**The SAS CMS Project**

**Functional Reference**

*Jeppesen Crew Pairing*

*Document Type: Functional Reference*

*Version: 1.2*

*Last Changed: 26 Feb 2013*

Table of Contents

[1 Introduction 7](#_Toc340138309)

[2 The plan concept in CMS 8](#_Toc340138310)

[2.1 Database plan 8](#_Toc340138311)

[2.2 File plan 8](#_Toc340138312)

[2.3 Pairing on the database 8](#_Toc340138313)

[2.3.1 Open a Database Plan 10](#_Toc340138314)

[2.3.2 Export Scenario 10](#_Toc340138315)

[2.3.3 Create a plan from SSIM 11](#_Toc340138316)

[2.3.4 Fetch Trips 11](#_Toc340138317)

[3 Crew need 12](#_Toc340138318)

[3.1 Service Manning 12](#_Toc340138319)

[4 Planning areas 14](#_Toc340138320)

[5 Base definitions and Constraints 15](#_Toc340138321)

[5.1 Base Definitions 15](#_Toc340138322)

[5.2 Base Constraints 15](#_Toc340138323)

[5.2.1 Rules and Parameters 15](#_Toc340138324)

[5.2.2 External Table 16](#_Toc340138325)

[5.2.3 Production Days 17](#_Toc340138326)

[5.2.4 Daily Production Calculation 17](#_Toc340138327)

[5.2.5 Constraints per Crew Position 18](#_Toc340138328)

[6 Minimum connection 21](#_Toc340138329)

[6.1 Configuration possibilities 21](#_Toc340138330)

[7 Check-In exception 23](#_Toc340138331)

[8 Optimizer Cost function 24](#_Toc340138332)

[8.1 Parameters 24](#_Toc340138333)

[8.2 Real costs 24](#_Toc340138334)

[8.2.1 Cost of production day 24](#_Toc340138335)

[8.2.2 Cost of deadhead 24](#_Toc340138336)

[8.2.3 Hotel cost 25](#_Toc340138337)

[8.2.4 Per diem 25](#_Toc340138338)

[8.2.5 Local transport 25](#_Toc340138339)

[8.3 Penalties 25](#_Toc340138340)

[8.3.1 Duty minute 25](#_Toc340138341)

[8.3.2 Aircraft change 26](#_Toc340138342)

[8.3.3 Points over 90-buffer 26](#_Toc340138343)

[8.3.4 Duty free break 26](#_Toc340138344)

[8.3.5 Layover at other base 26](#_Toc340138345)

[8.3.6 Check out after midnight on last duty 27](#_Toc340138346)

[8.3.7 Trip length 27](#_Toc340138347)

[8.3.8 Deadhead in roundtrip with same AC 27](#_Toc340138348)

[8.3.9 Middle duty deadhead 28](#_Toc340138349)

[8.3.10 More than one middle duty deadhead 28](#_Toc340138350)

[8.3.11 Max duty in x day trip 28](#_Toc340138351)

[8.3.12 Expensive hotel stations 28](#_Toc340138352)

[8.3.13 Day stop 29](#_Toc340138353)

[8.3.14 Duty time in duty period 29](#_Toc340138354)

[8.3.15 Duty time in duty period 29](#_Toc340138355)

[8.3.16 Duty time in duty period 30](#_Toc340138356)

[8.3.17 Duty time in calendar week 30](#_Toc340138357)

[8.3.18 Extended duties 30](#_Toc340138358)

[8.3.19 Table of regularity 31](#_Toc340138359)

[8.3.20 Soft Locks 31](#_Toc340138360)

[8.4 Constraints 31](#_Toc340138361)

[8.4.1 Base constraints 31](#_Toc340138362)

[8.4.2 Average duty per duty day 32](#_Toc340138363)

[8.4.3 Number of “trivselstopp” per base 32](#_Toc340138364)

[8.4.4 Trips starting and ending in time of day interval 32](#_Toc340138365)

[8.4.5 Number of duty days with same AC-type 33](#_Toc340138366)

[8.4.6 Min and max number of hard duties per day 34](#_Toc340138367)

[8.4.7 Max number of trips containing station 34](#_Toc340138368)

[8.5 Retiming – Costs and Rules 34](#_Toc340138369)

[8.5.1 General parameters 34](#_Toc340138370)

[8.5.2 Retiming Penalty 35](#_Toc340138371)

[8.5.3 Retiming Rules 35](#_Toc340138372)

[8.5.4 Legal A/C Turnaround 36](#_Toc340138373)

[8.5.5 Retiming Regularity 36](#_Toc340138374)

[8.5.6 Consistent Retiming 37](#_Toc340138375)

[9 Performing an optimization 38](#_Toc340138376)

[10 Base Variants 39](#_Toc340138377)

[10.1 SKN short haul 39](#_Toc340138378)

[10.2 SKI Flight Deck 39](#_Toc340138379)

[11 Aircraft Rotations 40](#_Toc340138380)

[12 Cabin Crew LH Distribution 41](#_Toc340138381)

[13 NOP to OAG 42](#_Toc340138382)

[14 Short/Long Haul integration 43](#_Toc340138383)

[14.1 Process 43](#_Toc340138384)

[14.2 AS/AH handling 43](#_Toc340138385)

[14.3 Parameter set for optimization runs 44](#_Toc340138386)

[15 TOR – Table of regularity 45](#_Toc340138387)

[15.1 Usage 45](#_Toc340138388)

[15.1.1 Commands 45](#_Toc340138389)

[15.1.2 Visualization and reports 46](#_Toc340138390)

[15.1.3 Configuration 46](#_Toc340138391)

[16 Soft locks 47](#_Toc340138392)

[16.1 Usage 47](#_Toc340138393)

[16.1.1 Commands 47](#_Toc340138394)

[16.1.2 Visualization and reports 48](#_Toc340138395)

[16.1.3 Configuration 48](#_Toc340138396)

[17 Trip Naming 49](#_Toc340138397)

[17.1 Naming using a sequence number 49](#_Toc340138398)

[17.2 Naming a single trip 49](#_Toc340138399)

[17.3 Naming a group of trips 50](#_Toc340138400)

[18 Retiming Commands 51](#_Toc340138401)

[18.1 Replace retiming alternatives with current retiming 51](#_Toc340138402)

[18.2 Set retiming alternatives... 51](#_Toc340138403)

[18.3 Merge and spread retiming alternatives in retiming groups 52](#_Toc340138404)

[18.4 Clean up Retiming Alternatives Table 52](#_Toc340138405)

[19 GUI 53](#_Toc340138406)

[19.1 Filters 53](#_Toc340138407)

[19.2 Mark fast 54](#_Toc340138408)

[20 Production estimation 55](#_Toc340138409)

[20.1 Calculating production from different sources 55](#_Toc340138410)

[20.1.1 Pairings 55](#_Toc340138411)

[20.1.2 Time table 56](#_Toc340138412)

[20.1.3 Fleet plan 56](#_Toc340138413)

[20.1.4 Etables 56](#_Toc340138414)

[20.1.5 Parameters 61](#_Toc340138415)

[20.2 Usage 62](#_Toc340138416)

[21 InterBids Trip Export 64](#_Toc340138417)

Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date | Description |
| 0.2 | Jonas Carlsson | 2006-11-07 | Added APC cost function chapter |
| 0.4 | Martin Karlsson | 2006-11-27 | Final adjustments for TX1 delivery |
| 0.5 | Anna Olsson | 2007-06-25 | Added new APC penalties |
| 0.6 | Anna Olsson | 2007-07-18 | Added new constraints and per diem cost |
| 0.7 | Jonas Carlsson | 2007-08-21 | Added TOR and Soft locks chapters |
| 0.8 | Jonas Carlsson | 2007-08-21 | Added chapters on Planning Areas, Base Definitions/Constraints and Short/Long haul integration |
| 0.9 | Anna Olsson | 2007-08-29 | Added chapters on Base Variants, Trip Naming and GUI - Selects. |
| 0.10 | Jonas Carlsson | 2007-09-06 | Added chapter on Base constraints |
| 0.11 | Anna Olsson | 2007-10-03 | Added chapter on Copy Trips CC Long Haul. Added text about selects. |
| 0.12 | Erik Kilborn | 2007-10-03 | Added chapter on Building Aircraft Rotations |
| 0.13 | Anna Olsson | 2007-10-09 | Added chapter on NOP to OAG |
| 0.14 | Anna Olsson | 2007-10-23 | Updated definition of select Nop leg incl. other fleet. |
| 0.15 | Jonas Carlsson | 2007-11-12 | Added chapter on export of trip data to CMP for production estimation |
| 0.16 | Erik Kilborn | 2007-11-21 | Added description of KFL |
| 0.17 | Anna Olsson | 2007-11-21 | Updated description for trip start/end interval constraint. |
| 0.18 | Jonas Carlsson | 2007-11-22 | Added section on crew need |
| 0.19 | Anna Olsson | 2007-11-23 | Added constraint Max number of trips containing station. |
| 0.20 | Jonas Carlsson | 2007-11-25 | Various updates and restructuring |
| 0.21 | Anna Olsson | 2007-12-04 | Updated text about aircraft rotations |
| 0.22 | Anna Olsson | 2007-12-04 | Added process description to section about the copy trips script. |
| 0.23 | Anna Olsson | 2008-01-17 | Added section about Interbids trip filtering. |
| 0.24 | Anna Olsson | 2008-01-21 | Added description of how hotel cost is calculated for a duty free break. |
| 0.25 | Jonas Carlsson | 2008-01-24 | Created v25 from merging alpha1 and v24 |
| 0.25-0.1 | Jonas Carlsson | 2008-02-21 | Various updates |
| 0.25-0.2 | Jonas Carlsson | 2008-02-22 | Added a revised chapter on pairing export to Manpower. |
| 0.25-0.3 | Jonas Carlsson | 2008-02-24 | Added information on process steps. |
| 0.26 | Jonas Carlsson | 2008-03-03 | Created a v26 from alpha5 doc |
| 0.27 | Jonas Carlsson | 2008-03-03 | Added documentation on minimum connection |
| 0.28 | Jonas Carlsson | 2008-03-05 | Cleaned up the use of FD/FC/CC/FlightDeck/CabinCrew |
| 0.29 | Jonas Carlsson | 2008-03-13 | Misc updates. |
| 0.30 | Erik Gustafsson | 2008-04-28 | Minor fix of KFL documentation |
| 0.31 | Erik Gustafsson | 2008-06-10 | Minor fixes for export to Manpower. |
| 0.32 | - | - | - |
| 0.33 | Anna Olsson | 2008-07-24 | Added information about penalty for duty time in duty period. |
| 0.34 | Erik Gustafsson | 2008-10-01 | Fixes for RC2 |
| 0.35 | Erik Gustafsson | 2008-10-01 | Updated Minimum connection |
| 1.0 | Erik Gustafsson | 2008-10-06 | Fixes for FAT |
| 1.1 | Erik Gustafsson | 2008-10-10 | Added info on CR29, CI Exception |
| 1.2 | Anna Olsson | 2008-10-22 | Updated process for SKI SH Distribution |
| 1.3 | Anna Olsson | 2008-11-07 | Updated AC-Rotation method |
| 1.4 | Anna Olsson | 2008-11-11 | Updated description of Export Scenario |
| 1.5 | Erik Gustafsson | 2008-11-17 | Fixes for SP1 |
| 1.6 | Erik Gustafsson | 2008-12-03 | Minor fix regarding planning period. Updated info on importing etables. |
| 1.7 | Anna Olsson | 2008-12-19 | Updated Build Rotations method |
| 1.8 | - | - | - |
| 1.9 | Jonas Carlsson | 2009-10-01 | Added info on tripserver filtering for SP5 |
| 1.10 | Erik Arnström | 2010-06-10 | Added info on cost elements for extended duties. |
| 1.11 | Erik Arnström | 2010-06-28 | Added info on how to select input for APC and updated info on selects |
| 1.11 | Pierre Nordblom | 2010-08-25 | Updated cost, Duty time in duty period/week |
| 1.12 | Anna Olsson | 2010-08-25 | ? |
| 1.13 | Stefan Hammar | 2012-04-24 | Added information about retiming. |
| 1.13.1 | Stefan Hammar | 2012-06-19 | Corrections after review of retiming info. |
| 1.14 | Arvid Berzelius | 2012-10-29 | Changed menu entries for Select to Filter |
| 1.15 | Per Andersson | 2012-11-08 | Updated Interbids Trip Export chapter according to the Interbids 5 migration. |
| 1.2 | Viktor Almqvist | 2013-02-26 | CMS2 updates. |

# Introduction

This document, the functional reference for Jeppesen Crew Pairing, is describing all functionality that is customized for pairing in the SAS CMS project.

Reports, rules and administrative tasks are described in separate documents. Also functionality shared between other parts of CMS is documented separately. Although the processes around certain areas of functionality are touched upon, the complete workflow will be described in a step-by-step guide handled by PG.

# The plan concept in CMS

## Database plan

The most important plan in the CMS system is the production database plan. The plan is a continuous roster where all departments active in the crew planning- and follow-up-process works against the same common production plan. However, different departments are responsible for writing in different parts of the roster. There are different applications working against the same plan. Such applications are Studio (for Tracking, Planning or PreRostering), Manpower and stand-alone applications such as Crew Info. Database plans enable multi-user access to one common plan so that the whole organization shares the same world-view.

## File plan

As a complement to the database-plan planners have the possibility to create file-based plans. The file-based plans are similar to what is used in today’s pairing and rostering installations. These plans are used for rostering and pairing scenarios, optimization and testing.

File-based plans can either be created from SSIM or CTF files or created by exporting a scenario from a database plan.

Access to file-based plans is restricted to one single user. If a second user attempts to access the same plan, a dialog is presented with a warning that the plan is locked and access will be read-only.

## Pairing on the database

The pairing step in the planning process is the step that benefits the least, and is the most detached, from the database. It is possible to work on file plans all the way up until the trips must be made available for rostering. However, for facilitating training planning and other rostering by hand the trips must be available in the database. It is also important to start from the database to correctly handle carry-ins.

OAG timetables are not available in the database, and weekly local plans cannot be created from the database. Therefore an SSIM-file with SK and OAG legs is needed for the pairing process. A schematic overview of an example of high-level planning process is shown below.

SSIM

SSIM select only OAG

Merge

Roll-out to dated

Subplan

Merge

Dated local plan from file

Load/open DB

Subplan with all fleets

Save to file “Export Pairing scenario”

Dated local plan

Weekly Subplan

Weekly localplan

1

1

2

2

3

4

2: Merge with ”skip flights that already exist in Local Plan”

3: Remove all flights that aren’t touching the planning period. Hide all pairings.

Steps for preparing data (one planner):

Steps for each planning area:

APC

5

3: only pairings that are in or touching the planning period

Subplan

Subplan

Subplan

(Carry-in trips)

Clean up

3

Figure

Planning area steps (over the dotted line):

1. Create a weekly localplan and subplan from SSIM
2. Optimize weekly solution
3. Roll-out-to-dated on the dated localplan created from the database/SSIM. Only trips that are in or touching the planning period.
4. Merge new dated subplan with carry-in subplan
5. Open database and fetch trip solution. Save.

Data preparation steps (under the dotted line):

1. a) Create a dated localplan from SSIM containing only OAG  
   b) Export pairing scenario from the database to create a dated local and subplan
2. Merge the OAG localplan with the database localplan using the option “Skip flights that already exist in localplan”. The resulting localplan is the one used in step 3 above.

Steps 1 and 2 can be done in a centralized manner for all planning areas at one time, or separately for each planning area.

1. Clean up the subplan to prepare carry-in subplans for merging with the dated solutions (step 5 above), remove all flights that are not touching the planning period.

The data preparation steps might have to be done daily in certain periods to make sure that the dated solutions can be moved to new time table versions and kept up-to-date.

### Open a Database Plan

A database plan is opened from the plan manager just like a file plan. When the plan is opened a load filter dialog appears where the period, planning area and parameter set can be selected before fetching data from the database.

The database-plan names are written in lower-case letters and are in the test environment called things like Deliveries/Alpha/a1\_static/a1\_static.

#### Period

Since the database plan contains all data not selecting a period and/or planning area will make opening the database plan very slow. The actual period opened from the database is the specified planning period and an additional five days before and ten days after the planning period. The buffer is needed to find plausible connections for trips inside the planning period. This makes it important to only have legs/trips in the planning period as input to optimization, otherwise the optimizer will have problems generating solutions.

**Example**: If the user enters start of planning period = 1Feb2008 and end of planning period = 1Mar2008 then the period opened from the database will be 25Jan2008-10Mar2008. The planning period in Rave will be set to 1Feb2008-1Mar2008.

#### Planning Area

The planning areas that can be selected are predefined. The planning areas are described in section 4 .

#### Product

CMS behaves a bit differently for Pairing and Rostering (i.e. which rule set to use, what happens during an export etc). Choose “Pairing” to use CMS in pairing mode.

#### Parameter Set

The parameter sets that can be chosen are those that are saved by the user in $CARMUSR/crc/parameters/<user>. When clicking OK the selected data is loaded from the database. If no parameter set is chosen the default parameters of the rule set will be applied.

It is possible at any time to load, or save, a parameter set from the parameter form.

#### Rule Set

After the load of the crew in the period Studio loads an appropriate rule-set, Pairing\_CC for cabin areas and Pairing\_FC for flight-deck areas.

### Export Scenario

A snapshot of the database can be exported to a file plan for doing optimization. To export a scenario, select File -> Export -> Scenario when having the desired area open. A confirm save dialog is opened. When save is chosen, a new dialog is opened where plan properties may be set before saving. The default local plan name is based on the data loaded from the database, and the default sub plan name is based on the user and time,

When the scenario is exported a snapshot of all etables used for planning is saved to LpLocal and SpLocal in the new plan. Tables in LpLocal are shared between all sub-plans in the same local-plan. Only tables usually changed in sub-plans are stored in the SpLocal. A user can override this by copy a file from LpLocal to SpLocal. In that case the file in SpLocal will be used instead.

### Create a plan from SSIM

The database does not support exporting a standard week scenario. For creating a standard week file plan, one has to use SSIM based input. Refer to the Crew Planning User Guide, available from the help menu, on how to create a new local plan from an SSIM file.

#### Adding etables

Legality and cost calculations require a number of etables available only in the database. When starting from a SSIM file these won’t be available. To help the user with the process of extracting these etables from the database a function is available:

Planning Tools > Local Plan > Import ETABs from database

It will extract the etables from the database defined in the configuration of CMS and copy them to the current local plan’s temporary storage, making them immediately available. If the local plan already contains etables a warning will be shown. It is necessary to save the local plan to keep the etables if the plan is closed.

### Fetch Trips

This is performed by using the standard functionality for fetching trips between sub-plans, available also on file plans.

1. Make sure that the sub-plan is based on the latest dated local plan on file.
2. Open the database with the appropriate planning area.
3. Remove the existing trips in the planning area. It they are not removed, a fetch will result in lots of overbookings.
4. Planning Tools -> Subplan -> Fetch -> Trips…
   1. Select localplan, subplan, and optimizer solution with the trips to publish.
   2. Select only trips that starts in planning period + x days
5. Create leg sets for fetched OAG legs, they are fetched as NOP.
6. Check overcovers, NOPs and other fleets on-duty (should not be too many, when the sub-plan was based on the latest localplan).
7. Save the database

# Crew need

Crew need is the Jeppesen label for crew composition. The total crew need is calculated from the JarOps crew requirement plus the SAS service manning level. The JarOps manning level is dependent on the aircraft type, or actually the number of available seats. The JarOps level is however defined in the external table *crew\_need\_jarops.etab*, where the required number of crew must be entered rather than strictly dependent on available passenger seats.

## Service Manning

For cabin crew, the additional crew required for SAS service manning must be entered into the service table *crew\_need\_service.etab* (see screenshot below). An Air Purser (AP) is always required and entered already in the JarOps table, but the extra levels for both Air Stewards (AS) and Air Hosts/Hostesses (AH) must be entered in the table. Please observe that only the additional need should be entered, not the desired total number of cabin crew. The total is defined by summarizing the JarOps crew need and the Service crew need.

The service need is priority based on a per region (SK iso SKI, SKD, SK, and SKS) and position (AS/AH) basis. The algorithm will try each priority level for the current region/position combination from lowest number to highest. On the first match found, it will return the value in the posval column.

For each priority level (per region/position combination) you can have one and only one keytype. A keytype is a property type of a flight, and tells the algorithm what value to calculate. This calculated property of the flight is then compared to the value in the keyval column. If the dynamically calculated value for a flight matches the value in a keyval field for a region/position/prio combination, then we have match and “posval” amount of crew in position “position” is the required service manning.

There are 9 different types of keyvalues to use:

*AcType*

*Flight*

*Route*

*Area*

*STC*

*Route+AcType*

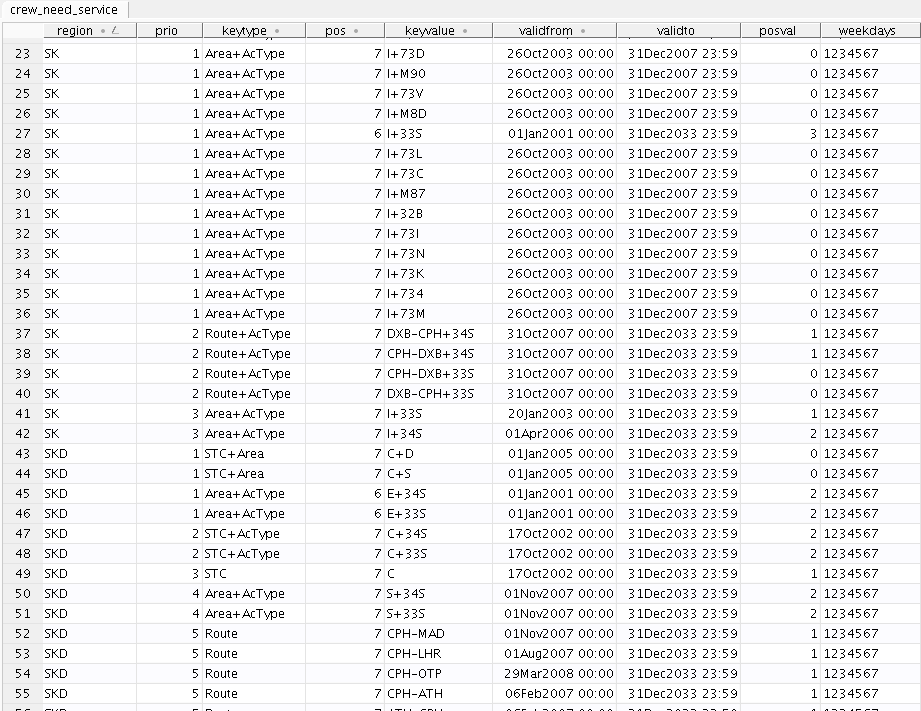
*Area+AcType*

*STC+Area*

*STC+AcType*

No wild cards are allowed in the table and if no hit is found the need will default to zero, i.e. there will be no additional requirements for that flight.

For flight crew, the additional need (only for flights with BLH > 6:30) is hard-coded in the system and not possible to influence by the user. For long haul flights with BLH > 6:30, one extra Relief Pilot (FR) is needed; BLH > 12:30, two extra FR are needed. For short haul flights with BLH > 6:30, one extra Copilot (FP) is needed; BLH > 12:30, two extra FP are needed. It is possible to define in a set "Exception airports NOT needing additional flight crew manning" it is possible to define a station which do not require additional flight crew need, e.g. SFJ.



# Planning areas

To be able to in a simple way filter the legs and trips pertaining to a certain planning area, a number of filters can be found in the parameter form, tab “Period and Area”.

The Planning Area filtering is active in most selections, exceptions are:

* Show and Hide (Trip General)
* Legs -> Show -> … (Trip General)
* Show … / Show Legal/Illegal -> (Window menu)

Using the options in sections Trip and Leg it is possible to define the following planning areas:

* Planned at CPH
  + Flight Deck A320
  + Flight Deck MD80
  + Cabin Crew A320, MD80
* Planned at OSL
  + Flight Deck + Cabin Crew F50
  + Flight Deck + Cabin Crew Leisure
  + Flight Deck B737 all bases
  + Cabin Crew B737 all bases
* Planned at STO
  + Flight Deck MD80
  + Flight Deck B737
  + Cabin Crew MD80, B737
* Planned at ICONT
  + Flight Deck A330, A340
  + Cabin Crew A330, A340

# Base definitions and Constraints

The standard base definition/base constraint functionality is active and makes it possible to create daily and total constraints (in absolute numbers) per base/period.

## Base Definitions

SAS operates from a number of bases, Stockholm (STO), Copenhagen (CPH), Oslo (OSL), Stavanger (SVG) and Trondheim (TRD). Since SAS has divided the production into local companies each country operates only from one base, except SAS Norge who operates their short haul traffic from three different bases. Long haul traffic is still, however, planned as a merged timetable, which implies that all three bases are used in the same plan. During summer a base in BLL is sometimes opened up for SAS Danmark.

There are also three additional bases, Beijing (BJS), Shanghai (SHA) and Tokyo (NRT), outside Scandinavia used by SAS Intercont for their international stationed cabin crew.

Command: Planning Tools -> Subplan -> Bases -> Edit Base Definitions

Use this command to edit the bases available, which are active and which is the default base

## Base Constraints

The trip creation process is based on the concept of a home base. A base serves as a geographical and/or organization entity which the crew members belong to. Normally each crew member is assigned to one of the predefined bases. Base constraints are used regularly in CPH, STO and OSL.

The trip creation process respects the constraint that each trip must start and end at the same base. The set of generated trips can then be divided into groups according to the base they start/end at. The number of trip days starting/ending at a given base must correlate with the crew capacity belonging to it. This correlation can be more precisely described in terms of constraints on base production days.

### Rules and Parameters

For Base Constraints there is a set of penalties and parameters defined.

Parameter: Use Base constraints

Turns on or off all of the base constraints all at once.

Parameter: Type of plan

Decides the type of plan, can choose between dated and standard. When set to standard, the base constraint start/end dates are not used, also the number of base production on the same weekday during several weeks will be combined into one number.

Constraint: Use min daily prod constraint

Constraint that adds a penalty for fewer base production days on all days during the period, than what is stated within the base constraints table.

Constraint: Use max daily prod constraint

Constraint that adds a penalty for more base production days on all days during the period, than what is stated within the base constraints table

Constraint: Use min total prod constraint

Constraint that adds a penalty for fewer base production days during the total period, than what is stated within the base constraints table

Constraint: Use max total prod constraint

Constraint that adds a penalty for more base production days during the total period, than what is stated within the base constraints table

Constraint: Use min total prod ratio constraint

Constraint that adds a penalty for fewer base production days during the total period, than what is stated within the base constraints table. Is expressed as a ratio percentage

Constraint: Use max total prod ratio constraint

Constraint that adds a penalty for more base production days during the total period, than what is stated within the base constraints table. Is expressed as a ratio percentage

The status report is configured using the following parameters, found on the Report tab:

Parameter: Combine positions within Base constraint status report

When running the Base Constraints status report, all positions will now be combined into one total value and not divided into several values per individual position.

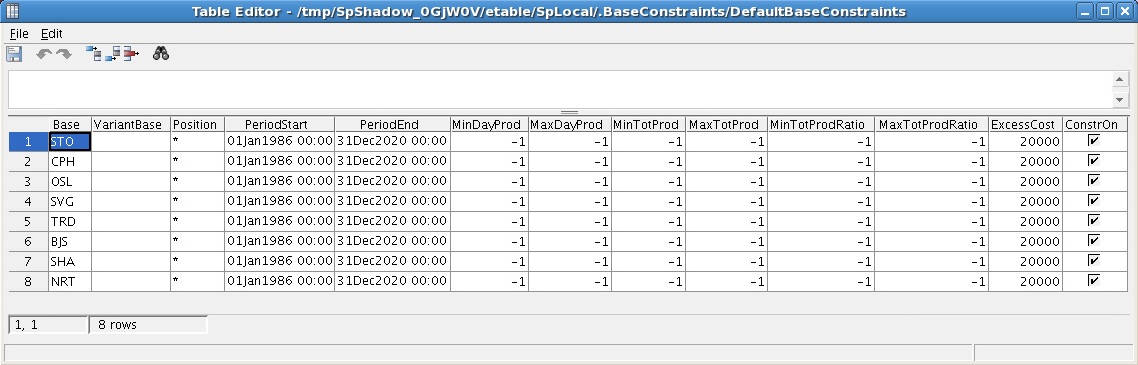
Parameter: Show Base constraints in Status report

Within the status report, a specific section will appear that states all base constraints that exist with their used values and limits.

### External Table

To set the base constraints, use the command

Command: Planning Tools -> Sub-plan -> Bases -> Edit Base Constraints



The definitions of base constraints are stored in an external table. The table have the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Example | Description |
| Base | String | “ARN” | Base name |
| VariantBase | String | “” | Used for base variants only and not considered in this package. See APC Crew Pairing Manual |
| Position | String | “AP” | Name of the crew position that you want to constraint. “\*” = position independent |
| PeriodStart | Abstime | 01jan1986 | Start for validity period |
| PeriodEnd | Abstime | 31Dec2037 | End for validity period |
| MinDayProd | Integer | -1 | Min daily production for given base |
| MaxDayProd | Integer | 10 | Max daily production |
| MinTotProd | Integer | -1 | Min total production |
| MaxTotProd | Integer | 345 | Maxtotal production |
| MinTotProdRatio | Integer | 5 | Min total production ratio (in percent 0-100) |
| MaxTotProdRatio | Integer | -1 | Max total production ratio (in percent 0-100) |
| ExcessCost | Integer | 1000 | Cost for each production unit exceeding the constraint |
| ConstrOn | Boolean | True | Validity of constraint |

Each row in the table can represent six constraints, each corresponding to one min/max value. A field value of -1 invalidates the constraints. The same effect for all the constraints is achieved by setting value ConstrOn to False.

### Production Days

The daily production at a base is a number of trips that:

* Start at the base at the given day or before and
* Will end at the base at the given day or later

A total production is the daily production summed over all or part of the planning period. The days for which the adding is performed can be specified in terms of the base constraint validity period.

### Daily Production Calculation

The add-in provides opportunity of specifying constraints in terms of percent or absolute numbers. By using the percent constraint you can keep the ratio between total production for one base and the total production for all bases within specified limits.

For dated problems the contribution to a particular day is one if

* given day overlaps the trip time period and
* day belongs also to base constraint validity period.

Otherwise the contribution is 0.

If the Position field is set to a particular crew position, the assigned value for the crew position may be more than one.

Consider trip A’s contribution to Saturday’s production. Saturday belongs to the constraint validity period and trip A continues into the Saturday. The trip’s contribution to Saturday’s production is one.



For standard problems the contribution equals the number of times the trip occurs on given day. For standard planning problems the calculation is more complex because of the problem’s periodical nature. It means that during a particular day there may be trips in production which started in one of the previous planning periods.

Consider a standard weekly problem. Assume there is a trip in the solution starting on Thursday with duration of 8 days. The contribution of the trip to Thursday’s production is two.



### Constraints per Crew Position

For all APC jobs except Variable Crew in One Shot (See APC Crew Pairing Manual), you use a constant crew complement vector that all trips will use. With the Position field set to “\*” for all constraints, all trips will contribute to the base constraints values with 1 position, regardless of the trips’ crew complement.

Choose what to show in the in the APC status report:

* For a short summary in the APC Status report (recommended), turn *Show Base constraints in Status report* on.



* For a more detailed description, turn on the parameter *use\_pdl\_status* and set *status\_pdl\_file* to “hidden/BaseConstraints.output”.



If you want to limit for example the FA position’s contribution to a specific base, set the “Position” field to “FA” in the base constraint table. The trips contribution to the base constraint will then be the number of FA on the trip. For instance, the contribution of the trip with crew complement 0/0/0/0//0/0/2/0 will be two.

It is possible to get more detailed information in the APC status report and

give each crew position a separate section. To do this, turn off *Combine positions within Base Constraints status report* and turn on *use\_pdl\_status* and set *status\_pdl\_file* to

"hidden/BaseConstraints.output".



# Minimum connection

CMS support for evaluating connections between aircrafts is an essential part of the legality used to determine if the connection between two flight legs is too short. In the following text the incoming flight is referred to as arriving and the outgoing flight is referred to as departing. When the system can’t determine a correct connection time, it will default to 99 hours, indicating that there is a problem with the connection.

## Configuration possibilities

The requirements for the different types of connections are specified in three tables: the *minimum\_connection* table, the *coterminals* table, and the *minimum\_connect\_pass* table.

Specifying the connection requirements for connection between flight legs should be done in the table *minimum\_connection*, with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| region | String | Yes | The region for the arriving flight (can be SKD, SKI, SKN or SKS). |
| place | String | Yes | Arrival airport or arrival area. The area can be SCAN, EURO, INTC. Airport will take precedence over place in this lookup. |
| islonghaul | Boolean | Yes | Indicates that the connection time is only valid for arriving long haul flights. |
| arrtype | String | Yes | The area where the arriving flight has been operating. The possible values are D, S, E and I. |
| deptype | String | Yes | The area where the departing flight has been operating. The possible values are D, S, E and I. |
| validfrom | Abstime | Yes | Validity period start time for the connection time entry. |
| validto | Abstime | No | Validity period end time for the connection time entry. |
| cnxfc | Reltime | No | The required connection time for flight deck crew. |
| cnxcc | Reltime | No | The required connection time for cabin crew. |
| trusted | Boolean | No | Indicates if a connecting airport is trusted. If an airport is not trusted, the connection will be checked even if there is no aircraft change. |

The table lookups for a flight will always try to identify the connection by the arrival region, place as arrival airport, arrtype, deptype, if the arrival flight is a long haul and if it is inside the validity period (validfrom and validto). If no connection is identified, the lookup will then be done with place set to the calculated area.

For a connection to/from an active long haul flight and a passive flight, the look up will be done in the table *minimum\_connect\_pass*, with the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| place | String | Yes | Arrival airport or arrival area. The area can be SCAN, EURO, INTC. Airport will take precedence over place in this lookup. |
| Ispass | Boolean | Yes | Specified if the arriving flight is a passive flight. If the value is set to False the entry is valid for departing flights. |
| arrtype | String | Yes | The area where the arriving flight has been operating. The possible values are D, S, E and I. |
| deptype | String | Yes | The area where the departing flight has been operating. The possible values are D, S, E and I. |
| validfrom | Abstime | Yes | Validity period start time for the connection time entry. |
| validto | Abstime | No | Validity period end time for the connection time entry. |
| cnxfc | Reltime | No | The required connection time for flight deck crew. |
| cnxcc | Reltime | No | The required connection time for cabin crew. |
| cnxaplh | Reltime | No | The required connection time for a purser. This will be used when the arriving or departing flight is a long haul flight. |

The table will be used for all connections between long haul active flights and passive flights.

The table *coterminals* defines the connection time between co-terminals. It is defined as fallows:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| airport1 | String | Yes | One of the terminal pairs defining a co-terminal. |
| airport2 | String | Yes | The other of the terminal pairs defining a co-terminal. |
| mincnx | Reltime | No | The required connection time for the co-terminals. |

Lookups in the *coterminal* table will only be done if the arriving airport differs from the departing airport. The lookup will start to test if the arriving airport is specified as airport1 and the departing airport is specified as airport2. If no airport is found, the lookup will be done the other way around.

# Check-In exception

CMS supports exceptions to standard check-in times. It is controlled by the table *ci\_exception*.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Key | Description |
| Maincat | String | Yes | Main category of crew. |
| Position | String | Yes | Position to fly in. DH is deadhead. \* matches any position. |
| Region | String | Yes | Region of the trip. |
| Airport | String | Yes | Airport. |
| Fd | String | Yes | Flight descriptor. \* matches any flight. |
| Validfrom | Abstime | Yes | Valid from date. |
| Validto | Abstime | No | Valid to date. |
| Citime | Reltime | No | The check-in time exception. |
| Si | String | No | Supplementary information. |

# Optimizer Cost function

The optimizer cost function consists of two parts: real costs and penalties. Real costs are the actual costs that come from paying salaries, buying hotels rooms and deadhead tickets for crew. Penalties are fictional costs applied to certain properties of the trips to steer the solution in a direction that is more stable on day of operation, easier to roster, more satisfactory for crew etc.

## Parameters

There are no overall parameters for the pairing optimizer cost function. Each cost element can however have any number of parameters. They are described after each individual cost element. The parameters for the various cost elements are collected in the parameter form under the tab “Trip Cost”. The names displayed in this document are the same that are shown in the parameter form.

## Real costs

This section covers the actual cost of covering the production.

### Cost of production day

This is the cost of having crew on staff during the planning. For each duty day (production day) in the schedule there is a cost for the crew covering that production. The total cost of production day is the number of crew working days in the schedule times the cost for having one crew work one day.

Parameter: Use duty day costs in cost function

Includes or excludes the cost element from the optimizer cost function.

Parameter: Cost of one duty day

This is the cost of one duty day (production day) for one crew

### Cost of deadhead

Sometimes crew have to fly passive, or positioning flights, to get back home from a trip or to be able to start working at a station other than the one they are currently at. The cost associated with this is dependent on whether crew can fly with its own fleet or if another carrier has to be used (OAG flight). The deadhead can also be a ground transport, for example if there are two airports in the same area or when using co-terminals (one station, 2+ airports) taxi or shuttle transport might be used.

Parameter: Use passive (DH, OAG, GT) costs in cost function

Includes or excludes the cost element from the optimizer cost function.

Parameter: Cost of one deadhead hour (own fleet)

This is the cost of one deadhead hour on own fleet

Parameter: Cost of one OAG hour (another carrier)

This is the cost of one OAG deadhead hour

Parameter: Cost of one ground transport hour

This is the cost of one ground transport hour

### Hotel cost

This cost element contains the cost of all hotel reservations that need to be made. If no hotel can be found for the current station, the system will revert to using the default parameter values for that station. Hotel cost is calculated both for duty stops and duty free breaks. For a duty free break that is a day stop the cost is half the cost of a duty stop. For a duty free break that is a night stop the cost is the same as for a duty stop.

Parameter: Use hotel costs in cost function

Includes or excludes the cost element from the optimizer cost function.

Parameter: Default cost of one hotel night

The default cost of one hotel night.

Parameter: Default earliest hotel Check-In time

The default check in time

Parameter: Default latest hotel Check-Out time

The default check out time

Parameter: Default transport time to/from a hotel

The default local transport time (one way)

### Per diem

An approximation of the actual per diem cost used by the optimizer cost function.

Parameter: Use per diem in cost function

Include or exclude the cost element from the optimizer cost function.

### Local transport

Local transport cost is the cost of transporting crew from an airport to the hotel and back. If no data can be found for the current station, the system will revert to using the default parameter values for that station.

Parameter: Use local transport costs in cost function

Includes or excludes the cost element from the optimizer cost function.

Parameter: Default transport cost to/from a hotel (when not using external table)

The default cost of one-way local transport

## Penalties

This section covers penalties that are used to steer the solution in a certain direction. Just as the real cost, penalties also result in a cost of sorts. However the penalties are fictional and the parameter settings need to be balanced with the real costs and among themselves to get an appropriate weighting of the relative importance of cost elements and to achieve the desired result.

### Duty minute

This cost element reduces duty time to give a better duty/block time ratio, which should lead to more effective rostering. It also reduces long breaks and deadheads. It is very simple; it just adds a penalty per duty minute in a trip.

Parameter: Use duty minute penalty in cost function

Includes or excludes the cost element from the optimizer cost function.

Parameter: Penalty per duty minute

Each duty minute in a trip is penalized by this amount.

### Aircraft change

For stability reasons it is desirable to let crew follow the aircraft rotations during a duty. Aircraft changes within a duty should therefore be penalized by a certain amount each time.

Parameter: Use aircraft change penalty in cost function

Includes or excludes the cost element from the optimizer cost function.

Parameter: Penalty for one aircraft change

Each aircraft change within a duty is penalized by this amount.

### Points over 90-buffer

If duty time is close the limit it is more difficult to create legal trips with that duty. The solution is also more unstable, if a delay occurs crew might not be able to fulfil the duty as planned because of the 90-point rule. If the number of duties with close to 90 points can be held at a reasonable level, the optimizer has a better chance of finding a more stable solution faster.

The penalty works on duty level by calculating the squared overshoot over the buffer value and then multiplying this with a cost. Thus when the number of duty points passes the buffer value the penalty for the duty will increase quadraticly the closer the point level gets to 90. The penalty of a trip is the sum of the penalties of the duties.

Parameter: Use 90-point buffer penalty in cost function

Includes or excludes the cost element from the optimizer cost function.

Parameter: 90 point buffer penalty threshold

This is the threshold above which the penalty has effect.

Parameter: Penalty per point (squared) above the 90 point buffer threshold

Each squared point above the threshold value is penalized by this amount.

### Duty free break

A break within a duty between 5 and 8 hours is considered a duty free break. This is not desired and the penalty penalizes a duty with a certain amount if it has a duty free break within.

Parameter: Use penalty for duty free break in cost function

Includes or excludes the cost element from the optimizer cost function.

Parameter: Penalty for having a duty free break in a duty

Each duty free break within a duty is penalized by this amount.

### Layover at other base

In the case of crew (in the particular planning area) having more than one base, it might be that crew have to fly production starting at a base other than their home base. To avoid this as much as possible a penalty can be used to reduce the number of layovers at other bases than home base.

Parameter: Use penalty for layover at other base in cost function

Includes or excludes the cost element from the optimizer cost function.

Parameter: Penalty for a layover at other base than home base

Each layover at a base other than home base is penalized by this amount.

### Check out after midnight on last duty

Late check out before a day off is undesirable, in practice that reduces the use of the day off. A penalty can therefore be used to influence the optimizer to put as many of the late check outs as possible inside trips and not at the end of trips.

Parameter: Use penalty for check-out after midnight on last duty

Includes or excludes the cost element from the optimizer cost function.

Parameter: Penalty for check-out after midnight on last duty

This is the cost of a late check out on the last duty.

### Trip length

Penalties exist for 1-5 day trips and can be used to steer the solution towards trips of the desired length.

Parameters: Use penalty for 1 day trip

Use penalty for 2 day trip

Use penalty for 3 day trip

Use penalty for 4 day trip

Use penalty for 5 day trip

Includes or excludes the cost element from the optimizer cost function.

Parameters: Penalty for 1 day trip

Penalty for 2 day trip

Penalty for 3 day trip

Penalty for 4 day trip

Penalty for 5 day trip

Costs for having trips that are x days long.

### Deadhead in roundtrip with same AC

Deadhead as part of a roundtrip is undesirable since crew does not like to change from working to passengers on the same aircraft.

Parameter: Use penalty for deadhead in roundtrip with same AC

Includes or excludes the cost element from the optimizer cost function.

Parameter: Penalty for deadhead in roundtrip with same AC

Cost of having a deadhead as part of a roundtrip.

### Middle duty deadhead

Deadheads in the middle of a duty are undesirable. The penalty is used to avoid deadheads other than on the first or last leg of a duty. This penalty is applied when the duty has a single middle duty deadhead.

Parameter: Use middle duty deadhead penalty

Includes or excludes the cost element from the optimizer cost function.

Parameter: Middle duty deadhead penalty

Cost of having one middle duty deadhead.

### More than one middle duty deadhead

Deadheads in the middle of a duty are undesirable. The penalty is used to avoid deadheads other than on the first or last leg of a duty. This penalty is applied when the duty has more than one middle duty deadhead.

Parameter: Use several middle duty deadheads penalty

Includes or excludes the cost element from the optimizer cost function.

Parameter: Several middle duty deadheads penalty

Cost of having several middle duty deadheads.

### Max duty in x day trip

Crew do not like to work too much in a trip. To avoid trips that are too long a time limit is set for different length trips. The penalty is applied for each minute that the duty time exceeds the limit.

Parameter: Use exceeded max duty in x day trip penalty

Includes or excludes the cost element from the optimizer cost function.

Parameter: Max duty in 1 day trip

Max duty in 2 day trip

Max duty in 3 day trip

Max duty in 4 day trip

Max duty in 5 day trip

Maximum duty “allowed” in x day trip.

Parameter: Exceeded max duty in 1 day trip penalty

Exceeded max duty in 2 day trip penalty

Exceeded max duty in 3 day trip penalty

Exceeded max duty in 4 day trip penalty

Exceeded max duty in 5 day trip penalty

Cost per minute for exceeding the max duty limit.

### Expensive hotel stations

Some stations have a shortage of reasonably priced hotel rooms leading to high hotel costs if too many crew are planned for layovers at that station.

Parameter: Use expensive hotel trip penalty

Includes or excludes the cost element from the optimizer cost function.

Parameter: Penalty for staying at an expensive hotel station

Penalty applied for each crew with a layover planned at a station in the set of expensive hotel stations.

Parameter: Expensive hotel stations

Set of stations where it is expensive to plan layovers.

### Day stop

Day stop is when a duty in a trip starts the same day as the previous duty ended. Day stops are undesirable and should be avoided.

Parameter: Use day stop penalty

Includes or excludes the cost element from the optimizer cost function.

Parameter: Penalty for day stop

Penalty applied for each crew in a trip with day stop.

### Duty time in duty period

For SKS CC there is a need to limit the duty time in a duty period. There is however no absolute limits to the amount of duty time allowed in a duty period since crew get overtime payment for hours exceeding the limit. A cost has been implemented that gives linear penalty for every minute the duty time exceeds the limit.

When using the cost the rule “Coll: Max duty time in duty period” should be turned off.

Parameter: Use penalty for duty time in duty period

Includes or excludes the cost element from the optimizer cost function. Is off by default since only used by SKS.

Parameter: Penalty for duty time in duty period

Penalty for each minute the duty period exceeds the limit.

Parameter: Max duty time in ordinary duty period

This is the limit duty time for trips that don’t have all legs > 4h. Default value10:30.

Parameter: Max duty time in duty period with all legs > 4h

This is the limit for trips with all legs > 4h. Default value 12:00.

### Duty time in duty period

For SKS CC there is a need to limit the duty time in a duty period. A one-off penalty is applied for duty time > 12:00 hrs if all active legs are charter or if all active legs have a block time > 4:00 hrs. When using the cost the rule “Coll: Max duty time in duty period” should be turned off.

Parameter: Use penalty for duty time, charter or block time > 4h, in duty period [CC]

Includes or excludes the cost element from the optimizer cost function. Is off by default since only used by SKS.

Parameter: Penalty for duty time, charter or block time > 4h, in duty period [CC]

A one-off penalty if the duty period exceeds the limit.

Parameter: Max duty time in duty period with all legs are charter or all legs have block time > 4h [CC].

This is the limit for trips with all active legs > 4h or all active legs are charter. Default value 12:00.

### Duty time in duty period

For SKS CC there is a need to limit the duty time in a duty period. A one-off penalty is applied for duty time > 10:15 hrs if no active legs are charter and no active legs have a block time > 4:00 hrs. When using the cost the rule “Coll: Max duty time in duty period” should be turned off.

Parameter: Use penalty for duty time in duty period

Includes or excludes the cost element from the optimizer cost function. Is off by default since only used by SKS.

Parameter: Use Penalty for duty time, not charter or no block time > 4h, in duty period

A one-off penalty if the duty period exceeds the limit.

Parameter: Max duty time in duty period with all active legs > 4h or all active legs are charter.

This is the limit for trips with all active legs > 4h or all active legs are charter. Default value 10:15.

### Duty time in calendar week

For SKS CC there is a need to limit the duty time per calendar week. A one-off penalty is applied for duty time close to 42:00 hrs / duty in calendar week.

Parameter: Use penalty for exceeding duty time per calendar week [CC].

Includes or excludes the cost element from the optimizer cost function. Is off by default since only used by SKS.

Parameter: Penalty for exceeding duty time per calendar week [CC].

A one-off penalty if the duty time exceeds the limit.

Parameter: Max duty time per calendar week [CC]

This is the limit for exceeding duty time per calendar week. Default value 41:00.

### Extended duties

To avoid potential fatigue problems caused by extended duties there exist two penalties, one to limit total number of extended duties in trip, and one to limit long trips ending with an extended duty.

Parameter: Use penalty for extended duties

Includes or excludes the cost element from the optimizer cost function. Off by default.

Parameter: Penalty for extended duties

Penalty for each extended duty in trip.

Parameter: Use penalty for extended duty last in trip

Includes or excludes the cost element from the optimizer cost function. Off by default.

Parameter: Minimum days in trip to check extended duty last in trip

Minimum number of days in trips that should be checked for extended duty last in trip.

Parameter: Penalty for extended duty last in trip

Penalty for a trip ending with an extended duty.

### Table of regularity

There is a soft cost element that penalizes deviations from the leg connections within the external table. Only on-duty legs within the external table give penalties. Each leg can result in two penalties: one for the connection before the leg and one for the connection after.

Parameter: Use TOR penalty in cost function

Includes or excludes the cost element from the optimizer cost function

Parameter: Penalty for a changed start or end

Penalty that is added to the cost function for each leg connection that previously started/ended the trip and does not do that anymore

Parameter: Penalty for a changed layover to/from On-duty

Penalty that is added to the cost function for each leg connection that previously started/ended a duty as on-duty and does not do that anymore

Parameter: Penalty for a changed layover to/from Deadhead

Penalty that is added to the cost function for each leg connection that previously started/ended a duty as deadhead and does not do that anymore

Parameter: Penalty for a changed day connection to/from On-duty

Penalty that is added to the cost function for each leg connection that previously consisted of an on-duty leg and does not do that anymore

### Soft Locks

There is a soft cost element that penalizes cases where a soft lock has been added as a penalty and the added soft lock is broken within the solution.

Parameter: Use Soft Lock penalties in cost function

Includes or excludes the cost element from the optimizer cost function

Parameter: Standard Soft Lock cost 1…10

When adding a soft lock with 1 through 10 as the predefined value, this is where the mapping between value 1 and the actual penalty amount for that value

## Constraints

This section covers constraints working on the whole solution as opposed to the trip based costs and penalties described above.

### Base constraints

The base constraints functionality is described in the Base constraints functionality and all the parameters are described there.

### Average duty per duty day

If the trips are too condensed, with a very high average duty time per duty day, it will be difficult to create efficient rosters and the result is additional blank or free days. This constraint can be used to bring down the average duty time per duty day to a reasonable and easier-to-roster level.

Parameter: Use max average duty per duty day constraint

Includes or excludes the constraint

Parameter: Max average duty per duty day

This is the maximum average duty per duty day that the optimizer will accept for a solution if the constraint is used.

Parameter: Cost of breaking constraint

Penalty applied when each time constraint is broken.

### Number of “trivselstopp” per base

Trivselstopp is a layover at a nice location. This constraint is used to ensure that the different bases get their share of trips with “trivselstopp”. For a trip to have a “trivselstopp” ist must have a layover after a leg with a flight number from the set of flight considered “trivselstopp”.

Parameter: Use number of trivselstopp per base constraint

Includes or excludes the constraint.

Parameter: Trivselstopp

Set of flight numbers considered “trivselstopp”.

Parameter: External table containing data for trivselstopp

Name of the table that containt information about “trivselstopp”.

Parameter: Cost of breaking constraint

Penalty applied when each time constraint is broken.

Table: trivselstopp.etab

Table that provides information about the number of trivselstopp with a certain flight number that should be allocated to a specific base during a specific time period. It can be replaced with another table without changing the code provided the column names and datatypes are the same.

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Column name** | **Name shown** |
| S | base | Base |
| I | flightnumber | Flight number |
| A | validfrom | Valid from |
| A | validto | Valid to |
| I | num | Number |

### Trips starting and ending in time of day interval

We want to be able to change the home base of a trip for a certain number of trips. To do that we need a number of trips that starts in a specific time interval and ends in a specific time interval so that deadheads can be added first and last in the trip without resulting in an extra duty day. We also want to limit the number of days in the trip.

The constraint uses the base groups SUITABLE and UNSUITABLE to define which trips matches the criteria. The bases for which the base groups are valid is defined in the BaseDefinitions table.

Parameter: Use trips starting/ending in time of day interval

Includes or excludes the constraint.

Parameter: Minimum number of trip days allowed

Minimum number of trip days allowed.

Parameter: Maximum number of trip days allowed

Maximum number of trip days allowed.

Parameter: Start time of first time period

The time when the first period starts. Trip should start in the first period.

Parameter: End time of first time period

The time when the first period ends. Trip should start in the first period.

Parameter: Start time of last time period

The time when the last period ends. Trip should end in the last period.

Parameter: End time of last time period

The time when the last period ends. Trip should end in the last period.

Parameter: Number of trips needed

The number of such trips needed in planning period.

Parameter: Cost of breaking constraint

Penalty applied when each time constraint is broken.

### Number of duty days with same AC-type

Since not all pilots are qualified for all aircrafts we need trips that only contain legs with a certain AC-type. A certain number of duty days in such trips are needed.

Parameter: Use number of duty days with same AC

Include or exclude constraint.

Parameter: Cost of breaking constraint

Penalty applied when each time constraint is broken.

Parameter: AC-types that need a certain number of duty days

Set of AC-types for which trips with only that type should be created.

Parameter: Number of duty days with ac-type table

Table containing information about the number of duty days needed in trips with only a certain AC-type.

Table: ac\_types\_duty\_days.etab

Table contains information about how many duty days in trips with a certain AC-type are required in a certain time period.

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Column name** | **Name shown** |
| A | starttime | Start time |
| A | endtime | End time |
| S | actype | AC\_type |
| I | numdays | Number of days |

### Min and max number of hard duties per day

Used to limit and evenly distribute the number of hard duties.

Parameter: Use global min and max number of hard duties per day

Includes or excludes constraint.

Parameter: Minimum time for hard duty

Minimum time of a duty for it to be considered a hard duty.

Parameter: Minimum number of hard duties per day

Minimum number of hard duties in a day. Used to distribute the hard duties evenly when the max value is high.

Parameter: Maximum number of hard duties per day

Maximum number of hard duties in a day.

Parameter: Cost of breaking min constraint

Penalty applied when each time min constraint is broken.

Parameter: Cost of breaking max constraint

Penalty applied when each time max constraint is broken.

### Max number of trips containing station

Limit the number of trips that contains a certain station. This is typically used for LCY airport.

Parameter: Use max number of trips containing station

Includes or excludes constraint.

Parameter: Max number of trips containing station

The number of trips that may contain the station without getting the penalty.

Parameter: Station that are only allowed to apperar in a limited number of trips

The station to limit

Parameter: Cost of breaking constraint

Penalty applied when each time max constraint is broken.

## Retiming – Costs and Rules

You find all retiming related parameters under the tab **RETIMING** the the Rave Parameter form.

### General parameters

**apc\_pac.consider\_retiming\_of\_flights** [False]

The parameter that switches retiming in APC on and off.

**apc\_pac.retiming\_purge\_dominated\_duties** [True]

If you set this to True duties that probably do not contribute to a good solution are skipped. For SAS it must be **True** if you use more than a few retiming alternatives or have a very small problem.

### Retiming Penalty

#### Use penalty

**Use penalty for retiming** [True]

This parameter makes it possible to switch off all penalties for retiming of legs.

#### Penalty for pure move in time

There are five parameter used to define penalty for pure move in time. See the picture.

Penalty

p3

p5: True / False

p1

Moved minutes

p4

p2

Default values:

**Minimum penalty for pure retiming of leg** (p1) [500]

**This pure retiming and shorter get only minimum penalty** (p2)[0:00]

**Maximum penalty for pure retiming of leg** (p3)[1000]

**This pure retiming and longer get maximum penalty** (p4)[1:00]

**Use quadratic function (else linear)** (p5)[False]

#### Penalties for change of flight length

**Penalty per minute the leg has been lengthened** [-100]

**Penalty per minute the leg has been shortened** [500]

### Retiming Rules

These rules are sometimes needed to avoid that optimization problems become too large for APC.

**Legs in multi-leg flights must have the same retiming** [Off]  
Consecutive legs in a trip from the same multi-leg flight must have the same retiming.

**Limit the number of retiming sequences in a duty** [Off]  
**The maximum** [2]  
A retiming sequence is a number of legs after each other with the same retiming.

**Limit number of separate deadheads in duty** [Off] **Max separate deadheads in duty** [3]

**Do not allow deadheads before and after layover** [Off]

### Legal A/C Turnaround

Aircraft rotations may become illegal because of retiming. In APC global constraints are used to satisfy this. In Studio a rule is used.

**apc\_pac.legal\_ac\_follower\_violation\_cost** [200000**]**

Here you define the cost for making one connection in the ac-rotations illegal in APC.

**European line stations with short ac cnx time** [AMS,BRU,DUB,DUS,GNB,GVA,HEL,LED,MAN,SVL,TXL]

The (complex) Rave code defining minimum connection time for the aircrafts is based on a document the Retiming project got from the Network department in March 2012. No external tables and just one parameter (the one below) is used. All other logic and all limits are hard coded in Rave. The Rave code must be changed to consider any changes.  
**Note:** The report “min\_cnx\_time\_retiming.py” can be used to get full information about the Rave definition. See the Functional Reference Manual for User Reports.

**Ac rotation with retimed legs must be legal (Studio rule)** [on]  
The Studio rule.   
**Note:** There is a figure in the **Custom KPI Report** showing the number of illegal AC-rotations.

### Retiming Regularity

Legs belong to the same retiming regularity group if they have the following in common:

* Flight identifier
* Airports
* Arrival and departure time (before retiming)
* Kind of day (Saturday, Sunday or Weekday)

**apc\_pac.use\_retime\_regularity** [False]

This parameter switches on/off the cost for regularity violations in APC.

**Penalty per change in a regularity group** [10000]

The legs in the regularity group are sorted in time order and then you go from leg to leg (including from the last to the first). For each change in retiming this cost is considered. (This is how APC works).

### Consistent Retiming

Retiming must of course be consistent (the leg must have the same retiming in all trips it is used). In APC an internal GLC cost is used for this. In Studio a rule is used instead.

**Retiming must be consistent** [on]

1. You can’t create inconsistent retiming in Studio planning manually, but APC sometimes create (early) solutions with inconsistent retiming.
2. There is a figure in the **Custom KPI Report** showing the number of inconsistently retimed legs.
3. The GLC excess cost used by APC to get consistent retiming is not possible to modify.

# Performing an optimization

An optimization can be performed either on legs, or on duties/trips. In both cases APC will try to create trips for everything in the window that is chosen as input. Legality only checks trips inside the planning period, which means that there are an infinite number of possible trips outside the planning period, which leads to APC stalling. To avoid this, select only the planning period as input to APC.

# Base Variants

Base variants are used to distribute production over several bases. A base variant is created by adding deadhead flights before and after a trip. The base variants functionality is used by SKN short haul and SKI Flight Deck.

There is a rule that can be used to limit the interval in which a deadhead is allowed to start before and after the original trip when making the base variants. The intervals used by the rule are given by four parameters.

Rule: (CCP) GUI: Base variants time limit

Rule applies only on trips in planning period that have base variants.

Parameter: Max time between dh start and trip start for base variants

First deadhead may not start more than this long before start of original trip.

Parameter: Min time between dh start and trip start for base variants

First deadhead may not start after this long before start of original trip.

Parameter: Max time between dh start and trip end for base variants

Last deadhead may not start more than this long after end of original trip.

Parameter: Min time between dh start and trip end for base variants

Last deadhead may not start before this long after end of original trip.

## SKN short haul

**Method:** Use constraint 7.4.4 when creating the pairing solution to ensure a certain number of trips that are suitable for making base variants. Before creating the base variants turn on the *Base variants time limit* rule and set the interval parameters to the desired times. Create base variants from all trips in pairing solution.

## SKI Flight Deck

**Method:** Before creating the base variants turn on the *Base variants time limit* rule and set the interval parameters to the desired times. Create base variants from original trips.

# Aircraft Rotations

Aircraft rotations are an important basis for building stable trips. Generally it is good to avoid aircraft changes in the crew trips so that the impact of a delayed flight is minimized. To know if there is an aircraft change or not the aircraft rotations (trips) needs to be known, and if they are not already available they need to be built. Aircraft rotations can be built manually using the process below or using scripts to automatically build different types of rotations.

To manually build aircraft rotations, first a local plan has to be loaded and then the following steps have to be done:

1. Load the BuildAcRotations rule set
2. Load a preferred parameter set for which AcRot building method to use
3. Possibly change parameters
4. Dissolve current AC-rotations
5. Build AC-rotations using Planning Tools -> Local Plan -> Build Rotations
6. Possibly loading back the original pairing rules and parameters

The applicable parameter sets for the BuildAcRotations rule set are found in the *ac\_rot* subdirectory. There are five choices: *opus, turn, fifo, lifo* ***and*** *multilegflights*; each of them representing a method for building AC-rotations. OPUS and TURN are SAS methods which takes an etable as input. FIFO and LIFO are built-in algorithms, and Multilegflights just connects multiple-leg flights and nothing else.

The OPUS parameter set has the following settings: Max connection time is 28:00. Default etable file: *ac\_rot/opus\_rot.etab*

The TURN parameter set has the following settings: Max connection time is 28:00. Default etable file: *ac\_rot/turn\_rot.etab*

FIFO, LIFO and Multilegflights have a max connection time of 8:00.

For OPUS (but for none of the others) a parameter is set for connecting only within the same calendar day.

For OPUS and TURN the proper etable has to be put in the place, as indicated by respective Rave parameters, for the methods to work correctly.

Also, if changing the Rave parameters manually, make sure to have only one of Methods 1-4 (rules) turned on (FIFO/LIFO are built-in and do not require any rule turned on).

If above mentioned default parameter settings are satisfactory to you, you may use the scripts to build AC-rotations. The scripts go through every step mentioned above, including loading back the original pairing rules and parameter settings. There are four scripts to choose from:

Command: Planning Tools-> Local Plan-> Build OPUS AC-Rotations

Command: Planning Tools-> Local Plan-> Build TURN AC-Rotations

Command: Planning Tools-> Local Plan-> Build FIFO AC-Rotations

Command: Planning Tools-> Local Plan-> Build LIFO AC-Rotations

# Cabin Crew LH Distribution

When planning long haul cabin crew the trips are divided between the bases STO, CPH and OSL. The distribution of trips is given by a spreadsheet in csv format. A script builds slices from long haul trips according to the distribution in the spreadsheet.

Data: Spreadsheet should be stored as a CSV file in $CARMDATA/ETABLES/TRIP\_COPY\_CSV\_FILES/

The spreadsheet should contain seven weeks of data for each flight.

Input: Input should be a set of unique long haul trips. The script works with a database plan or a dated file plan.

Use: Run script from menu Trip General -> Copy Trips 322

Other:

It is possible to undo the action if needed. Slices are only created if the flight number of the first long haul leg exists in the spreadsheet for the given week.

Possible process:

1. Start with a database plan or a dated file plan with parings for SKI Flight Deck
2. Mark all trips and remove all base variants from the trips or do this step before adding base variants to FC trips
3. Run the 3-2-2 script
4. Select all CC trips with homebase OSL. Most of those will be illegal since they start and end in CPH.
5. Mark all trips again if needed and set the base to CPH using Trip General -> Properties
6. Make sure that the rule “Base variants time limit is on”, see section about Base Variants, and that the interval parameters are set as desired.
7. Create OSL base variants for the trips
8. Select the OSL variant for all trips
9. If some trips do not have a legal base variant the interval parameters for the base variant rule may have to be changed. If necessary the rule can be turned off and base variants created for the remaining trips.
10. Select all trips again and remove all optional variants. Removing the optional variants is important otherwise they can be used by Matador when doing Rostering optimization. This gives trips with the right complement and deadheads added first and last.
11. Tag the trips with area SKN if working in a database. If working on a file plan remember to tag the trips when they are fetched to the database.
12. To do this for additional bases repeat from 4.

# NOP to OAG

When trips that contains OAG legs is imported to a database plan these legs are shown as NOP since they don’t exist in the database leg set per default. A function exists that takes all OAG NOP legs in the window and adds them to the leg set. They can be used both in the trip window and in the roster window.

Use: Trip General - > Create leg set for OAG NOP or

Assignment General -> Create -> Leg set for OAG NOP

# Short/Long Haul integration

Short haul and long haul are planned separately from each other and within different departments. As most of the long haul production departs from CPH, crew from OSL and sometimes STO must be transported to CPH to operate on the long haul flights. To create trips departing from another base than CPH, a deadhead is added, while respecting legality, both before and after the long haul trip. The long haul production is distributed between the three bases and sent to the local planning departments.

In order to use the day before the long haul trip more efficiently than just transferring crew as passengers, the fact that cabin crew have multiple qualifications can be taken advantage of and the short and long haul plans can be merged. This means that OSL and STO crew operate as ordinary crew members on a short haul flight between their home base and CPH (since most of the long haul traffic departs from CPH).

## Process

The system supports a short/long haul integration process, as described below.

1. Select a number of 1-day short-haul trips where an additional deadhead legally could be added to them
2. Lock the remaining trips together in such a way that the short-haul trips are locked together duty-wise (legal to add a deadhead before/after) and the long-haul trip are locked together except for the first deadhead.
3. Optimize the different trips, both short- and long-haul, together so that, as far as possible, the deadheads prior to long-haul trips are exchanged to a 1-day short-haul production day.

In order to keep the crew/trips together, and save as many 1/0/2 short-haul trips as possible for the main crew run, do three different runs for the AH, AS and AP positions respectively, and remove unused short-haul trips between the runs.

## AS/AH handling

Short haul flights don’t use the AS position, but this position is still needed on the long haul flights. Since Planning can’t handle varying position inside a trip the short haul flights should be booked with an AS position. CMS handles this without the legs appearing overbooked::

1. Turn on parameter: *Allow need to be transferred from AH to AS for short-before-long trips.*
2. Slice off an AH position of each remaining short-haul trip and transform them into AS positions.
3. Recalculate crew need: Planning Tools > Local Plan > Recalculate Crew Need. CMS will now handle the short haul flights booked in AS as having need transferred from AH to AS.
4. Optimize the AS position trips.

When Publish is performed in Rostering, the assigned position on the short haul flights will change back to AH and the trip will have varying position in the database.

## Parameter set for optimization runs

For these optimization runs a special parameter set has to be loaded: *pairing/KFL*.

Where KFL = Kort Före Lång. This parameter set is specially tailored for the OSL KFL runs, but can be used by any other area.

These are the features of the KFL parameter set:

* The rule *(CCP) Soft: KFL: DH calendar day before CPH or ARN longhaul duty* is turned on. The rule says that deadheads before a long-haul trip has to start the calendar day before.
* A FIFO square rest penalty is turned on to make the rest time between the previous deadhead/short-haul duty and the long-haul trip, as evenly distributed as possible.
* Since this is block building and other cost factors are not important the duty day cost in substantially increased. This is important since the square rest penalty can get high.
* The deadhead cost is turned off. We do not want the cheapest deadhead but the tightest. I.e. either the last one of the calendar day or a tight deadhead at the end of the short-haul duty. If the deadhead cost is used, the optimizer would chose the deadhead that is 5 minutes shorter and thereby adding some hours of idle time on ground for the crew.
* For the same reason as above, only the tight deadhead search parameters are set for the optimizer.
* The column generator is not suited for this kind of block building. The method General Generation is turned on. And with maximal search width, since it’s a small problem.
* The jumpers method is turned on for the AH runs.

# TOR – Table of regularity

It’s possible to use the appearance of how the leg connections look like at one point in time and imitate these leg connections; this is done via the Table Of Regularity (TOR)-functionality. It helps in the creation of solutions that should be influenced by another set of trips, either from within the same fleet or from another fleet or period.

Use case examples:

* Jumper solution for cabin crew should imitate the main crew solution to such a large degree as possible
* Regularity when rolling-out daily to weekly, or weekly to dated
* Cabin trips should be influenced by how the cockpit trips are built

Daily, weekly or dated solutions may be used as the source for the imitation, while the solution which you are applying the imitation to, must be the same level or higher, i.e. it’s possible to imitate a daily to a weekly plan, a weekly to a dated plan, but not a dated to a weekly plan.

The TOR-functionality works in such a way that when viewing the set of leg connections that should be imitated, the external table TOR.etab is created and saved within the Sub-plan external table directory. When creating the solution and imitating this set of leg connections, a number of penalties are applied when the created leg connections deviate from the saved leg connections within the external table.

The functionality has some support for creating the external table based on a jumper solution and imitating that solution. The support that exists is that all leg connections within the jumper solution will be looked at and imitated, but not necessarily the correct leg connection will be used.

Example: Two jumper leg sequences have been saved to the external table:

Sequence 1: SU1-SU2-SU3

Sequence 2: SU4-SU2-SU5

When imitating this solution the following two leg sequences won’t contribute with a

TOR penalty:

Sequence 3: SU1-SU2-SU5

Sequence 4: SU4-SU2-SU3

## Usage

Below is a brief listing of relevant commands and parameters for TOR.

### Commands

The following commands are available for TOR:

Command: Planning Tools -> Create Tables -> TOR Table

This command will first prompt the user by a form that states *Save the new TOR table as:* and requires the user to enter the name he/she wants the table to have. The table is then saved with the stated name, containing all leg connection within the Trip window.

Command: Planning Tools -> TOR Table Manager…

The table manager for all created TOR tables is started up and lists all tables. A specific table can be activated for use with the current sub-plan by marking the specific table and then clicking the button *Activate for Current Sub-plan*. The table is then copied to the local Sub-plan directory, given the name TOR.etab.

### Visualization and reports

There is a rudob to visualize certain aspects of TOR-functionality and a report to close examine regularity statistics.

Parameter: Show markers to indicate TOR connection changes

Parameter that states if rudobs should be used to indicate leg connection changes

Report: TORAnalysis

The report will give a brief overview of the most important analysis numbers that exist around how well the solution has imitated the leg connections within the external table. See Functional Reference – Reports for details.

### Configuration

There are three general parameters defining the behaviour of the package:

Parameter: Leg changed from on-duty to deadhead is considered an identical leg

A leg connection is not considered changed if a leg only has been changed from on-duty to deadhead.

Parameter: Consider the Duty Type as a part of the leg key

Parameter that controls if the duty type of a leg should be taken into consideration when deciding which legs that are identical.

Parameter: Add this number of days to the dates in the etable

Parameter that defines the number of days the dates in the TOR table should be moved by. The parameter is only considered if the loaded TOR table have been created from a dated plan. The parameter is useful when the leg connections in a dated solution are to be kept in the solution during the next planning period. Default value is 0.

# Soft locks

If some requirements exist on how legs should be connected within a trip, these can be expressed with the Soft Locks functionality. Via the SoftLock GUI some different types of leg requirements can be entered and when the solution is created a rule (or penalty) will make sure that these requirements are fulfilled.

A Soft Lock is always based on one leg (except for the rest time soft lock, which is based on a duty). It expresses constraints for that leg and/or for the connection after/before the leg. A leg is defined by carrier code together with flight number and/or departure-arrival station. Additionally it’s possible to specify if the leg is active or deadhead, as well as the day/date of operation by means of traffic day and validity period.

When adding a soft lock, the possibility will exist to either add it as a rule or as a penalty and then either as one of 10 predefined penalty levels or as changeable value.

The requirements that can be expressed are the following:

* enforce/avoid a base covering a leg
* additional connection buffers before or after a leg
* specific connection time must be within an interval before or after the leg
* rest time must be within an interval before or after the duty
* enforce/avoid a connection to a particular leg, i.e. ”lock” or “prohibit” a leg-pair
* enforce/avoid a leg starting or ending a duty or a trip.

In addition, the following qualifiers can be used:

* the requirement applies only where an aircraft change takes place
* the requirement applies only if the leg is flown with a particular aircraft type
* the requirement applies only on specific traffic days
* the requirement applies only during the specified validity period
* the requirement applies only when the trip is flown from a specific base

## Usage

Below is a brief listing of relevant commands and parameters for Soft Locks. For a detailed explanation of different soft lock types, the GUI and the validation report please see the online Jeppesen Pairing Help.

### Commands

These are the commands used for the Soft Locks functionality:

Command: Planning Tools -> Soft Locks Manager…

Trip General -> Soft Locks -> Soft Lock GUI...

The GUI for administrating the soft locks is started up and lists all the soft locks in the currently used soft lock table.

Command: Trip General -> Soft Locks -> Validate Soft Lock Table

This command creates a simple report that states any problems that might exist within the actual soft lock table.

### Visualization and reports

There is a rudob to visualize legs with soft locks, failed or achieved, and a report to display details on soft locks in a trip.

Parameter: Show markers for achieved and failed Soft Locks

Parameter that states if marked should be used to indicate achieved and failed soft locks

Report: Soft Lock Details

The Soft Lock Details report shows detailed information per leg that soft locks are applied on. The report is available from the Trip Object menu.

### Configuration

Parameter: External table with soft locks

Name of the soft locks external table

Parameter: Use Soft Lock rules

Will turn off the rule part of the soft locks functionality, i.e. if adding a soft lock as a rule that soft lock will then not be applied

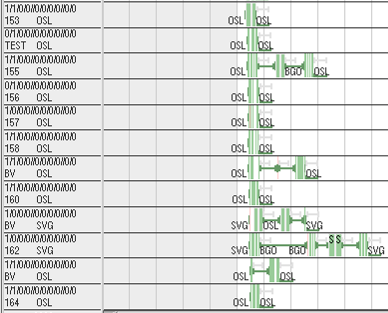
Parameter: Use Soft Lock connection buffer

Will turn off the minimum connection time buffer option within the soft lock functionality

# Trip Naming

Trip naming is used to mark trips in different ways. The methods described below are part of the standard functionality and more information can be found in System Help. Described below are naming using sequence numbers, naming single trips and naming groups of trips.

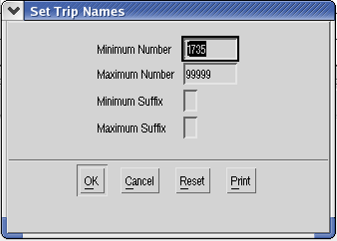
Trip names are shown at the bottom left of the trip margin. Below is an example of a trip window with trip naming made using sequence numbers and after that a group of trips were named BV and a single trip named TEST.



## Naming using a sequence number

One way to name trips is using sequence number with a predefined start number. This gives the sequence number to all trips in the window in the order they appear. All trips get a unique sequence number.

**Command: Trip General -> Set Names …**

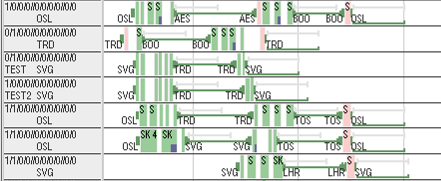


## Naming a single trip

Set a trip name for a single trip using the trip properties.

**Command:** **Trip Object –> Properties -> Trip**

Two trips named TEST and TEST2 are shown in the picture below.



## Naming a group of trips

Several trips can be given the same name using the trip properties. Naming affects all trips in the window.

**Command:** **Trip General –> Properties**

# Retiming Commands

## Replace retiming alternatives with current retiming

Location: **Trip General/Object menu -> Retime -> Modify retiming alternatives**

Set the retiming alternatives to the currently active retiming for all marked legs /current leg.

## Set retiming alternatives...

Location: **Trip General/Object menu -> Retime -> Modify retiming alternatives**

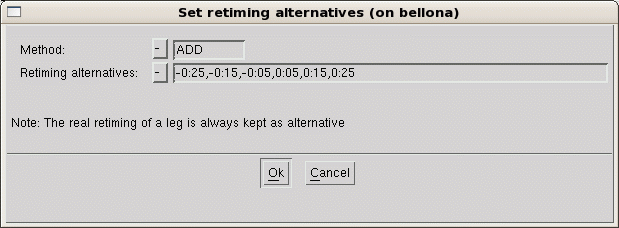
A dialog is shown asking for the retiming alternatives for the marked legs / current leg.

The retiming alternatives are given as a comma separated list of retiming alternatives. A retiming alternative is given on the format (hh:mm,hh:mm).

The retiming alternative is given as pairs of reltime, positive or negative, where each pair defines a shift of the departure time and arrival time.

If the shift is equal for both times, a short form can be used on the format hh:mm.

1. The marked legs can be retimed with the following:  
   10 minutes forward for both departure and arrival, or 5 minutes backward for departure time and non for arrival time. This would correspond to the following string:  
   0:10,(-0:05,0:00)   
   or the string:  
   (0:10,0:10),(-0:05,0:00)



It is possible to define if the new retiming alternatives should be added, replace or be removed from already existing retiming alternatives.

* ADD: The entered retiming alternatives will be added to any existing retiming alternatives found on the selected legs.
* REMOVE: The entered retiming alternatives will be removed, if they already exist on the selected legs.
* REPLACE: The entered retiming alternatives will replace any existing retiming alternatives found on the selected legs.

1. The active retiming will always be kept as one retiming alternative.
2. You can only set retiming alternatives on legs which are active in the sub-plan.

## Merge and spread retiming alternatives in retiming groups

Location: **Trip General menu -> Retime -> Modify retiming alternatives**

The command operates on marked legs.

For each retiming group all existing retiming alternatives are assigned to all members.

1. For the definition of regularity groups see .

## Clean up Retiming Alternatives Table

Location: **Sub-plan**

Removes information from the retiming external table that do not match any leg in the plan and makes sure that all used retiming alternatives are present as alternatives.

The command is needed after commands like **Roll out** and **Move to New Version.**

1. You can access the retiming alternatives table (Sub-plan External Tables/leg\_retiming\_alternatives.etab) from the **External Table Manager**, but normally there is no need for that.

# GUI

## Filters

All filters are defined at the start of the planning period, unless otherwise noted.

Menu items ending with “…” (three dots) brings up a form where more options are available.

Menu items ending with “>” (an arrow-like shape) have a sub-menu.

All standard filters use the planning area as an added criterion.

The filters available on the sub-filter menu work in the same way as the corresponding standard filter, except that they don’t use planning area.

Leg window:

* By Filtering Mask *– Opens a filtering form*

Trip window:

* By Filtering Mask *– Opens a filtering form*
* Optimizer input *- Filters optimizer input*
* Planning area *– Filters trips in planning area*
* Outside Planning area *– Filters trips not in planning area*
* NOP Leg *– Filters trips containing NOP legs*
* NOP Leg incl. other fleet *- Filters trips containing NOP legs, or legs outside the planning area*
* Deadhead *– Filters trips containing deadheads*
* Changes illegal *– Filters trips that has a change marker*
* Not plausible *– Filters non-plausible trips (e.g. trips with different departure and arrival station)*
* Overbooked *– Filters overbooked trips*
* Marked *– Filters trips containing marked legs*
* By Region *(SKD, SKN, SKS, SKL, SKI) – Filters trips belonging to the region, as defined by area tag, or employer if untagged*
* Training
  + SIM and PGT *– Simulators and PGT trips*
  + SIM *– Any simulator*
  + PGT
  + CRM
  + SIM for 2 FC *– Simulators intended for two FC*
  + REC *– Filters recurrent trips for cabin crew*
* Bases *(CPH, STO, OSL, SVG, TRD, BJS, NRT, SHA)*
* Standby– *Filters trips that are standbys of the filtered type*
  + Standby blocks
  + Standby lines (as defined by task code)
  + Scattered standby (as defined by task code)
* Qualification *– Filters trips by their required qualification*
* Shorthaul Production *– Filters short haul trips and standbys*
* Longhaul Production *– Filters long haul trips and standby lines*
* Soft Lock legs (and mark) *– Filters trips with soft locks, and marks the legs*
* Trips with overlapping legs *– Filters trips with legs that are overlapping, due to delays etc.*
* **Retimed legs –** *At least one leg in the trip is retimed*
* **Legs with retiming alternatives –** *One entry for each used retiming alternative in the sub-plan.*
* **Retiming impossible because of hard lock** – *Filter legs with retiming alternatives that can’t be used (in APC) because of hard locks.*
* **Overlap without retiming** – *Filter all trips that must be split before you can remove all retiming.*

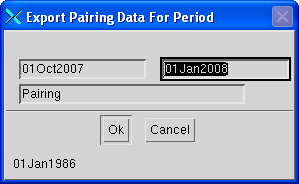
|  |  |
| --- | --- |
| Mark fast “Mark” is a sub menu of the “General popup menu” that enables an easy way to mark legs or trips. Added for SAS:  Trip window:   * **Retimed legs –** *Mark retimed legs* * **Legs with retiming alternatives –** *One menu entry for each used retiming alternative in the sub-plan.* * **Retiming impossible because of hard lock –** *Mark legs with retiming alternatives that can’t be used (in APC).* * **Overlap without retiming *–*** *Mark all legs you must split after before you can remove the retiming.*  |  | | --- | |  | |

# Production estimation

## Calculating production from different sources

The pairing system is an excellent tool to produce accurate estimates of production data like man days block hours etc. This can be done in three different ways depending on the data that is available, the preferred option is to generate pairings but this might be to time consuming, if there is timetable data available that is also enough, although the quality of data is reduced. The third option is to generate the information from a fleet plan, for each of these steps there is a decrease in data quality where pairing data will generated the most accurate data, and fleet plan the least accurate, as more and more estimates is needed. To be able to generate data from pairings some support for generating pairings from a timetable has been added, this means that the system will make calculations on the crew need for each time table leg. The main reason for this is that it is rather difficult to go from the accurate number of block hours to the number of production days the block hours will generate.

The actual export is done by opening the plan which production should be exported and then select **Admin Tools->Export pairing data to Manpower***.* This will show the following dialogue where the two dates define the period which will be exported and the bottom field is the name of the driver to be used when using the data in Manpower.



### Pairings

Once the pairings has been produced there is not much to estimate, hence almost no extra input data is needed. The only extra data needed is how to report mixed pairings where there are different qualifications needed (Cabin), i.e. a pairing consists of both MD 80 and Airbus 320 legs (see table CMP\_ac\_qual\_group). When the pairings are available it is also possible to calculate other interesting values for example the number of long haul flights, number of layovers, number of trips that are over a certain length, this is also rather easy to customize if additional values are needed.

Even though the estimates of the production days are very good they are still only estimates, as some accuracy is lost due to the fact that the pairings has not been rostered. Some examples of this can be when it is still possible to use the rest of the day of a pairing. Say a pairing ends a few minutes after midnight it is probably still possible to use the rest of the day, but at what time is it not possible to add more production. Then there are further complications with the long haul pairings as these pairings include extra rest time both before and after the actual pairing. These times have got to be estimated as there is no way to know these times before the pairing has been rostered.

Normally the production days are calculated as follows the date the trips starts to the date that the trip ends. Say a trips starts august 1:st and ends august 3:d this means that the trip will be 3 days long i.e. 1:st 2:nd and 3:d of august. How ever if the trips ends early the last day it might still be possible to use, so there is a parameter which can be set, called “Time of day when still possible to use crew for reminder of day” the default for this parameter is 0:00 which means that a pairing ending after midnight will generate a extra working day. If there is rest time included in the pairing it will be considered as production days.

### Time table

It is possible to get man day calculations even if there are no pairings available, then one go the other way around and calculate the man days from the block time divided by the productivity for the day in question. This means one has to estimate the productivity for different periods. This is done by entering the productivity values into an etable. It is possible to define different productivity values for each day of the week and for different periods of time, as long as they are not overlapping (*CMP\_productivity.etab*).

The number of crew on each flight leg is defined in an *etable CMP\_crew\_comp.etab*, this table defines a number of criteria that a leg must full fill to have a certain crew complement.

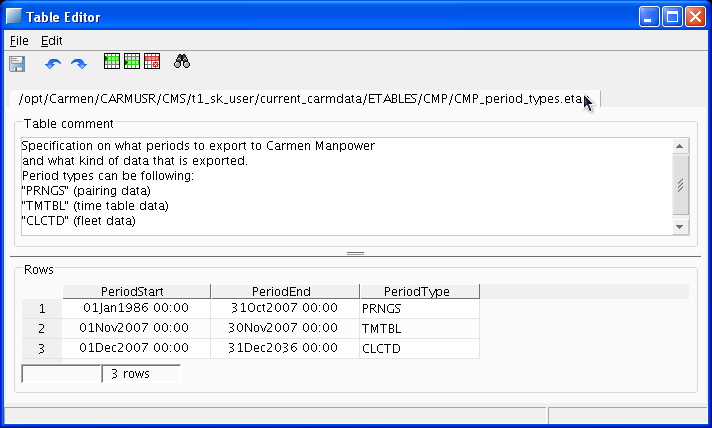
### Fleet plan

If there is no time table available either, it is possible to get data from the fleet plan that is available. The fleet plan is entered into a etable named *CMP\_fleet\_plan.etab*. This etable gives the number of aircrafts that are available and the productivity of those aircrafts, i.e. how many hours per day it is expected to fly. The average crew need of the aircrafts is defined in *CMP\_ac\_crew\_needs.etab*. Another table defines the base distribution of the workload in percent. This table is called *CMP\_base\_distribution.etab*.With all these values it is possible to get an estimate of the block hours per crew category, which is used in the same way as it was for the time table data.

### Etables

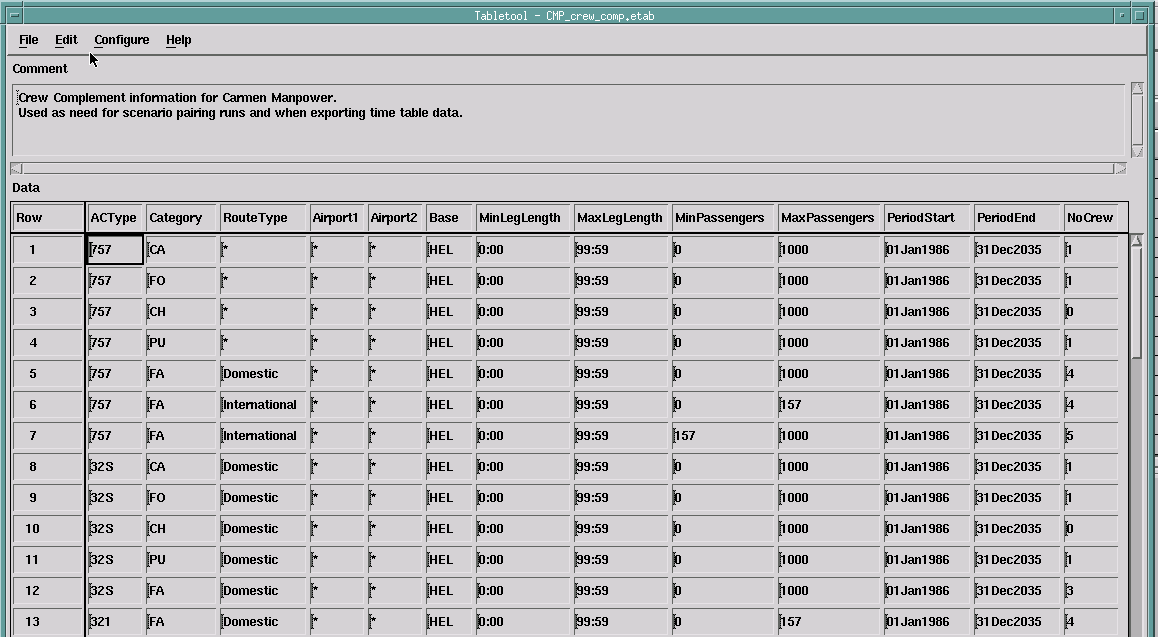
All the following tables are accessible through **Planning Tools->External Table Manager ->CMP Tables**.

**CMP\_period\_types.etab**



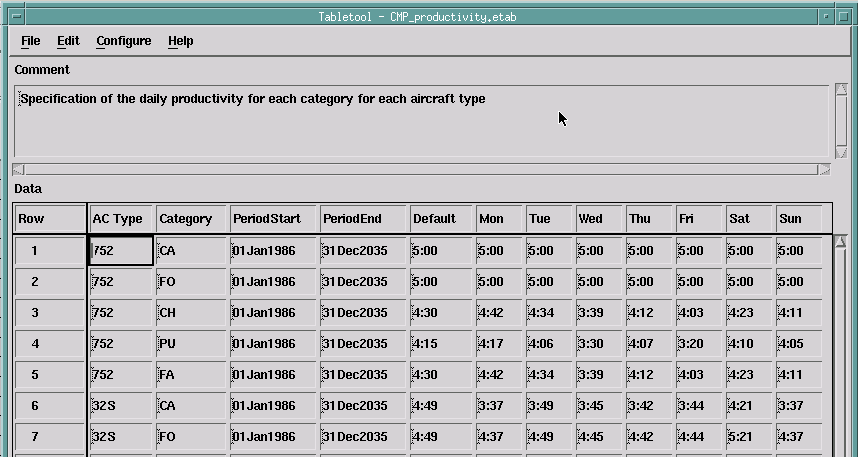
In this table it is defined what kind of data will be used for different periods. There are three kinds of data to use Pairings, Timetable and Fleet plan.

**CMP\_crew\_comp.etab**



This etable is used to get the crew composition and the base distribution for every leg in the plan. Each line of the table defines a specific crew need for the specified crew category for one type of legs. To get a crew need for a leg, one looks at a legs departure and arrival airport, the route type of the leg, the block time of the leg, and the number of passengers on the leg, and on what date the leg starts. With that given a lockup is done in the table with addition to Category and Base which gives a unique crew composition for that leg. Used for timetable period where the individual flight legs are available.

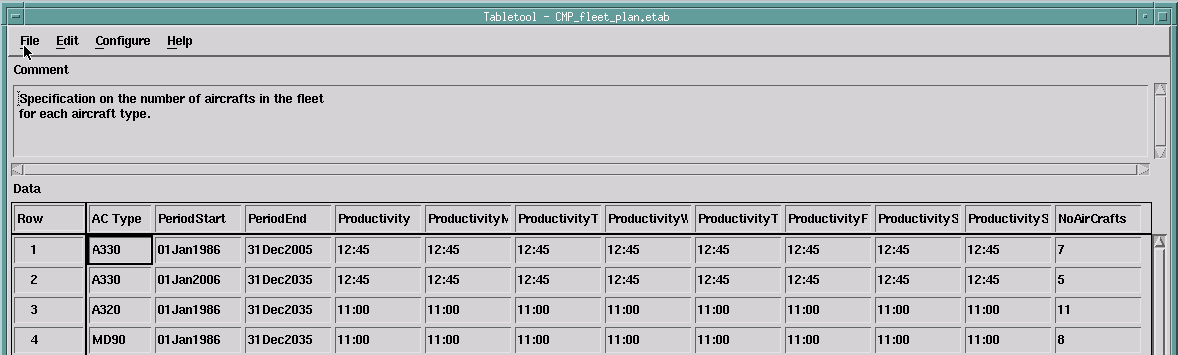
**CMP\_productivity.etab**



Here the productivity numbers are given per aircraft type and category.

Each line states productivity numbers for each day of the week for an aircraft type and crew category combination. Used for timetable and fleet.

**CMP\_fleet\_plan.etab**

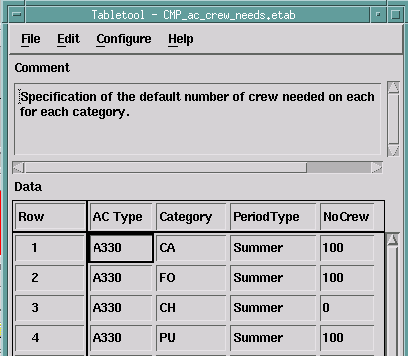
****

Here the number of aircrafts and their productivity is stated.

Each line states the productivity for each day of the week and the number of a certain aircraft type, for a specific period. Whenever there are changes to the fleet a new line has to be added to this table.

Used for fleet.

**CMP\_ac\_crew\_needs.etab**

****

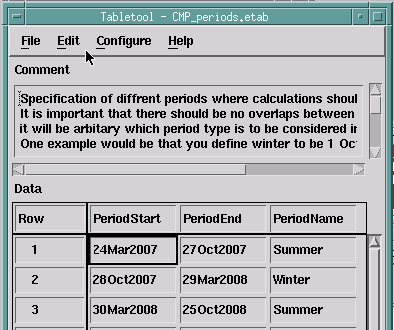
The normal crew composition for an aircraft type is stated here.

Each line gives a crew need for a specific crew category and aircraft type for a specific period.

The table has four columns column one is the aircraft type, the second column states the current category, the third and states period for which the current crew composition is valid, column four states the number of crew multiplied by 100 to be able to give decimal numbers as there is no way of giving a decimal number, the number is divided by 100, once every calculation is made.

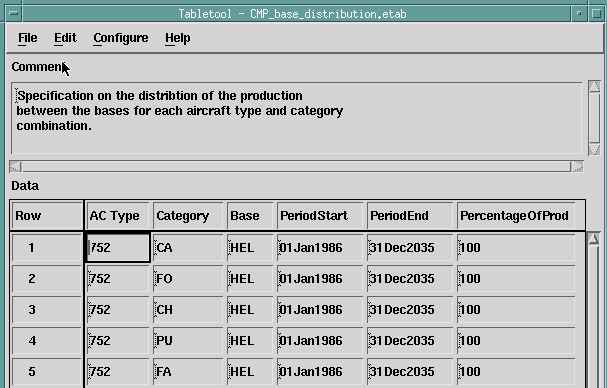
Used for Fleet

**CMP\_periods.etab**



This table defines the different periods used in manpower it is usual that something has got to do with a period for example summer or winter rather than a point in time. This table is used to define these periods new ones can be added, then it is important to add these periods to the other tables where they are used as well.

**CMP\_base\_distribution.etab**



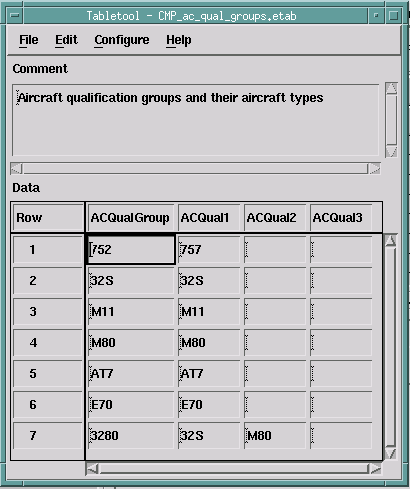
This table distributes the production to different bases, with a certain percentage.

Each line states how much production (in percent) a BCQ in a certain period will get.

This table has got six columns, the first three states the Aircraft type, Category and the Base, column four and five states the period this is valid for, the sixth column states how much of the production will be awarded to the specified BCQ.

Used for Fleet

**CMP\_ac\_qual\_groups.etab**

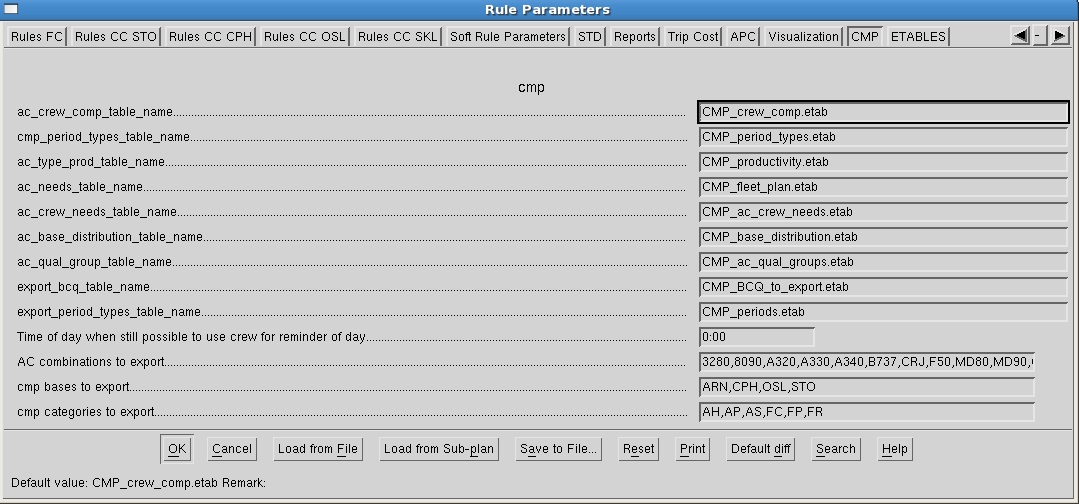


Here the different qualification groups are stated mostly used for cabin. Each line defines a qualification group and the aircraft types that are contained in that group. The aircraft types must be given in an alphabetically sorted order, for example 32S must stand before M11. The table contains four columns where the first column states the name of the qualification group. The columns number two to four states the aircraft types in a sorted order.

Used for pairing.

### Parameters

The CMP Parameters which can be found under the CMP tab in the parameters view.



The first nine fields are the names of the etables to use, the content of those is described in another section. CMP\_BCQ\_to export.etab is not yet used. The next parameter is the “Time of day a pairing including rest must end after for the day to be considered as a production day” described above. Aircrafts types to export are the qualifications that are of interest in the manpower system. Bases to export are the bases that should be exported to the manpower system. The cmp categories to export is the crew categories to export.

## Usage

Step by step guide how to export production data from Jeppesen Crew Pairing.

First of all it is important to distinguish between what kind of data you have got, and for what periods that data is valid. First of all pairing data, this is the ideal data for CMP to work with but is quite hard to get. If we have got pairing data for a period it is important to realize that there will be edge problems with that data. This means in the beginning and end of a period all legs might not be available as they will be flown by pairings that start before or continues outside of the period. This problem has got a increasingly growing effect the longer the pairings are, i.e. the biggest problem is for long haul which has got the longest pairings. This might cause us to cut the period where pairing are available especially in the beginning by as much as 10 days or the maximum length of a pairing. In the end it should be enough to cut with a couple of days.

The other kind of data is timetable data and fleet plan data, here no such consideration has got to be taken.

If one wants to export production to CMP one still have got to change a few parameters and etables. Normally no parameters have got to be changed, and only one of the etables has to be changed. This is CMP\_period\_types.etab where one has to enter what data is available and for what periods that data is available.

But how does one get data in the pairing system? Here I will assume that one knows how to generate pairing solutions and will not describe that further. However if one only have got a ssim file as input and want to generate production from the legs in that file I will give a short description.. So if we have got a ssim file and see how much production it contains, this is what should be done.

1. Open the pairing system.
2. Then select File/New/New Local Plan/From File, this will generate a dialogue. Where you should select the ssim file of interest, one should also click Dated plan so it states yes. One should also give a period of interest, here as a rule of thumb you could enter the period that is stated after the ssim file you selected (Period Info (from file header)). One should also enter a selection of what data should be extracted from the file which aircrafts and airlines, as a rule of thumb one could enter \* and \* in these two fields which will extract all aircrafts and carriers from the file. Finally one should also enter a path to the local plan that one will create, this path should look something like “fleet/period/localplan\_name”, or any other naming convention, but there has got to be two “/” in the name or it will not work. Then press OK, this will generate a local plan. There are also two other tab pages with information that can be changed they do not have to be considered for this, for more information on this step see the pairing manual.
3. Load the local plan, that was generated in the previous step, by file/
4. From the local plan one should now create a subplan with the legs of interest, from the local plan, this is done by local plan/select legs, all legs should normally be chosen here so just press the OK button, if a time has got to be chosen select udop. this will show a new dialog, stating the number of legs that was selected, chose add to supplan. This again will open a new dialog create subplan. Enter a good name for the subplan, and press OK.
5. Load the sub plan that you just created
6. Show legs and from the Legs general menu select create duties/trips and select trips. You will first be shown a dialog for the crew complement, just press OK as this information will not be considered for the production. This will transform every leg in the window to a trip.
7. Check period stated in the CMP\_period\_types.etab by Planning Tools/External Table Manager/CMP Tables (this menu entry is available if you have a file based plan). Enter periods with correct Period Types and then select File/Load/Reload External Tables.
8. Now it is time to export the production to CMP, which will be done by Show trips and then select Export Pairing Data to CMP from the Admin Tools menu. This will open a dialogue where you should enter the time period that data should be exported for. It is also possible to enter a drivername, default name is Pairing. Then press OK, this step finishes the export from the pairing system. Data will be available in the Manpower system after a couple of minutes, depending on the amount of data.

**Warning:**

It is important to understand that only the categories, bases and ac\_quals that are of interest should be entered. Other ways there is a risk that the export will over write other data. Say that the pairings for cabin has already been exported for a period and then the flight deck data shall be exported. If the cabin categories is still shown they to will be exported, which means that they will probably be exported without any data which means they will generate no production at all, which is probably not what is wanted. If you use different names on the drivers exported they will not be overwritten.

# InterBids Trip Export

Trips are shown to crew in Interbids to enable crew to browse trips for the open bid period and show which trips matches each bid.

The trips exported to Interbids are the ones stored in master which:

* Is flight duty
* Not fully assigned
* Not single leg trip

The last condition avoids exporting open time trips generated by Sysmond for Tracking.

The trips are further filtered by Interbids based on crew attributes for the logged in user. See FunctionalReference\_CrewPortal for further details.