ATNP WG3/WP 14- _____

AERONAUTICAL TELECOMMUNICATIONS NETWORK PANEL

WORKING GROUP 3 (APPLICATIONS AND UPPER LAYERS)
Bordeaux, France
September 29 - October 2, 1998

Backwards Compatibility Considerations

Prepared by: G. Saccone

SUMMARY

This paper presents an approach to modifying the CM protocol and user requirements in order to facilitate backwards compatibility, based on the request of WG3 at Utrecht.

1. Introduction

This paper gives a discussion current backwards compatibility operation of the CM ASE, and gives an option of modifying the protocol and user requirements in order to accommodate future version negotiation.

2. Discussion

CM version negotiation was implemented in the CM ASE based on the fact that CM can never be "forward compatible". However, as currently specified, a version 2 CM-air-ASE will never be backwards compatible with a version 1 CM-ground-ASE. Here is why:

If a version 2 CM-air-ASE attempts a logon with a version 1 CM-ground-ASE, the CM-ground-ASE will reject the logon, and CM-air-ASE provides the CM-ground-ASE version number to the CM-air-user. Since the CM-air-ASE version number is "hardwired" into the ASE, the CM-air-user has no choice but to revert to a version 1 CM-air-ASE. This means that the CM-air-user will have to carry both versions of the CM application, and provide a "super-user" that would control both the version 1 and 2 CM-air-users. So while the application could be made backwards compatible by virtue of using the same CM-logon service as for version 1, once an incompatible version number is determined, the version 2 application stops. Note that the situation on the ground is different for the case when the CM-ground-ASE version number is greater than the CM-air-ASE. It is assumed that the ground will accept the logon (thereby implicitly requiring all future CM initiations to be done with a CM-logon service) and respond, telling the CM-air-user its version number. Therefore the ground is always assumed, as the receiver of the initiation, to be backwards compatible. This means that a mechanism is in place to accommodate backwards compatibility on the ground, but not in the air.

In order to rectify this, and perhaps not require subsequent CM-air-ASE implementations to carry multiple CM versions, some changes to the protocol and user requirements are suggested. These changes have the advantage of being implementable in either package 1 or package 2 without affecting interoperability. Since a package 1 aircraft will never be logging on as a package 2 aircraft, the new protocol will never get executed, nor will the user requirements come into play. If it is deemed that these changes are not necessary for package 1, then they can be implemented in package 2 and will give more of a measure of backwards compatibility without having to carry multiple CM versions.

3. The Concept

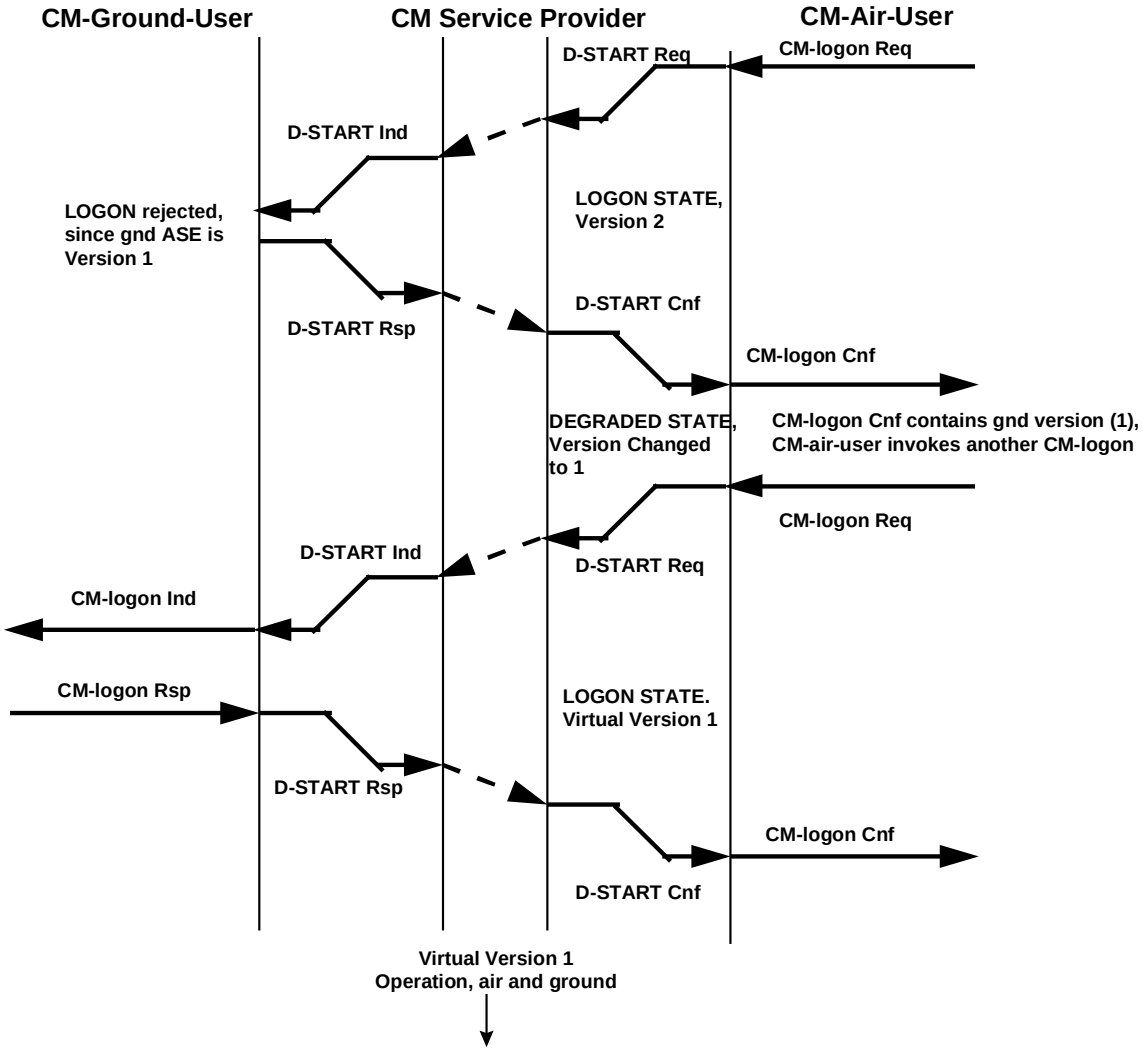
The part of the application that would need to be modified is the CM-air-ASE, specifically where a D-START confirmation is received during a LOGON. Currently, if the DS User Version number of the D-START confirmation (which is the CM-ground-ASE version number) is less than the CM-air-ASE version number, the CM-logon service confirmation is invoked with the CM-ground-ASE number passed. The CM-air-ASE then goes into the

IDLE state. From there, it is up to the user to determine the proper CM application (i.e. CM version) to use, and restart the process.

The suggestion is to create a new state, called something like “DEGRADED”. In the above example, instead of going into the IDLE state the ASE would go into the DEGRADED state. There would be no other inputs into the DEGRADED state other than another CM-logon service request. Once in the DEGRADED state, the CM-air-ASE will need to set the CM-air-ASE version number to the version number it received from the CM-ground-ASE. This should be possible without too much trouble, by creating a temporary version number. This number will then be used in the subsequent CM-logon.

Upon receipt of CM-logon service confirmation which contains a version number (the only result from an attempted logon when the CM-ground-ASE version number is less than the CM-air-ASE version number), the CM-air-user should invoke another CM-logon service. This may be done automatically, of course, and an aircrew need not be aware of this.

This is depicted in the following time-sequence diagram. The CM-ground-ASE version number is 1, the CM-air-ASE version number is 2.



A new section would also be created in the CM-logon Service Request section of the CM-air-ASE protocol. This section would be similar to the existing 2.1.5.3.2.6.1.1, and would in fact be numbered 2.1.5.3.2.6.1.2 and would read along the lines of the following:

2.1.5.3.2.6.1 Upon receipt of a CM-logon service request:
2.1.5.3.2.6.1.1 (as is currently)
2.1.5.3.2.6.1.2 If the CM-air-ASE is in the *DEGRADED* state and the CM-logon parameter value is a *Logon Request*, the CM-air-ASE shall:

(ed note: the check of the *Logon Request* is to ensure that only a CM-logon service is being attempted, as no other action is allowed except for abort)

a) create a CMAircraftMessage APDU with a cmLogonRequest APDU-element based on the *Logon Request* parameter value,
b) set the CM-air-ASE version to the value contained in the *DS User Version Number* parameter,
c) invoke D-START request with the following:

1)…(same as 2.1.5.3.2.6.1.1)
2)…
3) the modified CM-air-ASE version number as the D-START *DS User Version Number* parameter value,
4) …
5) …(same as 2.1.5.3.2.6.1.1)

d) start timer t_{logon}, and
e) enter the *LOGON* state.

Now the logon continues as usual. There would need to be a some mechanism in the protocol to reset the CM-air-ASE version number upon entering into the IDLE state (i.e. erase the temporary variable, leaving the hard-coded version number). The state tables would also need to be updated accordingly.

More user requirements would have to be included. First, the user must always start CM services with a CM-logon, and this must remain unchanged. The user must also be aware that additional services (for example, the CM-server logon) would not be available when running in degraded mode, and must not be allowed to be invoked. This could easily be accomplished by the user function. Further, since the CM-air-user is not responsible for setting the CM-air-ASE version number, and there is no way the version number can be decremented except for an incompatible logon attempt, the "sanctity" of the version number is retained.

4. Conclusion

This is an example of how backwards compatibility may be accomplished without the need for carrying separate CM applications for each subsequent CM version.