS SARPs supports the full functionality as defined in the SARPs, except for the Authentication functional unit of ACSE. This means that the Security Requirements parameter in the D-START service is not supported.

The TES implementation of the Session protocol supports the ATN profile of ISO/IEC 8327-2 as defined in version 4.0 of the ULCS SARPs, i.e. the kernel service restricted to the efficiency options (short-connect and null-encoding options). The negotiation to standard OSI session protocol is not supported.

The TES implementation of the Presentation protocol supports the ATN profile of ISO/IEC 8823-2 as defined in version 4.0 of the ULCS SARPs, i.e. the kernel service restricted to two of the defined efficiency options (short-connect and null-encoding options). The negotiation to standard OSI presentation protocol is not supported.

The TES implementation of the ACSE protocol supports the ATN profile of ISO/IEC 8650-2 as defined in version 4.0 of the ULCS SARPs. The Authentication functional unit is not supported.

2.4.2. ULCS SARPs Baseline

Unlike the air-ground application SARPs, the ULCS SARPs is dependent upon externally developed standards, i.e. it incorporates a number of ISO/IEC international standards by reference. The “efficiency enhanced” ISO/IEC standards have not yet passed their final ballots, so are still subject to modification. The “baseline” standards for TES have been taken to be the Draft Amendment (DAM) texts. The ULCS SARPs will be updated to refer to the final published standards when these are available.

Version 4.0 of the draft ULCS SARPs was released as the output version of the Munich WG3 meeting on 21st June 1996.

Since this meeting, two new versions have been produced:

1. Version 4.1 was presented as a working paper at the Alexandria WG3 meeting. This version includes the modifications agreed by SG3 at its Toulouse meeting in late September 1996, based on defect reports UL-DR 023, UL-DR 035, and UL-DR 074 to UL-DR 105 submitted to the ULCS editor. Once converted into Word Perfect format, this version is identified as ICAO Version 1.0 (ATNP/2 output version).
2. Since ATNP/2, more defects have been reported to the ULCS editor, UL-DR 106 to UL-DR 109. The resolution of these defects in ICAO version 1.0 will form the proposed ICAO version 2.0.

The implementation of the following defect resolutions has been formally approved by the TES-CCB.

ICAO reference	TES reference	Reason for implementing this DR
UL-DR 88	EXTERNAL PCR-58	INTEROP
UL-DR 89	EXTERNAL PCR-59	INTEROP
UL-DR 93	EXTERNAL PCR-60	INTEROP
UL-DR 94	PCR-47	INTEROP
UL-DR 98	PCR-51	INTEROP
UL-DR 100	EXTERNAL PCR-61	INTEROP
UL-DR 102	PCR-45	BASELINE
UL-DR 103	PCR-46	BASELINE
UL-DR 104	PCR-56	BASELINE
UL-DR 111	PCR-57	ASN.1

UL-DR 108	PCR-80	INTEROP

2.4.3. Open Issues

The ASN.1 description for ACSE edition 2 implemented in TES fixes two errors of ISO 8650-1:1996 DAM 2 (incorrect use of the extensibility markers in lists of enumerated values attached to an INTEGER type). This allows the ASN.1 description to be correctly compiled. This modification impacts the way a negative AARE PDU is encoded. As a consequence, the TES ACSE will only be able to exchange AARE APDUs with SARPs compliant systems implementing **the same modification**.

Note that additional ASN.1 modifications have been proposed by national bodies to ISO. None of these modifications have been implemented in TES. A risk of discrepancy between the ATN ACSE ASN.1 and the ISO ACSE ASN.1 is therefore possible.

3. TES SYSTEM ARCHITECTURE

The TES Software implements subsets of Part 2 and Part 4 of the ICAO SARPs issued from the ATN Panel Working Group 3 Sub Group 2 and Sub Group 3. Thus the System Architecture is based on these SARPs.

3.1. Architecture Overview

The TES software implements an OSI telecommunication stack based on an environment from MARBEN Produit called C-OSIAM. The architecture of TES is closely related to the C-OSIAM approach for building a telecommunication stack.

The TES software is implemented over the OSI transport service, which is provided either by the OTS 9000 product from Hewlett Packard, or by the Internet SARPs-compliant TAR/TTS system which has been developed within Eurocontrol.

The TES contains the software which provides the Upper Layers of the OSI basic reference model: layers 5 and 6 and part of layer 7 including the Association Control Service Element (ACSE Edition 2) and Control Function (CF), and 3 Application Service Elements (ASEs) implementing the ICAO SARPs of CM, ADS and CPDLC.

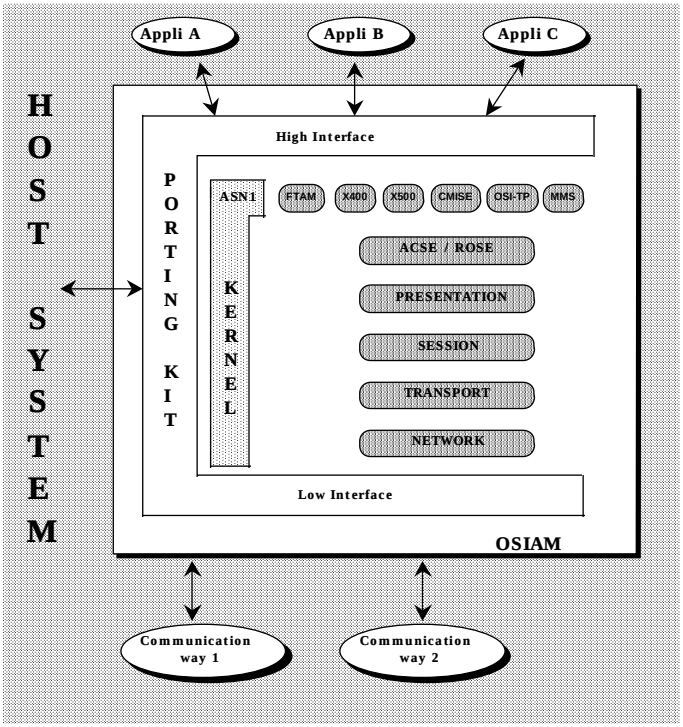
The Upper Layers are seen by the ASEs through a Dialogue Service Interface (DSI), a programming interface which corresponds closely to the abstract Dialogue Service defined in the ULCS SARPs.

3.2. The OSIAM Environment

The OSILOGIE product line from MARBEN provides a suite of telecommunication software products, based on a portable software technology. OSILOGIE products have been supplied to computer vendors, system integrators, world-wide communication operators, banks and public administration.

OSILOGIE is composed of OSIAM (the core technology for the OSILOGIE product line) and compilers (ASN.1, GDMO).

C-OSIAM (Concurrent OSI Access Method) is a suite of portable OSI compliant communication protocols, designed to deliver high performance communication solutions. It is a hosting structure in which the user can plug some existing entities, or develop specific entities. The OSIAM product line includes a set of OSI layer protocols and applications including Network Management (CMISE), Transaction Processing (OSI-TP) FTAM, X.400, X.500, ROSE, RTSE, ACSE, Presentation, Session, Transport and Network layers of the OSI reference model. Both connectionless and connection oriented capabilities are provided to support local and wide area network connectivity. The figure below illustrates a typical OSIAM environment :



Within an OSIAM product, communication interfaces with external environment are well identified and separated from the OSIAM core (High and Low Interfaces). To facilitate the writing of those communication interfaces, the Porting Kit Toolbox offers to the developer a framework that only needs to be customised for the target OS in a limited number of well defined areas. The OSIAM Generic UNIX Port (also referred as PKU) is a pre-ported implementation of an OSIAM stack for UNIX systems. The PKU is available under HP-UX. The OPU (OSIAM oPerator UNIX) is a part of the PKU. It is the human interface to administrate the different OSIAM processes.

As part of the PKU, the OPU process allows operators to manage all aspects of any OSIAM stack hosted by the local workstation. It provides the local operator with a management interface of each OSIAM stack. The interface allows the operator to control the processes, to enter textual commands, and to print messages (in response to commands or spontaneous messages) coming from the OSIAM stack.

The TES Upper Layer Process (TULP) and the TES application processes for CM, ADS and CPDLC have been developed as OSIAM processes . An OSIAM process is a UNIX process developed in the OSIAM environment. This process is made of generic OSIAM entities available in the OSIAM product providing common services (kernel, entities interfacing with the user or the underlying communication service), and specific OSIAM entities developed in the scope of a given project (e.g. the ASEs, the dialogue handler).

The dispatcher function of the kernel entity is in charge of the communication between entities (exchange of data blocks). Each entity can use the common functions made available by the kernel as the memory management (optimised to avoid unnecessary internal copies from buffer to buffer), BER/PER encoder/decoder, timer management and trace management.

The OSIAM off-line configurator is a program allowing the generation of C source code representing the required configuration. The configuration is described in a text file. The off-line configurator generates the corresponding C source code, which has to be compiled and linked with entities or external libraries.

A stack configuration defines what entities are included in the stack (the protocols supported by the stack), the SAPs that links entities, the addressing information carried by the SAPs, and the sizing of resources that the stack is allowed to use.

3.3. The Validation Tool

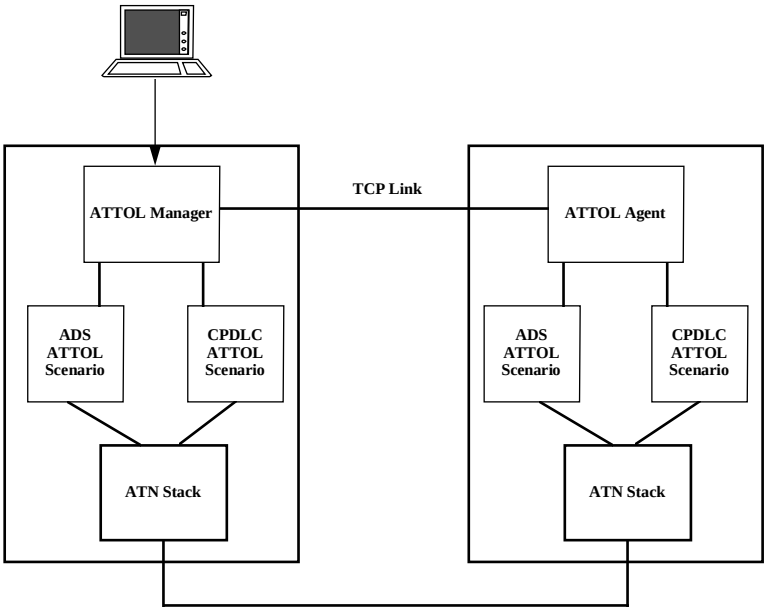
The ATTOL System Test Tool from MARBEN has been used for writing and executing test scenarios for TES development. A number of scenarios have been written and executed during the development phases. Some of these scenarios could be used as SARPs validation scenarios without any modification.

ATTOL System Test

The ATTOL product line is divided in two distinct groups, depending on the test finality : ATTOL Unit Test, for unit and software integration tests, and ATTOL System Test for software validation, system integration and system acceptance.

It enables the test phases to become formal and traceable : Scenarios are written by the user with the ATTOL language. Then, the code associated with the defined scenario is automatically generated. During its execution, the scenario generates binary traces and after its completion, a post-processor allows to generate some test reports.

As a COTS, ATTOL System Test allows scenarios to be run on the local machine. The scenario is linked with an API, which can be a communicating API. A TES specific development, identified as Validation Tool , allows the scenario management to be centralised. It is possible to run and synchronise scenarios on different machines of the TES platform. The figure below illustrates an example of the configuration :



ASNEDIT

ASNEDIT is a tool developed by MARBEN for generating BER/PER encoded PDUs, via a user-friendly Motif-based interface. This tool is used off-line by the validation software in order to prepare encoded data for ATTOL System Test. At run-time, ATTOL System Test reads these data, and injects them into the ATN stack (at the required level up to the dialogue level).

3.4. TES Software Design Description

The TES Software has 2 main parts : the components dedicated to the APIs and the ones managing the telecommunication Stack.

3.4.1. TES APIs

The TES Software APIs are CM, CPDLC, ADS, DSI and ADB. These APIs are located in different libraries and the CM, CPDLC and ADB interfaces share a common library. These libraries work together. In fact, a User process (ATTOL) can simulate CM (ADB), CPDLC, ADS and DSI applications. The CM, CPDLC, ADS and DSI APIs are asynchronous and the ADB one will be synchronous.

CM and ADB API

The CM API exhibits connectionless services through a connection oriented mechanisms. The airborne and ground services are managed by this API. It uses some common functionality.

The ADB (Addressing Database) API exhibits connectionless services.

CPDLC API

The CPDLC API exhibits connectionless services through a connection oriented mechanisms. The airborne and ground services are managed by this API. It uses some common functionality. The CPDLC-Forward service is not offered by this API.

ADS API

This API exhibits connectionless services. The airborne and ground services are managed by this API. It uses some common functionality. The ADS-Forward service is not offered by this API.

DSI API

The Dialogue Service Interface API provides a defined lower service boundary to an ATN Application Abstract Service Element and allows two ATN Application Abstract Service Elements to communicate. This API offers and exhibits connection oriented services.

Common Functionality to APIs

This library offers the following functionality :

- Initialisation and closing of the API component
- Registration and Cancelling Registration to the TES stack
- Waiting for events from the TES Stack
- Checking of common parameters to CM, CPDLC and ADS APIs (UserIds, Class of Communication etc.)
- Management of Error return codes
- Global variables to the APIs
- Global functions to the APIs

3.4.2. TES Stack

The TES Stack components are :

- the CM, CPDLC and ADS entities
- the DS, ACSE, CF, ADB entity

- the Efficiency Enhancement Option Presentation entity
- the Session Efficiency Enhancement Option entity
- the XTI entity

There are separate airborne and ground entities for CM, CPDLC and ADS ASEs. However the ASE services are common to the airborne and ground version of each ASE. Moreover, the ASE services include both User and the Provider parts for provision for future use in a stack configuration than the TES one. Common functionality shared by the ASE entities includes:

- tracing functions
- error logging functions
- Finite State Machine managing

3.5. OSIAM Entity and Stack Realisations

The OSIAM environment features an off-line configurator which facilitates the distribution of the protocol elements in different UNIX processes. Thus, a number of different stack configurations can be built for testing purposes.

The configuration used for system testing is shown below. In this figure the following objects are represented :

ADS interface - the exposed ADS service,

CM interface -the exposed CM service,

CPDLC interface -the exposed CPDLC service,

DSI interface - the exposed/used DSI service,

XTI interface - the used XTI service,

ADS ASE - the OSIAM entity implementing the ADS protocol,

CM ASE - the OSIAM entity implementing the CM protocol,

CPDLC ASE - the OSIAM entity implementing the CPDLC protocol,

Dialogue/ACSE/CF - the OSIAM entity implementing the ULCS protocols,

Presentation FB - the OSIAM entity implementing the efficiency enhanced Presentation protocol,

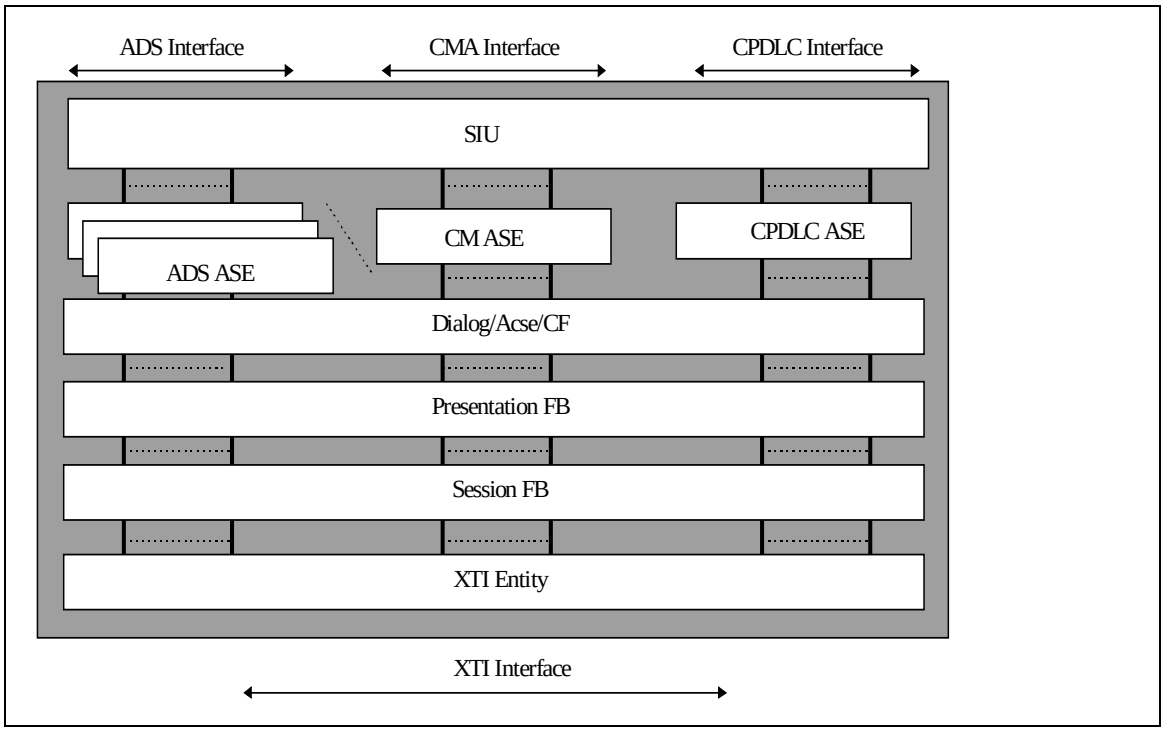
Session FB- the OSIAM entity implementing the efficiency enhanced Session protocol,

XTI entity - the OSIAM interface entity which allows an OSIAM stack access to an external transport service provider,

SIU (Standard Interface for UNIX) - the OSIAM interface entity which allows OSIAM stacks to communicate through IPC mechanisms and allows external API users to access different OSIAM stack services.

For readability of the figure, some simplifications have been made :

- there is no distinction between air and ground entities
- there is no SAP between the entities.



TES Stack allowing more than one user.

The "railway tracks" in this figure are the representation of different SAP trees.

It is intended that the user should be able to configure the "operational" stack. But during the system integration/qualification (and later validation) phases, some other configurations might be useful. From an existing configuration, the "stack(s) cut-out" is performed with the OSIAM OMK tool.

3.6. APIs

An API is the user view of a service. It allows the user to exercise the service. So the API definition and the underlying OSIAM service definition are very closely linked. The API specification defines the "external view" of the service.

The main objectives of an API definition are the following:

- Fix on the stack initialisation/registering mechanisms.
- Fix on the API nature
- Fix on the exhibited types (ASN.1 generated types or use of formatting/unformatting functions).
- Fix on miscellaneous implementation constraints.

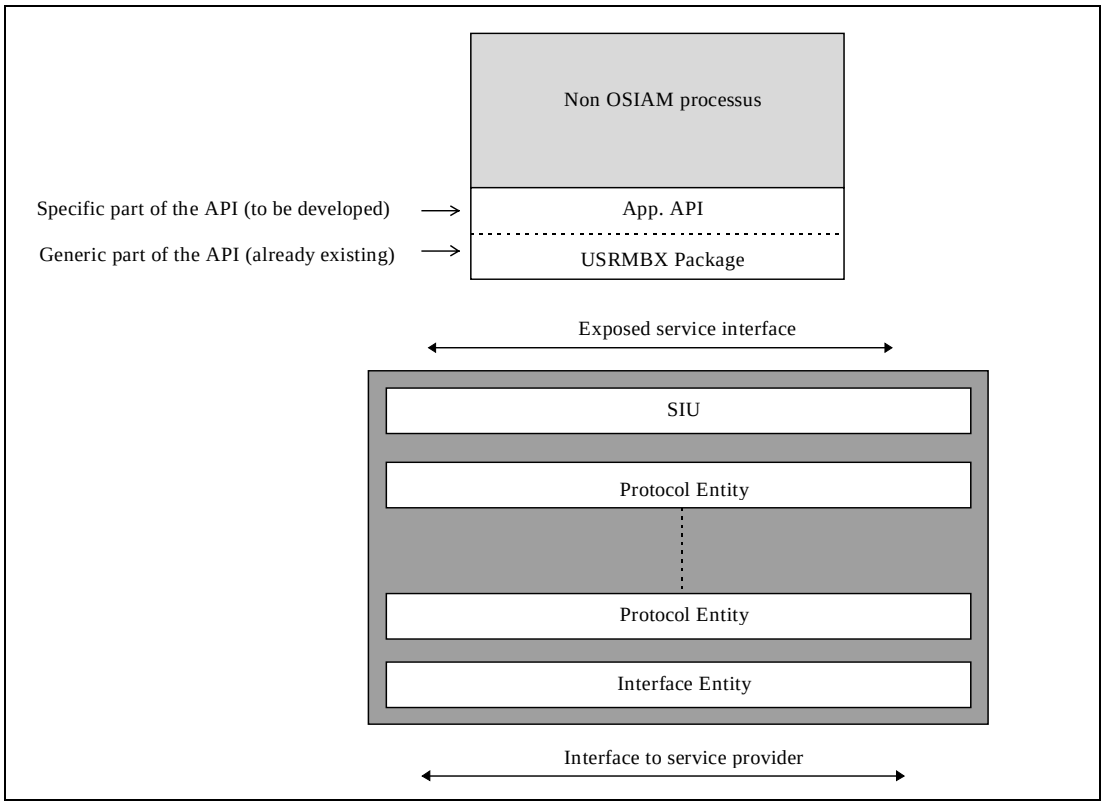
In the TES software, three APIs are developed, one for the ADS service, one for the CM service and one for the CPDLC service. A first set of functions of the APIs is common to the three APIs and is called the generic part. In this part, the APIs will make use of the USRMBX package to:

- manage the communication between the telecom stack and the user application (management of the mailboxes)
- manage buffers

The APIs also have a specific part which is responsible for

- checking the parameters passed by an API user
- encoding/decoding messages to or from the PER format before transmitting them to the telecom stack via mailbox or before transmitting them to the user application by reference via function call return.
- Checking that the sequence of function call invocations is correct.

The following figure illustrates the two parts of the APIs.



API / stack general architecture

3.6.1. Initialisation Mechanisms

Generally, the API initialisation is done at the beginning of the user process execution. The initialisation needs a specific entry point in the API. The main goals are the following:

- Allocation and set up of all the internal data of the API.
- Give a "user-name" which will be used by the API to create (in read mode) the OS dependant system name of the API user mailbox (named pipe) used for the reception of the incoming events from the stack.
- Open the OSIAM stack mailbox (in write mode) for the sending of the outgoing events. Generally the OSIAM mailbox is retrieved through environment variable value.

After the API initialisation, the user has to register to the stack. The registration needs a specific entry point in the API. From a general point of view the main goals of the registration are to declare the "bind address", the "used service" and the registering mode (shared/exclusive) ; the registration mechanisms allow multi-registering from the same API (on different "bind addresses" for instance). In the case of the TES implementation, the following features will be chosen :

- At the registering phase the user has not to specify the service since this is implicitly the one of the used API. Consequently the "used service" will be set up by the API itself.
- A user application registers only once by giving its own Identifier (ICAO identifier) and its own qualifier (CM, CPDLC or ADS). For the same reason the registering mode has to be exclusive to avoid concurrent application registering on the same Identifier.
- Moreover, the "bind address" is useful to give the current application type (either the AircraftId or the ICAOFac) in the registration service primitive.

Conversely, a cancel-registration phase has to be done by the user for cancelling a previously issued registration. The cancel-registration needs a specific entry point in the API.

An API termination has to be done just before the user process exits. The termination needs a specific entry point in the API. The goals are the reverse of the initialisation function :

- Release all the internal data of the API,
- Delete the API user mailbox (named pipe) used for the reception of the incoming events from the stack.
- Close the OSIAM stack mailbox.

3.6.2. API Nature

This section details the TES APIs :

- There are 5 APIs. these APIs allow the user to handle CM, CPLDC, ADS, DSI and ADB services. However, these APIs offer also common functionality's related to CM, CPDLC and ADS APIs. Physically, these APIs consist of 6 libraries : CM, CPDLC, ADS, ADB, DSI and COMMON libraries.
- In the SARPs, the CPDLC services are exhibit connectionless services. The CM services offer connectionless services to the user. The ADS services seem to be connectionless. Thus, the choice has been made to have exhibited connectionless services for ADS.
- The APIs handle the Multi processes / Mono User registrations. The User has a unique ICAO Identifier (airborne or ground) and may consist of 1 CM application and/or 1 CPDLC application and/or 1 ADS application. A single ATTOL process can access to 1,2 or 3 APIs at the same time but it can also access these APIs in several processes. There are 5 possibilities :
    - ♦ 1 process CM + CPDLC + ADS
    - ♦ 2 processes : CM + ADS, CPDLC
    - ♦ 2 processes : CM + CPDLC, ADS
    - ♦ 2 processes : ADS + CPDLC, CM
    - ♦ 3 processes : CM, ADS, CPDLC
- For testing purposes, the TES software is able to handle 20 users. Thus, the API handles 20 user registrations. But the APIs authorise only one User per UNIX process. For example, if 20 users are needed each one consisting of 3 applications CM, CPDLC and ADS in the same UNIX process, there will be 20 UNIX processes to simulate the behaviour of these Users. Each process can access the CM, CPDLC and ADS APIs concurrently in order to minimise the number of processes. The limitation to 20 users is handled by the stack. This choice of the limitation to 1 User in a single UNIX process allows a simplified API and thus testing phase which can be minimised.

- The APIs exhibit a File Descriptor which will be used by ATTOL environment to wait on events from several sources.

3.6.3. ASN.1 Exhibited Types

All the structures issued from the PLC 409 ASN.1 compilation of CM, CPDLC and ADS abstract syntax are exposed by the API. However, they will not be handled by the users but populated by the users through the use of formatting functions provided by the API. These data structures will be encoded within the API entity by means of the encoding routines generated by PLC 409. For received data, the decoding process is inverse of the encoding operation. The users extract parameters from the C structs by using "unformatting" functions, inverse of the formatting ones.

3.6.4. Addressing Database

The addressing database provides the one-to-one mapping between an application identifier and its presentation address. The information is put in the addressing database by the CM user (information from logon, update, forward functions). The information is got by the Dialogue entity to retrieve the presentation address of a given application with its identifier as search key.

For each entry in the addressing database, the information fields are as follows:

- The identifier which could be either the aircraft-id or the ICAO-Facility-Designator.
- The application entity qualifier ("ADS", "CMA", "CPC").
- The presentation address which is made of SAPs concatenation : null presentation selector, null session selector, transport selector and network address.
- The version number of the application

The addressing database is implemented as an extension to the Dialogue entity. The main advantages of such a solution are :

- Concurrent access (read by the Dialogue entity and written by the CM user) are solved by the OSIAM asynchronous interactions dispatching.
- The addressing data base could be accessed through several ways : from the data base hosting entity, from a non OSIAM external process, from an other OSIAM entity, from the OSIAM operator.
- Access from the data base hosting OSIAM entity. Such an access is needed by the Dialogue entity for retrieving the presentation address from the application name (AE-title). The Dialogue entity may access directly to the data base.
- Access from a non OSIAM external process is needed by the CM user for updating the contents of the data base.
- Access from an other OSIAM entity (either in the same OSIAM stack or in an other OSIAM stack) is available through an OSIAM service (in case of an entity in an other OSIAM stack, access is transparent through the SIU component). Such a need is not currently identified.

From the OSIAM service and API specification point of view, the identified primitives are the following:

- Add or replace an entry (all fields have to be set).
- Remove an entry by application search (identifier, qualifier, version).
- Remove an entry by presentation-address search.
- Retrieve an entry by application search (identifier, qualifier, version).

- Retrieve an entry by presentation-address search.

For administrative access the following set of commands is provided :

- Add or replace an entry (both fields have to be set).
- Remove an entry by application search (identifier, qualifier, version).
- Remove an entry by presentation-address search.
- Retrieve an entry by application search (identifier, qualifier, version).
- Retrieve an entry by presentation-address search.
- List the current entries in the data base.
- Upload the addressing data base from a configuration file.
- Download the current content of the addressing data base in a configuration file.

3.6.5. Control And Monitoring Function

The control and monitoring function is implemented with the support of COTS products (OSIAM administration task and ATTOL System Test).

The control and monitoring of the communication/protocol objects (OSIAM stacks and their entities) is issued by OSIAM administration task. The control and monitoring of the tests scenarios/results are issued by ATTOL System Test.

The OSIAM administration task provides features which allow management of OSIAM stacks. The IPC mechanisms used for the exchanges between the OSIAM administration task and any OSIAM stack are portation dependant. For the TES implementation, an HP-UX pre-ported environment is used.

The information exchanged is formatted in BCR (Binary Command Request) from the OSIAM administration task to the OSIAM stack and in BMR (Binary Message Response) from the OSIAM stack to the OSIAM administration task.

The OSIAM administrative tool is a COTS but it can be seen as a "programmable tool". In fact some general/generic functions are provided and some others have been implemented by the entity developers according to their needs. The OSIAM administration task is a framework in which the entity developers can plug their instances of the generic commands as well as some specific ones.

For OSIAM stack administration, there are 4 categories of commands :

1. The general commands which are related to the kernel or to the stack itself. Such commands are available for any OSIAM stack, they are not related to a specific implementation / entity. Some of these commands are directly executed by the OSIAM administration task (such commands always begin with a $).

2. The general commands which are related to the kernel or to the stack itself. Such commands are available for any OSIAM stack, they are not related to a specific implementation / entity. Some of these commands are submitted by the OSIAM administration task to one of its administrated stacks (such commands never begin with a $).

3. The generic commands, such commands are provided for any entity but their processing has to be coded by each entity developer (such commands never begin with a $).

4. The specific commands which are related to a given entity, such commands are not available in a standard way and their provision is the entity developer responsibility (such commands never begin with a $).

There are three types of commands in the first set (general commands executed by the OSIAM administration task)

- Process execution ($load, $open, $close, $abort, $shutdown, $stop).
- Event management ($wait, $strwait, $timeout, $acton, $actoff).
- Miscellaneous ($status, $default, $explicit, $register, $rex).

The second set of commands (general commands submitted by the OSIAM administration task to the OSIAM stack) are the following:

- OSIAM stack activity control (start, stop, vary sap on/off, vary ent on/off/force).
- Warm reconfiguration of OSIAM stack objects (define sap/entity/pool/interaction ...).
- Tracing and logging mechanisms (debug, vary sap trace, vary ent trace, log).
- Consultation of OSIAM stack objects (find sap, show sap/address/buffer ... , show ent all/cnx).
- Miscellaneous (macro).

The third set of commands (generic commands submitted by the OSIAM administration task to the OSIAM entities) are the following:

- Search/consultation of entity context (find ctx, show ent ctx).
- Modification of entity objects (vary ent parm/flag/cnx).

The fourth set of commands (specific commands submitted by the OSIAM administration task to the OSIAM entities) are entity specific.

Hereafter as a summary of the administration constraints for an entity / service developer :

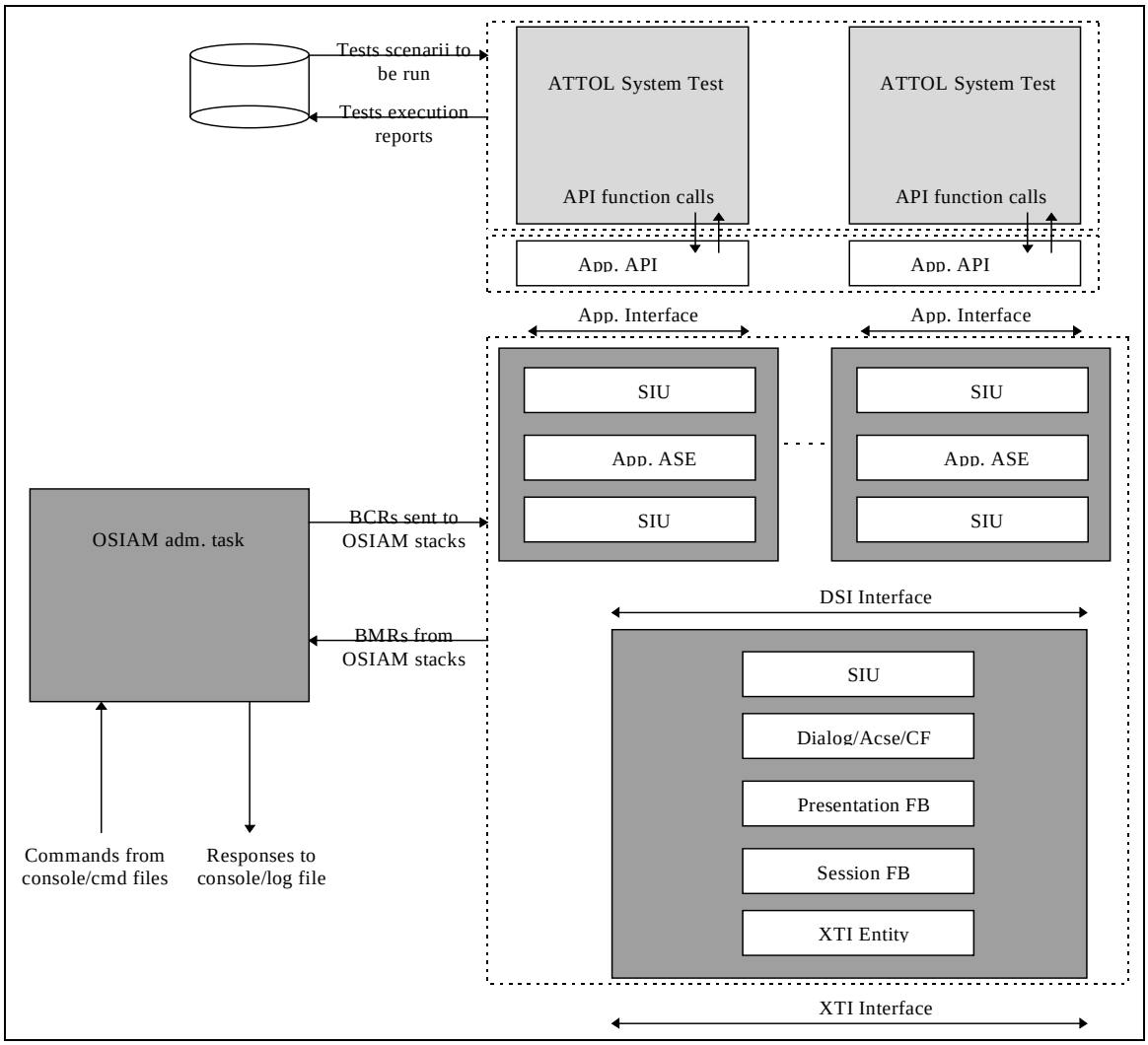
- Provision for a search context function (this not mandatory).
- Provision for a show context function (this not mandatory).
- Provision for a cancel connection function (this not mandatory).
- Definition of the entity parameters and flags (this not mandatory). A flag can be seen as a Boolean parameter. Entity parameters allow to manage/tune some variables related to the entity behaviour (for instance waiting queue size, timer value, ...). The exact semantic of a parameter is left to the entity designer convenience.
- Provision for the service interactions formal description (this not mandatory). This description allows the ITF function (Intelligent Trace Facility) to dump the interactions passing on the SAPs in user friendly format.
- At any time during its execution, an entity is able to build and to send spontaneous BMR (for instance to notify protocol machine state changes).

Moreover for the use of specific commands, an entity developer has to :

- Describe its specific commands tree with the CBF syntax
- Declare its specific commands tree to the kernel by the AddBcom() call during its boot phase
- Manage the corresponding BCRs/BMRs for the exchanges with the OSIAM administration task.

As described above, the OSIAM administration task allows the control and monitoring of any OSIAM stack and inside the stack of any entity (provided the entity developer has developed the needed provisions for such a function). For the API, the OSIAM

administration task does not provide any features for control and monitoring functions. The figure below describes the OSIAM control and monitoring of an Upper Layers stack.



Control and monitoring representation