Program documentation for the logistics model for Sweden

MICHIEL DE BOK
JAAP BAAK
GERARD DE JONG

March 2026

For Trafikverket

significance
quantitative research

Preface

A logistics module has been developed for both the Swedish national freight model system (Samgods model) and its Norwegian counterpart NEMO. This logistics module introduces logistic elements to the freight model systems, such as the determination of shipment size and the use of consolidation and distribution centres.

This technical document is written for Trafikverket in order to help users to perform analysis with the logistic module. It contains a technical description of the program. The focus is on the logistics model for Sweden. The documentation is not written for programmers in the first place, but mainly for users of the model.

For more information about Significance or this document, please contact Gerard de Jong at:

Significance
Grote Marktstraat 47
2511 BH Den Haag
Netherlands
+31-70-3121533
dejong@significance.nl

Contents

Version history

Starting with version 1.2 of the logistic model, this section contains an overview of the changes made to the model and the paragraphs of this report describing these changes.

Version 1.1.1 (May 2017)

- This version introduces the possibility to specify a group of (neighbouring) nodes that share the same consolidation factor and consolidation volume. Consolidation groups are specified by a new column in the nodes file. This change affects paragraphs 4.5.
- Introduction of a common log file for error messages. This change affects paragraphs 4.5 and 5.5.

Version 1.2 (November 2019)

This version contains a new commodity classification, additional model zones and updated PWC-matrices for 2016 and 2040. The special RCM versions of the BUILDCHAIN and CHAINCHOI programs have been integrated with the standard versions, which has some impact on the control files of these applications. The deterministic ADA approach has not been changed.

Version 1.2.1 (February 2023)

This version of logistic model uses consolidation factors by mode, instead of by sub mode. The possibility has been introduced to use exogenous (inherited) consolidation factors in the model, although the use of endogenous consolidation factors is also still possible as before. Furthermore, the rules for vehicle type availability have been changed. The rule based on total consolidation volume has been replaced by a lower bound for the ratio between the EOQ and the vehicle capacity. The program now also reads a file with availability exceptions.

Version 1.2.2 (February 2024)

Compared to version 1.2.1, no changes to the model functionality have been made in this version. Version 1.2.2 includes 2 bug fixes, related to locked solutions and the use of consolidation factors in LP2CC.

Version 1.2.3 (March 2026)

This version does not include any methodological changes to the model software. Only two bug fixes are included:

- When reading the pwc-matrix to set the typical shipment size, the last origin zone was omitted by the BUILDCHAIN-program. This has been fixed to include the last origin zone.
- When modelling rail capacity management (RCM), zone indices were out of sync in the BUILDCHAIN-program when origin zones were missing in the RCM-input file. This has been fixed.

CHAPTER 1 Introduction

1.1 Introduction

This document contains a technical program documentation of the logistics model, as developed for Sweden. This chapter gives an overview of the logistics model, version 1.2.2. The structure of the model is discussed, the assumptions behind the model and the input and output information is given. This information is useful for those wishing to use the logistics model that is part of the Swedish national freight model system.

1.2 The logistics module

The logistics model is developed for the Work Group for transport analysis in the Norwegian national transport plan and Trafikverket in Sweden. Actually there are two logistics models with the same structure, one for Norway and one for Sweden. As most freight transport models, the existing Swedish national freight model system (Samgods model) and its Norwegian counterpart (NEMO) are lacking treatment of logistics choices, such as the determination of shipment size, consolidation and distribution. Both in Norway and Sweden a process to update and improve the existing national freight model system is going on. An important part of this is the development of the logistics model.

In 2005/2006, a prototype version of the logistics model was developed, that uses Swedish and Norwegian network and cost data and data on the locations of terminals and distribution centres (at these "nodes" there are transfers between different types of vehicles/vessels and/or between different sizes of vehicles/vessels). The parameters of this prototype were not calibrated to observed data. The purpose of this version was mainly to show the feasibility of the approach.

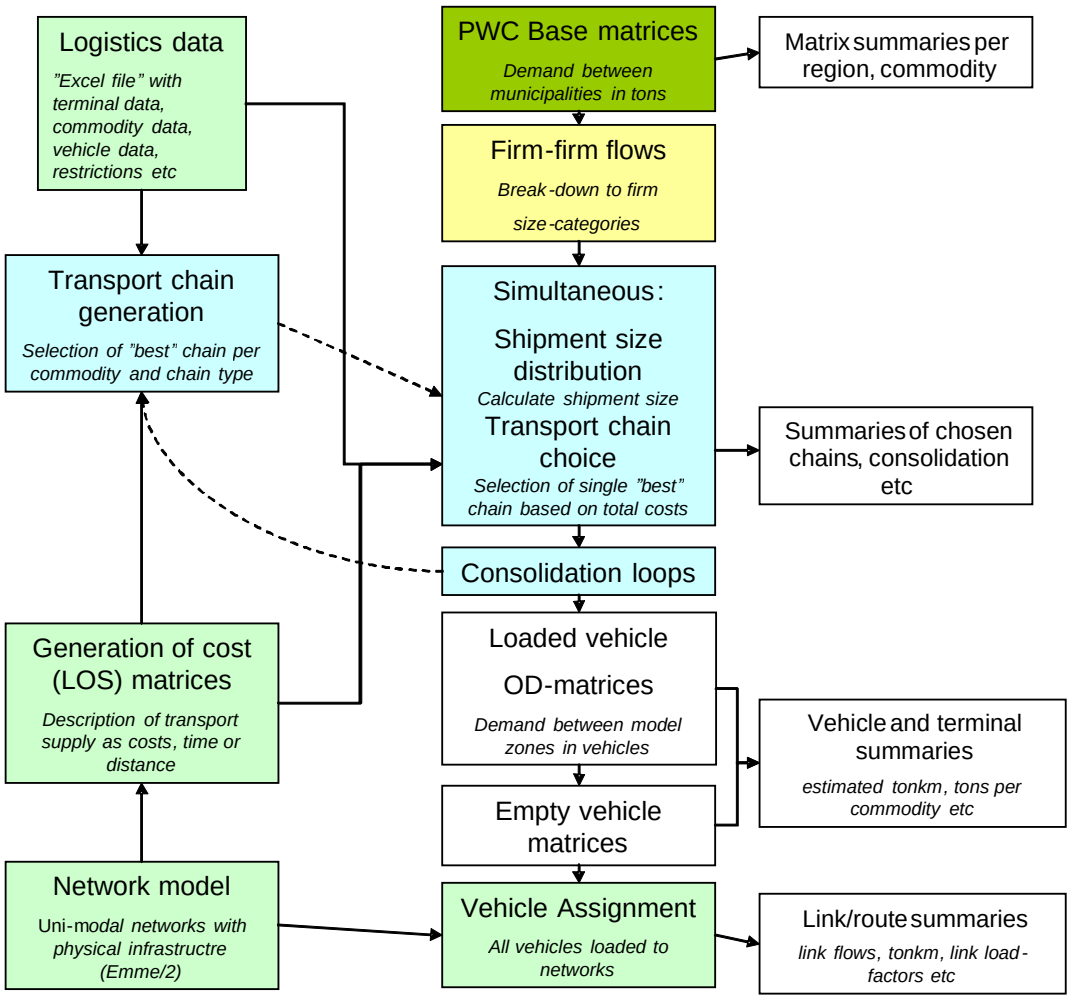
After having shown that the approach was feasible, a new and extended version of the logistics model has been specified and applied for both Norway and Sweden within the framework of their national freight transport forecasting systems.

The basic mechanism in the transport chain choice is minimisation of a deterministic total annual logistics cost function. Estimation of a random utility-based logistics model on disaggregate data (partly available, partly still to be collected) is foreseen for future years for both countries. We have developed a procedure to calibrate parameters in the cost function to available aggregate data

(observed origin-destination flows by mode and commodity type for aggregate zones).

The structure of the national freight model system (either for Sweden or Norway) as a whole is given in Figure 1-1. The logistics model is only part of this. For Sweden the logistics model consists of all the blue boxes in the figure. For Norway, the logistics model is defined as the blue and yellow boxes.

Figure 1-1: Structure of the national freight transport model system with the logistics model (source: John McDaniel, VTI)



Inputs to the logistics model (green, red)

For both countries, the logistics model uses inputs on transport costs, based on time and distance calculated on transport networks, using a network model. Other inputs for the logistics model consider the locations and other attributes (e.g. which commodities can be handled, terminal time and cost) of terminals, commodity-specific costs (e.g. order and storage costs), vehicle data (e.g. capacity and which vehicle types can be used for which commodities) and

restrictions (e.g. on draught in ports). These input data are used together with the transport costs to calculate the total logistics costs. The final type of input of the logistics model refers to the PWC (production-wholesale-consumption) matrices. These are commodity flows from the production zone to the consumption zone (PC flows), also containing flows from production to wholesale and from wholesale to consumption. For Norway, the logistics model takes the zone-to-zone (z2z) PWC flows (base matrices) and in a first step allocates these to firm-to-firm (f2f) flows, which are flows from the sending firm to the receiving firm. These flows may consist of several origin-destination (OD) flows, because a transport chain may be used with multiple modes and/or vehicle types and one or more transshipments along the route. In Sweden there is an external program to generate the f2f flows, which is the last step of the base matrix calculations (Edwards, 2019).

Components of the logistics model (blue, yellow)

The Norwegian model takes as inputs commodity flows from production to consumption zone (that also include the wholesale function). The logistics model then disaggregates these PWC flows to f2f flows. For Sweden the disaggregation to f2f flows is done as part of the base matrix calculation, outside the logistics model, and the logistics model takes these as given.

After this disaggregation, for both countries, the logistics decisions (shipment size, use of consolidation and distribution centres, mode and vehicle/vessel type and loading unit type choice) are simulated at this f2f level (micro-simulation).

In the logistics model there is a choice between a number of transport chains (with one to four legs and with different aggregate modes and different vehicle/vessel types for each leg). The transfer locations between the legs are determined in a separate step: the transport chain generation program (on the left hand side in Figure 1-1). The transfers can take place within the road system (e.g. from van to truck), or between road, sea, rail and air transport (and the Swedish model also includes transfers within the sea and within the rail system: feeder and main haul transport). The transport chain generation program produces the choice set of available transport chains, but already optimises the locations of transfers within each available type of transport chain.

Then the logistics costs optimisation in the main program (labelled ‘simultaneous shipment size distribution and transport ChainChoice’) will determine the optimal shipment size, together with the transport chain choice from the choice set of available chains.

We define consolidation centres as locations where goods are transhipped (and possibly stored, but this is not modelled), with small loads getting in and larger loads getting out. Distribution centres are locations where goods are transhipped (and possibly stored), with large loads getting in and small loads getting out. Both consolidation and distribution centres exist not only in road transport, but can also be ports, airports or rail terminals. Each terminal can serve as both consolidation and distribution centre. Whether a shipment can be consolidated with other shipments depends on the amount of cargo shipped from this consolidation centre to the same destination zone as the shipment under consideration. This implies that consolidation is a function of the OD flows, which are endogenous in the logistics model. We solve this endogeneity problem,

and determine the degree of consolidation (or the load factor of the vehicles) between consolidation centres and distribution centres, by using the logistics model in an iterative procedure ('consolidation loops' in Figure 1-1). This starts with an assumed average consolidation factor for each link, but in a subsequent iteration includes information on the availability of other cargo (based on the available transport chains and port statistics), and in an even further iteration uses the OD flows between consolidation centres predicted in the previous model iteration. The predicted consolidation factor for each link is included in the output of the `ChainChoi` procedure, see section 4.3. Consolidation only takes place within a commodity group. The choice of starting factor has a small but not negligible influence on the final results. To solve this, one could do more than three iterations.

Consolidation in the model can take place at all transfer nodes (terminals, collection/distribution centres) that are included in the model system and along the route ("milk round" or "collection round": a road vehicle goes to several senders in the same zone to collect loads before it goes to the receivers). This implies that consolidated modes are not available for the first and the last legs of a transport chain, because these legs do not start/end at a terminal-node, but at a zone-node instead. To avoid this restriction the extra heavy lorry modes (X and c) are treated as unconsolidated modes within the logistic model. There are two exceptions to this rule:

1. Abroad consolidated modes can be used as first or last leg of a transport chain.
2. If the direct access flag is set for a zone node, consolidated modes can be used for access/egress to/from this zone.

It should be noted that the option to treat all lorry modes as consolidated modes, that is available within the logistic model, does not affect the above logic, it only affects the cost calculations.

Consolidation factors can be determined within the logistic model, but since the introduction of version 1.2.1 it is also possible to use endogenous (inherited) consolidation factors that are supplied to the model via an input file.

Outputs of the logistics model (white)

The output of the model consists of flows between origins and destinations (OD-level), where consolidation and distribution centres (including ports, airports and railway terminals) are also treated as origins and destinations. Furthermore, the model can provide information on total logistics cost (by costs item) between zones, which can be used in trade models or spatial interaction models. The core output of the logistics model is highly disaggregate: at the level of individual f2f flows it gives the shipment size and shipment frequency, the chosen transport chains (number of OD legs, vehicle/vessel type on each leg, transhipment locations) and the logistics costs (by cost item).

Once we have the outputs at the level of the individual f2f flows, these can be aggregated in different ways. The logistics model includes the possibility to produce OD matrices of tonnes and tonne-kilometres by mode and commodity type (but also more aggregate –non-OD- statistics on the total number of tonnes and tonne-kilometres by mode and commodity type), as well as OD matrices of

loaded vehicles by vehicle/vessel type, and of loaded and empty road vehicles. These can be used in assignment to the networks, applying the same network model as for the transport cost inputs. Also one can obtain output flows by vehicle type or by terminal. The procedures to calculate these outputs have remained the same as versions 0 (see RAND Europe and SITMA, 2005). This also goes for the calculation of empty vehicles flows. Furthermore, total logistics costs per PWC flow can be produced, for use in the future determination of PWC flows.

Testing of the program

Tests of the program have been carried out by the client groups as well as Significance, looking at the overall mode split, the number of f2f relations and the number of shipments per commodity type, the OD flows and the outcomes for specific PWC relations. The outcomes have been checked against observed data and/or for consistency/plausibility.

1.3 Overview technical documentation

This document contains a technical documentation on the implementation of the logistics module. Chapter 2 gives a brief description on how the model can be run on a desktop computer. Chapter 3 explains the structure of the model, how the main run parameters of the program can be set and how the structure can be modified. Chapter 4 describes the BuildChain procedure. An overview is presented of the input and output files, and it is discussed how alternative options are set for this procedure and how this influences the results. Chapter 5 describes the ChainChoice procedure. An overview is presented of the input and output files, and it is discussed how alternative options are set for this procedure and how this influences the results.

CHAPTER 2 User manual

2.1 Introduction

This chapter describes how the program files and input and output files are set on a desktop computer in order to make calculations with the logistics module. Furthermore it is outlined how the model is run. For a detailed description of options to define scenarios or modify model settings, see the more elaborate descriptions on the model structure and individual components in Chapters 3 to 5.

2.2 Installing the logistics module

The logistics module can be run on a desktop computer without any additional software installed, although most users will use Citilabs CUBE software package to run the model. As an indication for the required hardware, on a desktop computer with a 3.4 GHz Core i7 processor and 8 GB of internal memory, the run time of the Swedish logistic model was 5 hours, but this is greatly influenced by the settings in the model. For instance, the Norwegian model was considerably faster, about 2-3 hours, since it includes much less chain types compared to the Swedish model.

It is required to have enough free space available on the hard disk of the computer. The program and input files require 410 MB of free space. To apply the model, each completed simulation requires at least 5,8 GB of free space.

The programs and files that are part of the logistics module can be copied to any directory that is designated to perform model runs. For instance: `D:\Samgods\`. If all programs and files are copied to this folder correctly, it should have the following files and subfolders available in the structure that is shown below.

Table 2-1: Overview of the folder structure and main program

Files and folders	Description
\LOG	Contains the log files per commodity for a whole model run (all iterations of BUILDCHAIN and CHAINCHOI)
\BUILDCHAIN	produces a set of potential chains. Folder contains the BUILDCHAIN program file and all required control files.
\BUILDCHAIN\Output	Contains the output files of the BuildChain module.
\CHAINCHOI	Ranks the transport chains determined by BUILDCHAIN based on costs and assigns fractions to the best and second best transport chain. Folder contains the CHAINCHOI program file and all required control files.
\CHAINCHOI\Output	Contains the output files of the ChainChoi module.
\CHAINCHOI\Output\CoVo	Contains the consolidation output files of the ChainChoi module.
\EXTRACT	generates vehicle demand matrices. Folder contains the EXTRACT program file and all required control files.
\EXTRACT\output	Contains the output files of the Extract module.
\INPUT	Contains all input files. These input files are stored in the following 4 subfolders:
\INPUT\COST	contains input files with general control data parameters
\INPUT\GENERAL	Contains file with calibration factors
\INPUT\LOS	contains input files with network link cost matrices. These files are delivered by Samgods from EMME/2
\INPUT\LOCKED	Contains input file with the definition of locked transport solutions
\INPUT\NODES	contains input files with a list of nodes and characteristics of each node, including transfer restrictions
\INPUT\PWC	contains demand matrices in tonnes between zones
commodity.bat	batch file to run all submodules for one specific commodity

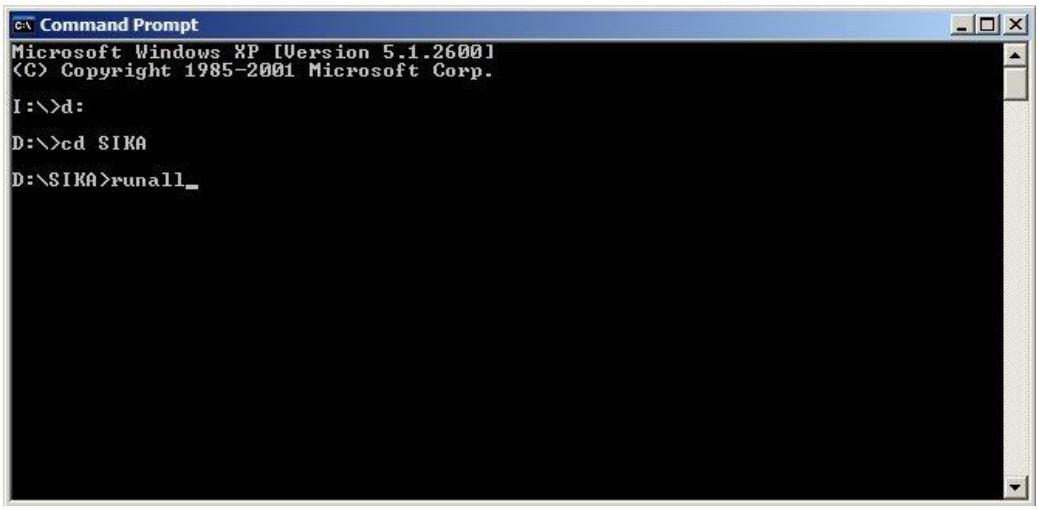
`runall.bat`	batch file to make a model run for all commodities

2.3 Running the logistics module

To make a full run

The logistics module is developed as a console application that is run by starting up the batch job `runall.bat`. This batch job can be run through Windows Explorer, or on a command line. To run the model in Windows explorer, double click the file `runall.bat`. To run the model from the command line, open a console and navigate to the directory where the logistics module is stored, for instance: D:\SIKA. On the command line type 'runall' and press <enter>. Next the logistics module is executed in the console, as visualised in Figure 2-1.

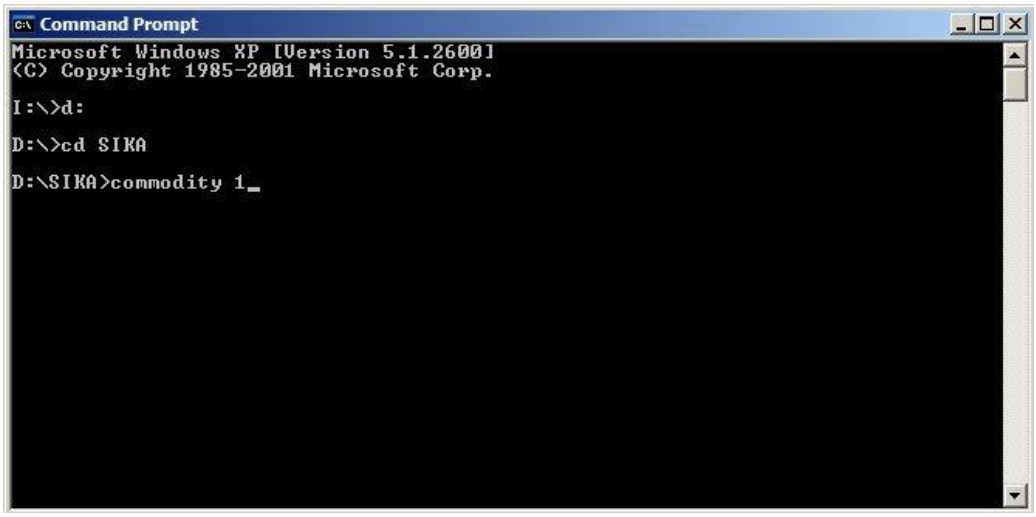
Figure 2-1: Running the logistics module from the command line.



To make a run for one commodity

The logistic module can be run for an individual commodity type as well. To run the model for a specific commodity, open a console and navigate to the directory where the logistics module is stored, for instance: D:\SIKA. On the command line type 'commodity' with an additional argument for the commodity type, and press <enter>. See where the model is run for commodity type 1 only. See Table A-1 in the appendix for an overview of commodity types.

Figure 2-2: Running the logistics module for commodity type '1' from the command line.



Controlling and analysing runs

The model settings can be modified through the input files in the `\INPUT` directory, or through the control files that are used to run subprograms. The following section will discuss some typical examples of changing the input data for a model run to test policy options. A detailed description of all input files, how these files can be modified, and what effects can be expected from these changes, is provided in Chapter 4 and Chapter 5 that describe the `BuildChain` and `ChainChoice` procedures.

After a model run, all required output files need still to be extracted from all calculation results with the `EXTRACT` program. To generate the required output files, the batch job `extractall.bat` should be run.

2.4 Changing input data to implement scenarios

Policy options can be analysed by changing the input data and running the model. Here five examples are given of the implementation of typical policy options.

Adding direct access from a node

The nodes in the logistic model consists of zones and terminals. Changing the direct access settings consists of two steps: changing the nodes file, and checking and changing the direct access file(s). Both steps vary whether the terminal already provides direct access to another zone or if it is new in providing direct access. The two steps are explained in more detail below.

First, add direct access from a zone *i* to a terminal *t*:

1. Open the nodes file[1] (\INPUT\NODES\NODES.TXT)
2. Insert the zone number *i* in the 3rd column of the record for terminal *t* .[2]
3. Close input file and save changes.

Second, check and/or modify the direct access settings in the direct access files (by mode or commodity group):

4. Depending for which transport mode direct access is provided, open the relevant direct access file[3]. For instance for direct access for sea: \INPUT\NODES\DIRECTSEA.TXT[4]
5. Identify the terminal by its node number in the first column of the file.
6. If it already exists: check if the direct access for each commodity group are set correctly.
7. It the terminal doesn't exist in the direct access file:
    - add node code and name of terminal to first two columns.
    - place commodity numbers in the following columns if direct access is provided for that commodity. For instance: to add direct access for commodity 2, add value '2' to the fourth column.
8. Close input file and save changes.
9. Repeat steps 4 to 8 for each relevant transport mode.

Add a terminal

Adding a terminal involves the introduction of a new node to the network. In addition to changing the node files, all network matrices need to be updated too. To add a terminal to the model the following steps should be followed:

1. Open the nodes file (\INPUT\NODES\NODES.TXT)[5]
2. Add a new node record to the file under the node corresponding to the zone in which it is located, preserving the sorted order of the nodes. Insert the new node number in the first column[6].
3. Place a zone name in column 2, starting with the mode type of the terminal, e.g.: "Road: ….".
4. If the terminal provides direct access to existing zones, place the node number of the zone that has direct access to this terminal in column 3.

---

[1] For a description of the node file see Table 4-6

[2] Please note that multiple terminals can provide direct access for the same zone. In these cases, the node number of the zone returns more than twice in the 3rd column in the nodes file.

[3] For an overview of all direct access files see section 4.2

[4] It is suggested to open the file in excel so that the data is put into columns according with the format. The tab delimited file format should be maintained when saving the file.

[5] It is suggested to open the file in excel so that the data is put into columns according with the format. The tab delimited file format should be maintained when saving the file

[6] The new node number need to correspond to the node numbers in the EMME/2 network.

5. Fill in the attributes for the respective node. For a description of the node attributes see Table 4-6.
6. Close input file and save changes.
7. *EXTERNAL TASK:* Recalculate EMME matrices for network with new node. For an overview of the LOS matrices for distance, domestic distance, time and extra costs by vehicle type see Section 4.2.
8. Update all LOS matrices (in the folder: `\INPUT\LOS\`).

Add a zone

Adding a zone is very much similar to adding a terminal. It involves the introduction of a new node to the network, and changing the node files and all network matrices. To add a zone to the model the following steps should be followed:

1. Open the nodes file (`\INPUT\NODES\NODES.TXT`).
2. Add a new record to the file preserving the sorted order of the nodes. Insert the new node number in the first and the third column.
3. Place a zone name in column 2, starting with: “Zone: ….”.
4. Fill in the attributes for the respective node. For a description of the node attributes see Table 4-6.
5. Close input file and save changes.
6. *EXTERNAL TASK:* Recalculate EMME matrices for network with new node. For an overview of the LOS matrices for distance, domestic distance, time and extra costs by vehicle type see Section 4.2.
7. Update all LOS matrices (in the folder: `\INPUT\LOS\`).

Adding a new logistics chain

A new logistic chain can be added by changing the lists of all logistic chains. The following steps can be followed:

1. Open the chain type list (`\INPUT\CHAINTYPE.LIS`)
2. Insert the character combination for the new chain type.
3. Close input file and save changes.

Chain types with either a roro connection at the begin or end of the chain, or a roro connection with different transport modes on either side, are not accepted by the model.

Adding a vehicle type within a mode

The settings for vehicle type and mode combinations are controlled in three different sets of files: the vehicle data files, and the control files for the `BuildChain` and `ChainChoice` modules. The following steps can be followed in vehicle type v need to be added to mode m:

1. Check if the vehicle type is defined in the cargo specific parameter files (`\INPUT\COST\VHCLS_GEN_CARGO.TXT, VHCLS_DRY_BULK.TXT VHCLS_LIQ_BULK.TXT`)
2. Add the vehicle type number for *v*, to the list after mode parameter `VHCL`*m* for mode *m* to the controlfile `BuildChain%1%.ctl` for commodity group *%1%*.
3. Close control file and save changes.
4. Repeat step two and three for each commodity group that has an additional vehicle type available within a mode.
5. Add the vehicle type number for *v*, to the list after mode parameter `VHCL`*m* for mode *m* to the controlfile `ChainChoi%1%.ctl` for commodity group *%1%*.
6. Close control file and save changes.
7. Repeat step five and six for each commodity group that has an additional vehicle type available within a mode.

Enforce the use of a specific transport chain

If the analyst has complementary knowledge about a particular logistic choice that the model does not encompass, the use of a specific transport chain can be enforced by adding a locked transport solution:

1. Open the locked transport solutions file (`\INPUT\Locked\LOCKED.DAT`)
2. Insert the locked transport solution
3. Close the input file and save changes.

The nodes specified in the locked transport solutions file determine the route that will be used, they do not define real transfer terminals.

CHAPTER 3 Program structure

3.1 Introduction

This chapter will first describe the structure of the model and how this structure is implemented in a program. It is explained what sub-programs the program calls upon (`BuildChain` and `ChainChoice`) and what type of input is required for each component. Next it is explained what run-parameters can be controlled in the main program, and how the iterative procedure can be modified. Appendix B gives an overview of all parameters and variables in the model. This overview describes the parameter or variable, it specifies in which file it is stored, by which module it is used, and the type and range of information (if applicable).

3.2 Program structure

The logistics decisions (shipment size, use of consolidation and distribution centres, mode and vehicle/vessel type and loading unit type choice) are simulated at the firm to firm (f2f) level (micro-simulation). The Swedish logistics module uses firm to firm (f2f) commodity flows as input. The initial PWC commodity flows (from production to consumption zone that also include the wholesale function) are disaggregated to f2f flows as part of the base matrix calculation outside the logistics model.

To perform a run with the logistic model, two batch jobs are used. First, `RUNALL.BAT` is used to run the logistic modules for each commodity successively (for the content of this file see Figure 3-1). For Sweden, 16 commodity types are distinguished. An overview of these commodity types is given in the appendix.

Figure 3-1: Contents of [Runall.bat].

```
call commodity 1
call commodity 2
call commodity 3
call commodity 4
call commodity 5
call commodity 6
call commodity 7
call commodity 8
call commodity 9
call commodity 10
call commodity 11
call commodity 12
call commodity 13
call commodity 14
call commodity 15
call commodity 16
```

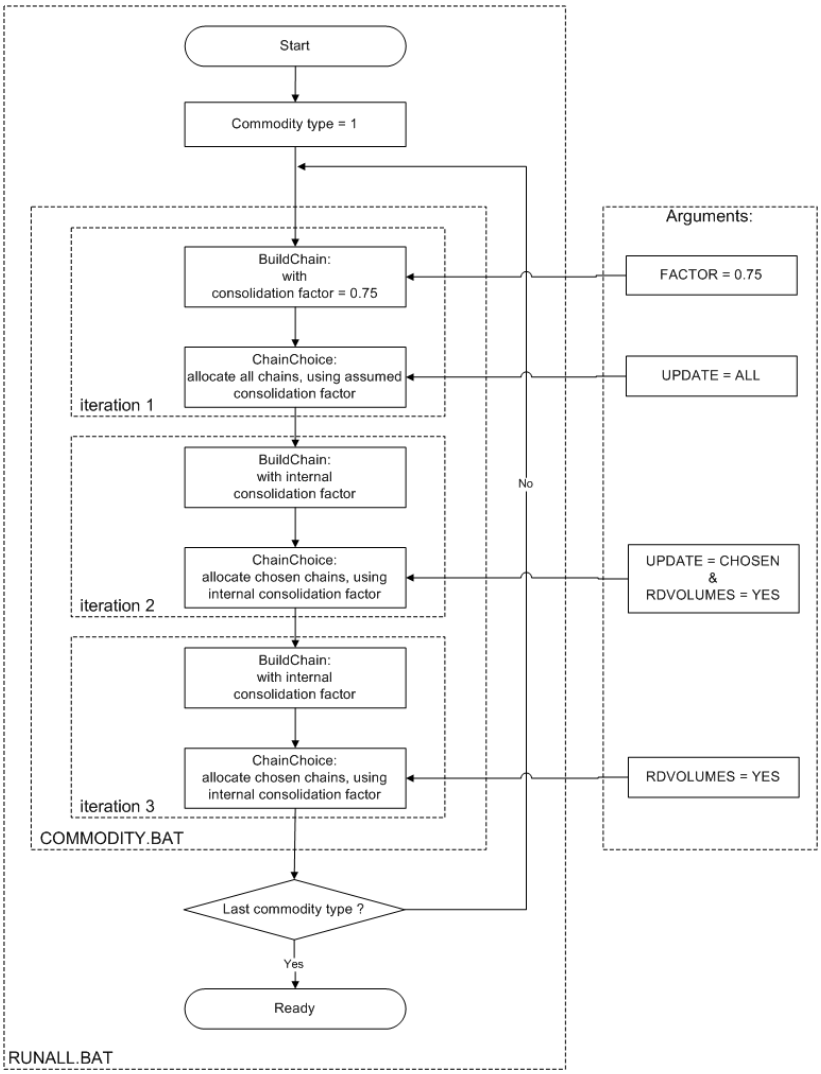
Next, the main procedure `RUNALL.BAT` calls upon the batch job `COMMODITY.BAT` for each commodity group (for the content of this file see Figure 3-3). This logistic model calls upon two logistic components in an iterative procedure, as visualised in Figure 3-2. These two procedures are:

- `BuildChain`: selects the "best" chain per commodity and chain type
- `ChainChoice`: calculates shipment size and ranks the transport options determined by BUILDCHAIN based on total costs, and assigns fractions to the best and second best transport chain.

The ChainChoice program is in fact followed by a small program ConsolidRateMode that is controlled by the same control file as the one that controls the ChainChoice program. It is executed in the same directory as ChainChoice and reads and then overwrites some of the ChainChoice output. It is not identified as a separate program in Figure 3-2, and will be discussed in the section about the ChainChoice procedure given below.

Each component will be discussed in detail in Chapter 4 and 5. The function of the modules and how they are used by the logistics program in an iterative procedure is outlined in Figure 3-2. This iterative procedure is discussed in more detail in section 3.3, but first the function of BuildChain and ChainChoice modules is explained.

Figure 3-2: Structure of the logistic program



The `BuildChain` procedure

The `BuildChain` module selects the "best" chain per commodity and chain type for each origin – destination pair. A new feature introduced in version 1.1.3 of the Samgods model is the possibility to generate chains with alternative, second best, transfer terminals. The module first simulates the choice of the optimal transfer locations for a number of transport chains (with one to four legs and with different aggregate modes for each leg). Transfers can take place within the road system (e.g. from van to truck), or between road, sea, rail and air transport. The Swedish model also includes transfers within the sea and within the rail system: feeder and main haul transport. Hence, the sea and rail legs can be composed of multiple legs, increasing the maximum possible chain length to 8 legs. The transport chain generation program produces the choice set of available transport chains, but already optimises the locations of transfers within each

available type of transport chain. Table A-2 in the appendix contains a full list of chain types that are implemented in the Swedish model.

The `ChainChoice` procedure

The ChainChoice module optimises the shipment size and vehicle type usage for each chain type and for each firm to firm relation based on total costs. Up to version 1.1.2 of the model, the total firm to firm transport volume was assigned to the single best transport chain. Version 1.1.3 of the logistic model introduced the possibility to split the firm-to-firm flow between the best and second best transport chains by assuming the costs of both of these chains follow a normal probability distribution that has the calculated transport costs as its mean value. An assumption is made about the standard deviation.The chain choice procedure accounts for the consolidation and distribution of shipments to optimise the load factor of vehicles. Consolidation centres are locations where goods are transhipped (and possibly stored, but this is not modelled), with small loads getting in and larger loads getting out. Distribution centres are locations where goods are transhipped (and possibly stored), with large loads getting in and small loads getting out. Both consolidation and distribution centres exist not only in road transport, but can also be ports, airports or rail terminals. Whether a shipment can be consolidated at a consolidation centre with other shipments, depends on the OD flows, which are endogenous in the logistics model.
The level of consolidation (or the load factor of the vehicles) between consolidation centres and distribution centres, is determined in an iterative procedure. This happens because the question whether or not a particular shipment will be consolidated on a certain OD leg, depends on whether other cargo (other shipments) will be transported over the same OD leg, so there will be something to consolidate with. However, the latter is an output of the model (also see the Method Report on the Swedish logistics model).
The model derives the level of consolidation of OD flows between consolidation centres from a flow size ranking. This flow size ranking is the product of the allocated OD flow and the observed port output. In the first iteration, no shipments have been allocated yet, so a typical consolidation factor is assumed (this can be set in the BuildChain control files, see section 4.5). In this iteration, ChainChoice allocates shipments to the optimal chain for each chain type (this is set with the argument `/UPDATE=CHOSEN`, see section 3.3). The calculated consolidation factors from the first iteration are input to the second iteration. If the OD flow has its origin or destination in an important port, the flow is weighed with the observed terminal throughput, to direct more consolidation towards important ports. The OD flows are ranked according to this weighed consolidation factor and the final consolidation level is determined by allocating increased consolidation levels to the ranked OD flows. The range of consolidation levels is set in the ChainChoice control files (see section 5.5; by default the range is 0.05 to 0.95).

In version 1.2.1 of the model a program (ConsolidRateMode) was added that combines the sub mode specific consolidation factors generated by the ChainChoice program into mode specific consolidation factors, by taking the maximum consolidation factor among all sub modes belonging to the same mode. The modes that are distinguished are Road, Rail, Sea (including IWW)and Air. ConsolidRateMode overwrites the sub mode specific consolidation factors

written by the ChainChoice program. In principle the output files for sub modes belonging to the same mode are identical, with two exceptions related to:

- The final consolidation factors are restricted to the sub mode specific range of consolidation levels.
- A user supplied list of exceptions (specified by the Cap_Except key in the ChainChoice control file) can overwrite the consolidation factors generated by the model. In fact users are recommended to use these exogenous consolidation rates, but the option to use the model generated consolidation factors is maintained.

In the current model, consolidation only takes place within a commodity group. The choice of starting factor has a small but not negligible influence on the final results. To solve this, one could do more than three iterations. Consolidation in the model can take place at all transfer nodes (terminals, collection/distribution centres) that are included in the model system.

3.3 Controlling the iterative procedure

The structure of the logistic model can be modified or controlled on two aspects: the iteration between the BuildChain and ChainChoice procedures, and the starting value for the consolidation factor. The batch job `Commodity.bat` contains the core of the logistic model and is used to modify the model's structure. The content of this file is visualised in Figure 3-3. Please note that this program calls upon the logistic procedures `buildchain.exe` and `chainchoi.exe`. These procedures require control files and optional arguments.

By default, the iterative procedure consists of three rounds. In the first iteration step the consolidation on the network links is unknown (no logistic decisions have been simulated yet). Therefore, in the first call upon the procedure `buildchain.exe` the consolidation factor is set with the argument `/FACTOR`. This value can be set to any desired level between 0 and 1. It is advised to use the default value of 0.75.

Figure 3-3: Contents of [Commodity.bat]

```
cd buildchain
buildchain.exe buildchain%1.ctl /FACTOR=0.75 /ITRNO=BC1
cd..
cd ChainChoi
chainchoi.exe chainchoi%1.ctl /UPDATE=ALL /ITRNO=CC1
java.exe -d64 -Xmx8g -Xms4g -jar ConsolidRateMode.jar %1
cd..
cd buildchain
buildchain.exe buildchain%1.ctl /ITRNO=BC2
cd..
cd ChainChoi
chainchoi.exe chainchoi%1.ctl /UPDATE=CHOSEN /RDVOLUMES=YES /ITRNO=CC2
java.exe -d64 -Xmx8g -Xms4g -jar ConsolidRateMode.jar %1
cd..
cd buildchain
buildchain.exe buildchain%1.ctl /ITRNO=BC3
cd..
cd ChainChoi
chainchoi.exe chainchoi%1.ctl /RDVOLUMES=YES /ITRNO=CC3
```

```
cd..
```

note that the batch job calls java.exe to run the ConsolidRateMode program. If the batch job fails because Java.exe cannot be found, the batch job should be adjusted to include the full path of the Java executable. When this path includes space-characters the path should be quoted.

The ConsolidRateMode is not called in the final iteration because the consolidation factors generated will not be used further.

Explanation of flags in [Commodity.bat]:

FACTOR	: exogenous consolidation factor. Range: [0,1] or [empty]. In the first iteration this is set to 0.75, which is advised as representative value. If the flag is missing, the consolidation factor is taken from the previous iteration
UPDATE	: sets the allocation of shipments to chains. ALL shipments are allocated to all available chains CHOSEN shipments are allocated to chosen chains
RDVOLUMES	: tells the chain choice program if OD-flows are available from previous iteration. If so, this argument should be set to YES
ITRNO	: Iteration number to be logged in the log file

The computation of the consolidation factor is set with the argument /UPDATE of the chainchoi.exe procedure. In the first iteration this argument is set to ALL, which means shipments are allocated to all available chains. The set of available chains consists of the optimal chain for each chain type). Based on these allocated shipments, the consolidation factor is determined.

In the second iteration round, the argument for the initial consolidation factors can be excluded from the call upon buildchain.exe. Now, the logistic chains are build with the consolidation factors that were computed in the first iteration. When the logistic choices are simulated in ChainChoi, the consolidation factors are determined and updated. In this second iteration the /UPDATE argument is set to CHOSEN, indicating that shipments are only allocated to the chosen chain and used in the computation of the consolidation for the third iteration round. The ChainChoi program now takes another argument /RDVOLUMES=YES, indicating that OD flows are available from the previous iteration, to calculate the consolidation levels and transport costs for each shipment. The final ChainChoi iteration ends with the /UPDATE argument blank[7], in order to keep the consolidation factors that are stored, consistent with the consolidation factors that were used to determine the chain choices.

[7] The default value for the UPDATE argument is NONE

If desired, the batch job `Commodity.bat` can be extended by adding additional iterations. Any additional iteration requires a call on the BuildChain and ChainChoi programs. It is advised to call `buildchain.exe` without consolidation factor argument and `chainchoi.exe` with the following arguments: `/UPDATE=CHOSEN /RDVOLUMES=YES`. The last iteration is always run without the consolidation update switch to `/UPDATE=CHOSEN`. , so the additional iteration is inserted between the 2nd and 3rd iteration in Figure 3-3. The following lines should be added for each additional iteration:

Figure 3-4: Extension of [Commodity.bat] for additional iterations

```
cd..
cd buildchain
buildchain.exe buildchain%1.ctl /ITRNO=BC4
cd..
cd ChainChoi
chainchoi.exe chainchoi%1.ctl /UPDATE=CHOSEN /RDVOLUMES=YES /ITRNO=CC4
java.exe -d64 -Xmx8g -Xms4g -jar ConsolidRateMode.jar %1
```

3.4 Input and output to the model

Inputs to the logistics model

The logistic costs are calculated with the transport costs and several input data. An overview of data that is used as input is given below:

- Transport costs: based on time, distance and extra costs (e.g. tolling) calculated from a network model
- Terminals, e.g. direct access, location, which commodities can be handled, terminal time and costs
- Commodity-specific costs, e.g. order and storage costs
- Vehicle data, e.g. capacity and which vehicle types can be used for which commodities
- Restrictions, e.g. locked transport solutions or on draught in ports

Detailed lists of input files are given in the next chapters where the BuildChain and ChainChoice procedures are discussed. In addition to that an overview of all input parameters and variables is provided in Appendix B.

Outputs of the logistics model

The output of the model consists of

- f2f flows, including the shipment size and shipment frequency, the chosen transport chains (number of OD legs, vehicle/vessel type on each leg, transhipment locations) and the logistics costs (by cost item).
- Various aggregates from the f2f flows:
    - OD matrices of tonnes and tonne-kilometres by mode and commodity type (but also more aggregate –non-OD- statistics on

the total number of tonnes and tonne-kilometres by mode and commodity type),

- OD matrices of loaded vehicles by vehicle/vessel type, and of loaded and empty road vehicles. These can be used in assignment to the networks, applying the same network model as for the transport cost inputs.
- output flows by vehicle type or by terminal.
- total logistics costs per PWC flow, for use in the future determination of PWC flows.

To generate these outputs a separate procedure is used to process and aggregate the calculated logistic chains that were chosen in the model (see RAND Europe and SITMA, 2005).

A list of input and output files is provided in the following chapters where the two logistic components are described in detail.

CHAPTER 4 BuildChain procedure

4.1 Introduction

The `BuildChain` module builds and selects the "best" chain per commodity and chain type for each origin – destination pair. A new feature introduced in version 1.1.3 of the Samgods model is the possibility to generate chains with alternative, second best, transfer terminals.This Chapter describes the `BuildChain` procedure in the logistic program. First an overview is given of the input and output files. Next the content of the procedure is described. It is indicated how the procedure can be controlled by modifying input files or the control files that define the settings for the simulation.

4.2 Input files

The `BuildChain` uses a large number of input files to generate logistic chains. Below an overview is given of all files that are used. The overview contains a description of the content of all input files. If applicable, the units of the data are presented, or a reference is made to a detailed file description with units of variables elsewhere in this document.

Table 4-1: Overview of BuildChain input files

Folder	Filename	Description
\BUILDCHAIN\	Buildchain.ctl	Control file for running the BuildChain procedure, containing common settings for all commodities. Structure of file is explained in Section 4.5
\BUILDCHAIN\	Buildchain1.ctl	control file for running the BuildChain procedure for commodity type 1. Structure of file is explained in Section 4.5
\BUILDCHAIN\	…	…
\BUILDCHAIN\	Buildchain16.ctl	control file for running the BuildChain procedure for commodity type 1. Structure of file is explained in Section 4.5
INPUT\COST\	cargo.txt	contains value, holding costs and order costs for all commodities. For a description of the content of the file, and the units of parameters see Section 4.5.
INPUT\COST\	PilotFees.txt	specifies the fees at each terminal [SEK per vehicle]
INPUT\COST\	Vhcls_dry_bulk.txt	contains dry bulk parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to Vhcls_gen_cargo.txt described in Section 4.5.
INPUT\COST\	Vhcls_gen_cargo.txt	contains general cargo parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. For a description of the content of the file, and the units of parameters see Section 4.5.
INPUT\COST\	Vhcls_liq_bulk.txt	contains liquid bulk parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to Vhcls_gen_cargo.txt described in Section 4.5.

Folder	Filename	Description
INPUT\LOS\	FreqAir.314	frequency for air mode [#transports/week]
INPUT\LOS\	FreqCombi.314	frequency for combi mode [#transports/week]
INPUT\LOS\	FreqContainerVessel.314	frequency for container vessels [#transports/week]
INPUT\LOS\	FreqLorry.314	frequency for lorries [#transports/week]
INPUT\LOS\	FreqOtherVessel.314	frequency for other vessels [#transports/week]
INPUT\LOS\	FreqRailFerry.314	frequency for rail ferry [#transports/week]
INPUT\LOS\	FreqRoadFerry.314	frequency for road ferry [#transports/week]
INPUT\LOS\	FreqRoRoVessel.314	frequency for RoRo vessel [#transports/week]
INPUT\LOS\	FreqSystem.314	frequency for system [#transports/week]
INPUT\LOS\	FreqWaggonload.314	frequency for wagonload [#transports/week]
INPUT\LOS\	v101_dist.314	distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_ddist.314	Domestic distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_timeh.314	time matrix for vehicle type 101 [hour]
INPUT\LOS\	v101_xkr.314	extra costs matrix for vehicle type 101 [kSEK]
…	…	…
INPUT\LOS\	v401_dist.314	distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_ddist.314	Domestic distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_timeh.314	time matrix for vehicle type 401 [hour]
INPUT\LOS\	v401_xkr.314	extra costs matrix for vehicle type 401 [kSEK]
INPUT\GENERAL\	CalibrationParameters.txt	Scale factors for sea and ferry times per port area and STAN-group
INPUT\NODES\	nodes.txt	contains general node information for all commodities. For a description of the content of a node file and the units of parameters see Section 4.5
INPUT\NODES\	transferroadroad.txt	Contains a dummy variable indicating whether or not road-road transfers are allowed
INPUT\NODES\	transferroadtrain.txt	Contains a dummy variable indicating whether or not road-train transfers are allowed
INPUT\NODES\	transferroadsea.txt	Contains a dummy variable indicating whether or not road-sea transfers are allowed
INPUT\NODES\	transferroadcombi.txt	Contains a dummy variable indicating whether or not road-combi transfers are allowed
INPUT\NODES\	transferroadair.txt	Contains a dummy variable indicating whether or not road-air transfers are allowed
INPUT\NODES\	transferroadroadferry.txt	Contains a dummy variable indicating whether or not road-road-ferry transfers are allowed
INPUT\NODES\	Transferseasea.txt	Contains a dummy variable indicating whether or not sea-sea transfers are allowed
INPUT\NODES\	transfercombisea.txt	Contains a dummy variable indicating whether or not Combi-sea transfers are allowed
INPUT\NODES\	transferwagonloadrailferry.txt	Contains a dummy variable indicating whether or not wagonload-railferry transfers are allowed
INPUT\NODES\	transferwagonloadsea.txt	Contains a dummy variable indicating whether or not wagonload-sea transfers are allowed
INPUT\NODES\	transferfeedertrainwagonload.txt	Contains a dummy variable indicating whether or not feeder train-wagonload transfers are allowed
INPUT\NODES\	transfersystemtrainsea.txt	Contains a dummy variable indicating whether or not system train-sea transfers are allowed
INPUT\NODES\	Directsea.txt	Contains a dummy variable indicating whether or not direct sea access is allowed
INPUT\NODES\	directsystemtrain.txt	Contains a dummy variable indicating whether or not direct system train access is allowed
INPUT\NODES\	directfeedertrain.txt	Contains a dummy variable indicating whether or not direct feeder train access is allowed
INPUT\NODES\	directwagonload.txt	Contains a dummy variable indicating whether or not direct wagonload access is allowed
INPUT\NODES\	containerhandling.txt	Contains a dummy variable indicating whether or not direct containers can be handled

Folder	Filename	Description
INPUT\PWC\	pwc_01.txt	producer demand matrices, with the annual demand (flows from producers or wholesales to consumer between zones) for commodity type 1. [annual demand in tonnes]
…	…	…
INPUT\PWC\	pwc_16.txt	producer demand matrices, with the annual demand (flows from producers or wholesales to consumer between zones) for commodity type 1 [annual demand in tonnes]
\BUILDCHAIN\	LOGSELECT.DAT	selection of OD pairs for which the available chains are stored in BuildChain1.log - BuilCHain16.log . To enable this output refer to LOGSELECT parameter. If LOGSELECT=1 it is mandatory provide this file under the BUILDCHAIN folder.
\BUILDCHAIN\	select.dat	selection of OD pairs for which the available chains are stored in connection1.lst - connection16.lst . To enable this output refer to SELECT and CONNLST parameters.

4.3 Output files

In the overview below, the output files are listed that are generated by the `BuildChain` procedure. For each commodity type an output file is generated. This file contains the logistic chains between all zone to zone combinations. This result is input to the `ChainChoice` procedure.

Table 4-2: Overview of BuildChain output files

Folder	Filename	Description
\BUILDCHAIN\OUTPUT\	Chains1.dat	output file with logistic chains for commodity type 1
\BUILDCHAIN\ OUTPUT\	…	…
\BUILDCHAIN\ OUTPUT\	Chains16.dat	output file with logistic chains for commodity type 16
\BUILDCHAIN\ OUTPUT\	connection1.lst	List of all available connections for a selection of relations for commodity type 1[8]
\BUILDCHAIN\ OUTPUT\	..	..
\BUILDCHAIN\ OUTPUT\	Connection16.lst	List of all available connections for a selection of relations for commodity type 16[8]
\LOG\	Log01.log	log file with reports for commodity type 1. See LOGSELEC parameter to control the amount of logged information.
\LOG\	…	…
\LOG\	Log16.log	log file with reports for commodity type 16. See LOGSELEC parameter to control the amount of logged information.

The output files contain a varying number of logistic chains depending on commodity type and availability of transport modes. Below the output is illustrated with a sample of an output.

Figure 4-1: example of a BuildChain output file with header description

orig	*dest*	*No.of chains*	*Chain type*	*orig*	*dest*	*orig index*	*dest index*	*consol. factor [%]*
711400	716000	19	C	711400	716000	0	28	
			A	711400	716000	0	28	
			B	711400	716000	0	28	
			BSB	711400	718003	0	35	
				718003	718004	35	36	0.05
				718004	716000	36	28	

[8] This output is optional. The selection of relations is made in the 'select.dat' file in the \BUILDCHAIN-directory.

```
AFEA    711400  718121  0       53
        718121  718111  53      51      0.05
        718111  886111  51      497     0.5048
        886111  716000  497     28
CGHC    711400  718014  0       43
        718014  718017  43      46      0.05
        718017  908411  46      565     0.05
        908411  716000  565     28
ADA     711400  918011  0       581
        918011  718012  581     41      0.05
        718012  716000  41      28
```

The chain output file contains the origin and destination zone (in this case between zone 711400 and 716000). Next the number of chains is specified: 19. For each of these chains, the chain type is given (e.g. C= Heavy Road, non-container), the origin and destination of the respective leg, the origin and destination indices of the leg, the time, distance and extra costs. If applicable, the line contains a frequency and consolidation factor for the leg. This depends on the respective transport mode. The example below shows that for feeder trains (modes E) or wagonload trains (mode F) consolidation factors are computed. For Heavy lorry mode A, for containerised transport, consolidation is not an option. For non-containerised transport, consolidated heavy lorry is coded as mode S. Consolidation with heavy lorry occurs only in the middle of 3 or more legged chains, and cannot occur at the destination or origin leg.

The connection list output files specify the available chains for a selection of relations. This selection of relations is optional and controlled in the Select.dat file. The output lists have a simple structure: origin node, available mode, destination node.

Figure 4-2: Sample of a connections list output file

```
Orig        Mode  Dest
711400      A     711400
711400      A     711401
711400      A     711402
711400      A     711403
711400      A     711500
711400      A     711700
711400      A     712000
711400      A     712300
711400      A     712301
```

Depending of the LOGSELECT-setting, the log files (*.log) may contain warnings that occurred during execution of the module for the specific commodity type. For instance, a warning is written when no frequency is available; in that case the default frequency is used (the default frequency, DfltFreq, is set in the vehicle cost parameter files, see section 4.5). A warning is also given if the LOS attributes are incomplete (e.g. frequency is given, but the time or cost is not available); in that case the connection is discarded. An example of a log file fragment is given in the figure below:

Figure 4-3: Sample of a log file fragment

```
Using default frequency: Orig=711400; Dest=711401; Mode=A; Time=0.044197; Dist=2.23; Freq=84
Using default frequency: Orig=711400; Dest=711402; Mode=A; Time=0.072314; Dist=4.29; Freq=84
Using default frequency: Orig=711400; Dest=711403; Mode=A; Time=0.0368; Dist=1.84; Freq=84
```

4.4 Description

The logistic module first simulates the choice between a number of transport chains (with one to four legs and with different aggregate modes and different vehicle/vessel types for each leg). Transfers can take place within the road system (e.g. from van to truck), or between road, sea, rail and air transport (and the Swedish model also includes transfers within the sea and within the rail system: feeder and main haul transport). The transport chain generation program produces the choice set of available transport chains, but already optimises the locations of transfers within each available type of transport chain.

The tables in the appendix give an overview of the vehicle types and transport chains for Sweden. The current version of the model distinguishes 40 vehicle types, and 99 transport chains. For each transport chain type, `BuildChain` produces all possible transport chains and their associated logistic costs. For each chain type, the alternative with minimal logistic costs is stored.

The logistic costs function

The logistic costs function is specified in RAND Europe and SITMA (2005) and also specified in the Method report (Significance, 2016). The total annual logistics costs G of commodity k transported between firm m in production zone r and firm n in consumption zone s of shipment size q using logistic chain l:

$$G_{rskmnql} = o_{k.}(Q_{kmn}/q_{kmn}) + T_{rskql} + Y_{rskl} + (w_k + (d.v_k)).(q_{mnk}/2) \quad (1)$$

Where:

G: total annual logistics costs

O_k: order costs, dependent of the annual demand, for commodity k. $o_{k.}(Q_{kmn}/q_k)$ in equation 1

o_k: the constant unit cost per order of commodity k

Q_k: the annual demand (tonnes per year) for commodity k

q_{mnk} : the average shipment size for commodity k transported between firm m and firm n.

T_{rskql}: transport, consolidation and distribution costs between production zone r and consumption zone s, for commodity k, shipment size q, using logistic chain l. T comprises of link-based costs, vehicle/vessel type specific costs, vehicle/vessel pair specific costs, commodity type specific costs. The transport costs function is specified in detail below.

Y_{rskl}: capital costs of goods during transit. $Y_{rskl} = (d.t_{rsl}.v_k.Q_k)/(365*24)$

d: the discount rate (per year)

t_{rsl}: the average transport time (in hours) between production zone r and consumption zone s, using logistic chain l.

v_k: the value of the commodity that is transported (in SEK per tonne).

w_k: the storage costs (in SEK per tonne per year) for commodity k.

Please note that the order and holding costs are omitted in the implementation of the logistic costs function in the `BuildChain` procedure. These cost components do not vary across logistic chain alternatives because a fixed average shipment size per commodity type is used in `BuildChain`. Therefore, these costs are irrelevant for the generation of the optimal chain for each chain type. The `ChainChoice` procedure does include these components as these costs are affected by the chosen shipment sizes.

The transport costs and capital costs during transit between production zone r and consumption zone s, using logistic chain l is derived by summing the transport costs and capital costs of each leg i in the logistic chain l:

$$T_{rskql} + Y_{rskl} = \sum_{i \in l} \left(T_{kqi} + Y_{ki} \right) \tag{2}$$

The transport cost function T_{kqi} is dependent on the transport mode that is used in the leg. In BuildChain a typical vehicle type is used (which is set in the control file), and ChainChoice uses the optimal vehicle type. The transport cost function is specified for container, non-container and ferry transport. The transport cost function for containers consists of a time cost, distance cost, other link costs (e.g. tolling), loading (and unloading) costs, fairway dues[9], and pilot fees. The computation of the link based transport and capital costs is implemented as follows:

$$\begin{aligned} T_{kqi}^{\text{container}} + Y_{ki} = {} & n \cdot (t_v^{load;\text{container}} \cdot (f_o^{time} + f_d^{time}) + t_i) \cdot c_v^h + q_k \cdot (t_v^{load;\text{container}} \cdot (f_o^{time} + f_d^{time}) + t^{wait} + t_i) \cdot d_k^h) \\ & + n \cdot dist_i \cdot c_v^d \\ & + n \cdot c_i^{oth} \\ & + (f_o^{cost} + f_d^{cost}) \cdot q_k \cdot c_v^{load;\text{container}} \\ & + n \cdot c_v^{fairw} + q_k \cdot c_v^{fairwT} \\ & + n \cdot c_v^{pilot}. \end{aligned} \tag{3a}$$

with: n = number of vehicles, $t_v^{load;container}$ = vehicle type specific containerload time, t^{wait} =

wait time (not at origin and destination) with $t^{wait} = \frac{0.5 \cdot 7 \cdot 24}{f}$, f = frequency of the

vehicle per week, t_i = link time, c_v^h = vehicle type specific time costs, d_k^h = interest costs per hour per tonne of commodity k calculated as: $d_k^h = v_k \cdot i / (365 * 24)$, i = interest rate per year, $dist_i$ = link distance, c_v^d = vehicle type specific distance costs, c_i^{oth} = other link costs, f_o^{time} = technology factor for terminal time efficiency at origin node, f_d^{time} = technology factor for terminal time efficiency at destination node, f_o^{cost} = technology factor for terminal cost efficiency at origin node, f_d^{cost} = technology factor for terminal cost efficiency at destination node, q_k = shipment size, $c_v^{load;container}$ = vehicle type specific load costs per tonne for

[9] Within the model fairway dues are specified by vessel type. Half of the specified fairway dues is attributed to ach end of the transport chain leg

containers, c_v^{fairw} = fairway dues by vehicle, c_v^{fairwT} = fairway dues by tonne, c_v^{pilot} = sum of the pilot fees at the origin and destination nodes.

The equation given above implies that the transfer costs between subsequent legs are calculated as the sum of the unloading costs of the first vehicle and the loading costs of the second vehicle. For the following cases the vehicles involved are in fact not (un)loaded:

- Transfers involving ferries
- Transfers involving RoRo-vessels
- Rail-Rail transfers (marshalling)

For these special cases the loading times ($t_v^{load,container}$) and costs ($c_v^{load,container}$) should be replaced by the transfer times ($t_v^{transfer,container}$) and costs ($c_v^{transfer,container}$).

The Swedish vehicle/vessel type classification (see Table A-2 in the appendix) is based on the assumption that unitised transport can be used with most vehicle/vessel types (exceptions: the first three light/medium road vehicles, system train and airplane cannot be used for container transport; the Kombi train and the container vessels are for container transport only). This means that the cost for the unitised variant includes costs for initial stuffing of the container (at the sender) and final stripping (at the receiver) and that there are differences in the transfer costs (generally speaking container transfers are cheaper than other transfers at consolidation and distribution centres). Therefore, the link costs at the origin and destination of the chain include an additional cost component for stuffing and stripping. If link *i* starts at production zone *r*, the link based costs are increased with the stuffing costs, calculated as:

$$T_{kqi=r}^{container} = T_{kqi}^{container} + n \cdot c_v^{stuffing} \qquad (3b)$$

If link *i* ends at consumption zone *s*, the link based costs are increased with the stripping costs, calculated as:

$$T_{kqi=s}^{container} = T_{kqi}^{container} + n \cdot c_v^{stripping} \qquad (3c)$$

The model assumes equal stuffing and stripping costs, which is set in the control file of the ChainChoice program.

For non-containerised transport the transport function is similar, except some non-container parameters are different:

$$
\begin{aligned}
T^{\text{non-container}}_{kqi} + Y_{ki} = {} & n \cdot (t_v^{load;non-container} \cdot (f_o^{time} + f_d^{time}) + t_i) \cdot c_v^h + q_k \cdot (t_v^{load;non-container} \cdot (f_o^{time} + f_d^{time}) + t^{wait} + t_i) \cdot d_k^h) \\
& + n \cdot dist_i \cdot c_v^d \\
& + n \cdot c_i^{oth} \\
& + n \cdot c_i^{position} \\
& + (f_o^{cost} + f_d^{cost}) \cdot q_k \cdot c_v^{load;non-container} \\
& + n \cdot c_v^{fairw} + q_k \cdot c_v^{fairwT} \\
& + n \cdot c_v^{pilot}.
\end{aligned}
\tag{4}
$$

with $c_v^{position}$ = positioning costs by vehicle.

Ferry links have a specific cost function because fairway dues, pilot fees and technology factors for loading costs do not apply. The categorisation to containerized or non-containerized is derived from the previous leg. If the previous leg was non-containerized, the transport costs at ferry links becomes:

$$
\begin{aligned}
T^{\text{ferry}}_{kqi} + Y_{ki} = {} & n_{ro} \cdot (t_v^{load;\text{non-container}} + t^{wait} + t_i) \cdot (c_v^h + d_k^h) \\
& + n_{ro} \cdot dist_i \cdot c_v^d.
\end{aligned}
\tag{5}
$$

with n_{ro} = number of vehicles for the shipment on the ferry. If the previous leg was containerized, the transport costs at ferry links becomes:

$$
\begin{aligned}
T^{\text{ferry}}_{kqi} + Y_{ki} = {} & n_{ro} \cdot (t_v^{load;\text{container}} + t^{wait} + t_i) \cdot (c_v^h + d_k^h) \\
& + n_{ro} \cdot dist_i \cdot c_v^d.
\end{aligned}
\tag{6}
$$

Again for the special cases given above the loading times ($t_v^{loadnon-,container}$) and costs

($c_v^{loadnon-,container}$) should be replaced by the transfer times ($t_v^{transfernon-,container}$) and costs

($c_v^{transfernon-,container}$).

The cost function parameters are set in separate input files and control files to facilitate running policy variants. In the following of this document, when the input files are discussed in more detail references are made to the cost functions specified above.

The annual discount rate can be set in the control file (see the description of the `INTEREST` parameter in section).

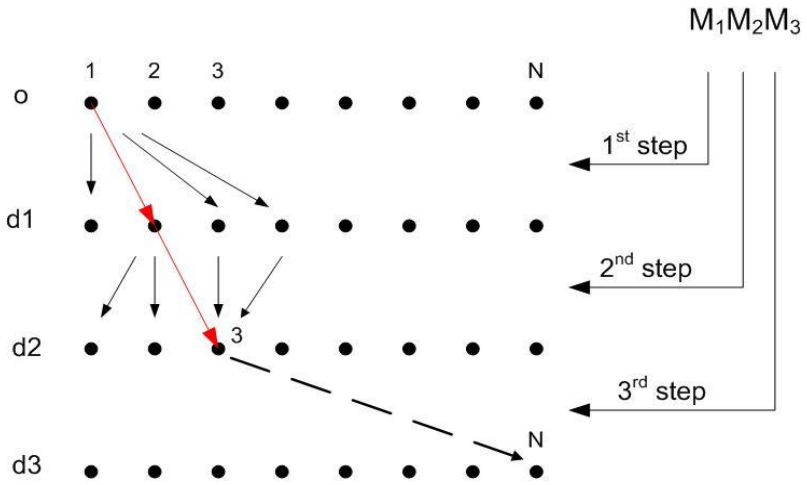
Build logistic chain algorithm

BuildChain algorithm for finding the best chains

The BuildChain procedure searches for all typical logistic chains and identifies the optimal chain for each of these chain types. The current model distinguishes

99 chain types for Sweden (see Table A-3 for an overview). For each chain type, BuildChain calculates the optimal transfer locations and logistic costs for the logistic chain. In doing so, the algorithm follows a stepwise approach in adding extra legs to chains and analysing the optimal transfer locations.

Figure 4-4: Search algorithm, the optimal two leg chain M_1M_2 from origin 1 to destination 3 is indicated in red



The algorithm is based on the assumption that the optimal route to a final destination consists at least of the optimal route to an intermediate destination, if this intermediate destination is on the optimal route to the final destination. The only additional information required is the optimal route between the intermediate destination and the final destination. This approach is explained in Figure 4-4.

For each origin o, the procedure generates chains that can consist of one leg (M_1) to for instance three legs ($M_1M_2M_3$). All transport modes are taken into account. The optimal chain of just one leg (M_1) to each destination is trivial: the alternative with the least logistic costs.

The algorithm generates chains from this origin to each possible destination d, and tries to use the information from the chains that are produced for shorter chains as efficient as possible. Now, suppose the procedure is searching for the optimal chain of three legs ($M_1M_2M_3$) from origin 1, to destination N, under the condition that the second transfer is made in node number 3. The program has already determined the optimal logistic chain of two legs to this transfer point, as indicated in red in Figure 4-4. It will use this chain as the first two legs of the new three legged chain from origin 1, to N, with a second transfer at node number 3. The program only needs to determine the optimal third leg of this chain. Please note that the program searches for three legged chains from zone 1 to N through all possible transfer nodes, not only through node 3. The optimal two legged chain between this transfer node and zone 1 is already determined by the program.

BuildChain algorithm for finding the second-best chains

As explained above the set of chains of type $M_1M_2...M_n$ to all destinations is constructed from the set of chains of type $M_1M_2...M_{n-1}$ to all destinations by adding an additional leg from the set of all connections with mode M_n. The same method is used to construct the set of second best chains to all destinations, but now the connections used to construct the best chains are excluded. The chain type obtained that way is marked as $M_1M_2..M_{n-1}2M_n$.

Please note that the `BuildChain` procedure uses a different shipment size than the `ChainChoice` procedure. The optimal shipment size, q_k in equation (1), is only available after the logistic decisions have been simulated. So `BuildChain` uses a fixed shipment size, representative for the specific commodity type and (possibly) zone, when the set of possible logistic chains are generated. This shipment size is determined (internally by the BUILDCHAIN program) as an average of the PWC-flows and is used in all iterations of the model.

4.5 Controlling the BuildChain procedure

The `BuildChain` procedure is controlled through the commodity specific control files. These control files are case sensitive, because the mode identifiers used within the logistic model are case sensitive (for example, mode D and d are different combi train modes). This section describes the usage of this control file and gives an overview the input files.

By using the INCL-parameter within a control file another control file can be included. Typically this will be done when several control files share a common set of parameters. Within the logistic model, parameters shared between different commodities are included in a common control file this way. This common control file, in turn, contains an INCL-parameter referencing a control file containing settings shared between different programs within the logistic model.

Figure 4-5 gives an example of the control file for running the `BuildChain` procedure for commodity 1. The parameters in this file specify parameter values for the procedure or paths to input or output files. The parameters in the control file are described in the overview that follows. This description is not exhaustive but illustrates the most important input files that are used to calculate the logistic costs for the transport chains.

Figure 4-5: Example of a BuildChain control file: [`BuildChain1.ctl`]

```
INCL=buildchain.ctl
COMMODITY=1
VHCL=..\Input\Cost\VHCLS_COM01.TXT
PWC=..\Input\PWC\PWC_01.txt
VHCLA=104
…
VHCLX=106
CHAINS=OUTPUT\Chains1.dat
LOG=..\log\log01.log
CONNLST=OUTPUT\connection1.lst
```

Explanation of parameters:

INCL	: reference to the control file that contains the common settings for all commodities. An example of this file is given in Figure 4-6.
LOGSELECT	: Indicator (0/1/2) that controls the amount of information in the log file about incomplete LOS: LOGSELECT= 0 means no information is logged. LOGSELECT= 1 means information is logged for a selection of relations. To specify the list of relations it is mandatory to provide an extra file (LOGSELECT.DAT) as input. Please refer to input file section for further details. LOGSELECT= 2 means information is logged for all relations (not recommended for disk space saving).
COMMODITY	: commodity code [1..16]
VHCL	: path to file with vehicle data for the specific commodity type
PWC	: path to file with producer demand matrices for the specific commodity
VHCLA	: Typical vehicle type number for mode A
…	…
VHCLR	: Typical vehicle type number for mode R
CHAINS	: path for output file containing the optimal chains
SELECT	: path to input file containing the collection of relations that should be included in the detailed connection list output. Example SELECT=BUILDCHAIN\SELECT.dat. This parameter works in connection with CONNLST.
CONNLST	: path to output file (see Figure 4-2: Sample of a connections list output file) containing all connections on a selection of relations. The selection list is specified by the SELECT specifier.
LOG	: path for log file

Figure 4-6: Example of a common BuildChain control file: [BuildChain.ctl]

```
INCL=..\Input\Cost\OtherCostMatters.ctl
LOGCTL=1
LOGFLS=1
LOGCST=1
LOGSELECT=0
INTEREST=0.1
CONSOL=0.05,0.95
CONSOLA=0.05,0.5
```

```
CONSOLB=0.05,0.5
CONSOLC=0.05,0.5
CONSOLS=0.5,0.9
TONNES=DYNAMIC_AVERAGE
ALL_LORRY_TYPE_CONSOL=1
TYPES=..\Input\chaintype.lis
NODES=..\Input\Nodes
CARGO=..\Input\Cost\cargo.txt
PILOTFEES=..\Input\Cost\pilotfees.txt
LOSDir=..\Input\LOS
LOSFAC=..\Input\General\CalibrationParameters.txt
CONSOLDIR=..\ChainChoi\OUTPUT
SELECT=BUILDCHAIN\Select.dat
CONNLST=OUTPUT\connection1.lst
```

Explanation of parameters:

INCL	: reference to the control file that contains the common settings for different programs within the logistic model. An example of this file is given in Figure 4-7.
LOGCTL	: Indicator (0/1) that determines whether or not CTL file settings will be logged in the log file.
LOGFLS	: Indicator (0/1) that determines whether or not input file information will be logged in the log file.
LOGCST	: Indicator (0/1) that determines whether or not cost parameters will be logged in the log file.
INTEREST	: Interest rate used in cost calculations [%/year]
CONSOL	: Default consolidation range used when no mode specific range is specified
CONSOL<mode>	: Consolidation range for mode <mode>
TONNES	: This parameter is used to set the logistic cost parameter q (see equation 1) and calculate the logistic costs for the chains [tonnes per shipment]. If this parameter is a number, then this number will be used as the typical shipment size for this commodity. If this the parameter is set to DYNAMIC_MAX, DYNAMIC_AVERAGE or DYNAMIC_GEOMEAN the typical shipment size will be different for different zones and be calculated as the maximum, average or geometric mean of the PC matrix.

ALL_LORRY_TYPE_CONSOL	: 0/1 switch that determines whether or not consolidation is allowed for all lorry types[10] (1 means all lorries are consolidated)
TYPES	: Path to file containing all available chain types
NODES	: Path to directory containing the nodes input files. Each nodes file contains a single node variable for all commodities. The file names of the nodes files are fixed.
CARGO	: path to file with the cargo values, holding costs and order costs used in the logistic cost function.
PILOTFEES	: path to file with pilot fees.
LOSDIR	:Path to directory containing the Level of service files. The level of service files have fixed names, referring to the applicable vehicle type.
LOSFAC	: Path to input file containing scale factors for ferry and vessel times, per port area and STAN-group
CONSOLDIR	:Path to directory containing the consolidation factor matrices. The consolidation factor files have fixed names, referring to the applicable mode and commodity.
SELECT	: path to input file containing the collection of relations that should be included in the CONNLST-file.

Figure 4-7: Example of a common control file: [OtherCostMatters.ctl]

```
STUFF=60
INTEREST=0.100
ERRLOG=..\LOG\Error.log
```

Explanation of parameters:

STUFF	: container stuffing costs.
INTEREST	: interest rate used to calculate capital costs.
ERRLOG	: path to a shared log file for error messages.

Explanation of input files:

Below the most important input files are discussed. The content of each file is explained and it is indicated how the input is used by the BuildChain procedure.

[10] Regardless of this setting a separate mode S (consolidated heavy lorry) is distinguished in the mode. Because this mode is only allowed between terminals and on international relations, a consolidation factor range may be used that differs from the other lorry modes (A,B,C).

The PWC (production-wholesale-consumption) matrix files (one per commodity) contain the transport volumes on each od-relation. These flows are categorized into 9 sub cells:

1. flows from small firms to small firms
2. flows from small firms to medium-sized firms
3. flows from small firms to large firms
4. flows from medium-sized firms to small firms
5. flows from medium-sized firms to medium-sized firms
6. flows from medium-sized firms to large firms
7. flows from large firms to small firms
8. flows from large firms to medium-sized firms
9. flows from large firms to large firms.

The format of the PWC matrix files is given in Table 4-3:

Table 4-3: Contents of `PWC-matrix files`

Column	Parameter	Description
1	Origin	Number of origin zone
2	Dest	Number of destination zone
3	SubCell	Shipment size category
4	Volume	Annual transport volume within sub cell
5	NRelations	Number of firm to firm relations within sub cell

BUILDCHAIN uses the input of this file to determine a typical shipment size and to select the od-relations that will be modelled within the logistic model.

The file `cargo.txt` (Table 4-4) can be used to test different assumptions in the commodity specific holding and order costs, or the average values of commodities. This input file contains the commodity specific parameters in the logistic costs function (se equation 1). The file can be used to modify the assumptions in commodity specific costs. For instance, if the value of a commodity is increased (column 2), this will increase the capital costs on both the inventory at the receiver and on the inventory in transit. This will affect both the chain generation procedure as the chain choice procedure. Increasing the holding or order costs, will affect the logistic choices in the `ChainChoice` procedure, when the optimal shipment sizes are determined. `BuildChain` uses a fixed average shipment size, so the order and holding costs do not vary across the logistic chain alternatives.

Table 4-4: Contents of `cargo.txt`

Column	Parameter	Description
1	Commodity	commodity code [1..16]
2	STAN-group	STAN-group for this commodity
3	Value	value of commodity, v_k in equation 1 [SEK / tonne]

4	HoldingCosts	storage costs, w_k in equation 1 [SEK / (tonne*year)]
5	FixedOrderCosts	constant unit cost per order, o_k in equation 1[SEK]
6	ProportionalOrderCosts	Parameter used for the calculation of the annual demand dependent order costs[11]
7	Alpha	Parameter used for the calculation of the annual demand dependent order costs

The files `VHCLS_COM01.TXT..VHCLS_COM16.TXT`(Table 4-5) contain the commodity specific vehicle costs parameters in the transport costs function (equations 3 to 5). The input parameters include capacity, the time and distance costs for the vehicle type, and a default frequency that is used if no frequency is available from the network cost matrices.

Table 4-5: Contents of `vhcls_gen_cargo.txt`:

Column	Parameter	Description
1	Nr	vehicle type number
2	VesselType	vessel type, for all water transport modes. 0 = not a vessel; 1 = container vessel; 2 = roro vessel; 3 = other vessel.
3	Capacity	capacity of vehicle type. This influences the shipment sizes and required number of vehicles [tonne]
4	Coordination factor	Factor used to catch the fact that the available volume for each vehicle movement will be lower than the calculated annual consolidation volume at the terminal
5	HourCost	link-based time costs for the vehicle, parameter c_v^h in transport costs functions 3 and 4 [SEK per hour per vehicle]
6	KmCost	link-based distance costs for the vehicle, parameter c_v^d in transport costs functions 3 and 4 [SEK per km per vehicle]
7	OnFerryHourCost	link-based costs (time) on a ferry link, parameter c_v^h in transport costs functions 5 [SEK per hour per vehicle]
8	OnFerryKmCost	link-based distance costs on a ferry link, parameter c_v^d in transport costs functions 5 [SEK per km per vehicle]
9	PositioningCost	initial transport costs for vessel to arrive at the loading point for the link, parameter $c_v^{position}$ in transport costs function 4 [SEK per vehicle]
10	DfltFreq	default frequency if frequency is unavailable from LOS input files [#transports/week]

[11] The annual demand dependent order costs are calculated as: OrderCosts = FixedOrderCosts + ProportionalOrderCosts x AnnualDemand^Alpha

11	ContainerLoadTime	time costs for loading/unloading container transport at distribution and consolidation centres, parameter $t_v^{load;container}$ in transport costs function 3 [hour per vehicle]
12	ContainerLoadCost	costs for loading/unloading container transport at distribution and consolidation centres, parameter $c_v^{load;container}$ in transport costs function 4 [SEK per tonne]
13	ContainerTransferTime	time costs for loading/unloading container transport at distribution and consolidation centres, parameter $t_v^{transfer,container}$ in transport costs function 3 [hour per vehicle]
14	ContainerTransferCost	costs for loading/unloading container transport at distribution and consolidation centres, parameter $c^{transfer,container}$ in transport costs function 4 [SEK per tonne]
15	NonContainerLoadTime	time costs for loading/unloading non-container transport at distribution and consolidation centres, parameter $t_v^{load;non-container}$ in transport costs function 3 [hour per vehicle]
16	NonContainerLoadCost	costs for loading/unloading non-container transport at distribution and consolidation centres. parameter $c_v^{load;non-container}$ in transport costs function 4 [SEK per tonne]
17	NonContainerTransferTime	time costs for loading/unloading non-container transport at distribution and consolidation centres, parameter $t_v^{transfer,non-container}$ in transport costs function 3 [hour per vehicle]
18	NonContainerTransferCost	costs for loading/unloading non-container transport at distribution and consolidation centres. parameter $c_v^{transfer,non-container}$ in transport costs function 4 [SEK per tonne]
19	VhclFairwayDues	Fairway costs for vehicle, parameter c_v^{fairw} in transport costs functions 3 and 4 [SEK per vehicle][12]
20	TonnesFairwayDues	Fairway costs for vehicle parameter c_v^{fairwT} in transport costs functions 3 and 4 [SEK per tonne][12]

Increasing the time and distance costs of a vehicle type (column 4 and 5) will increase the transport costs for each leg specifically. This will affect the identification of optimal transport chains in `BuildChain`. Modifying the capacity (column 3) of vehicle types affects the frequency optimisation (in the `ChainChoice` procedure). This also counts for typical vehicle types as used in

[12] Within the model fairway dues are specified by vessel type. Half of the specified fairway dues is attributed to ach end of the transport chain leg

BuildChain. Modifying the default frequency in column 6 will only affect those links where no frequency is available from the LOS input files (see Section 4.2). The frequency affects the waiting time t^{wait} in equations 3 to 5. This parameter will affect the optimal logistic chain that was built. Increasing the container or non-container loading time and costs, in columns 7 to 10 will increase the transport costs for each leg specifically. This will affect the identification of optimal transport chains. Increasing the fairway dues by vehicle or tonnes, in columns 11 to 12 will increase the transport costs for each leg specifically. This will affect the identification of optimal transport chains.

The availability of vehicle types depends on the transport mode and is set in the control files (e.g. parameter VHCLA in the Buildchain and ChainChoice control files). The loading and unloading costs for non-container and container transports are not relevant for some vehicle types. The vehicle input files contain the system value "999999" for such irrelevant cases.

The LOS-input files that contain the Level-of-Service between all nodes in the network. These input files are delivered by Samgods in EMME/2 format. The LOS input matrices have fixed names, only the directory where these files are located are input to the BUILDCHAIN module. For a complete list of all files that should be available in the LOS input directory see paragraph 4.2. These LOS files give the travel time, distance, domestic distance or extra travel costs between nodes in the network. Intra zonal distance, time and costs are included as well. BuildChain generates intrazonal 'chains' with these intrazonal data. Intrazonal flows can only consist of one leg, so for intrazonal flows only direct chains are generated. The LOS of available modes determines wich direct chain is optimal (usually this is lorry).

Figure 4-8: Sample of distance matrix for vehicle type 101.

```
c EMME/2 Module:  3.14(v9.05)  Date: 09-06-10 11:04   User: E061/SIKA.......jm
c Project:        Samgods version 1
t matrices
a matrix=mf08  dist       0 distance
 711400 711400:5.340
 711400 711401:.630
 711400 711402:4.290
 711400 711403:1.840
 711400 711500:12.370
 711400 711700:28.780
 711400 712000:50.110
 711400 712300:17.690
 711400 712301:19.540
 711400 712500:38.260
 711400 712600:42.910
 711400 712601:41.780
 711400 712602:34.390
 711400 712603:42.580
```

The files containing the node variables have fixed names, only the directory where these files are located are input to the BUILDCHAIN module. For a complete list of all files that should be available in the nodes input directory see paragraph 4.2. These files can be used to modify or add a transfer terminal. If transfer locations are added or changed, this can affect the logistic chains that are generated: different chains are found by a change in optimal transfer locations.

The nodes.txt file is a tab-delimited text file. Here the content of the file is explained. For an elaboration on how to add nodes, zones or terminals see the description in section 2.4.

Table 4-6: Contents of `nodes.txt`:

Column	Parameter	Description
1	Nr	node code
2	Name	node name
3	Zone	zone code of node location, node code of terminal that provides direct access to this zone or empty ('0')
4	ConsolAggr	code for the group of nodes that are used to calculate an aggregated consolidation potential
5	PortArea	Number of port area
6	Domestic	dummy for domestic location. 1 = domestic, 0 = non-domestic.
7	CostTechnoFac	Cost efficiency factor at terminal, f^{cost} in transport costs functions 3 and 4 [0..1]
8	TimeTechnoFac	Time efficiency factor at terminal, f^{time} in transport costs functions 3 and 4 [0..1]
9	MaxDwtContainerVessel	Load constraint of container vessels that port can handle [tonne].
10	MaxDwtRoroVessel	Load constraint of Roro vessels that port can handle [tonne].
11	MaxDwtOtherVessel	Load constraint of other vessels that port can handle [tonne]
12	SeaOutput	Yearly sea output. This exogenous factor is used to determine a size ranking for sea terminals [tonne]
13	SeaContainerOutput	Yearly sea container output. This exogenous factor is used to determine a size ranking for sea terminals [tonne]
14	AirOutput	Yearly air output. This exogenous factor is used to determine a size ranking for air terminals [tonne]
15	Comments	Column to add a descriptive comment for a node

The first column contains the node number. The second contains the node name, this field is not used in the model. The third column contains a zone number, or zero. If node number (column 1) and zone number (column 3) are equal, the node

is considered to be a zone[13]. If these numbers are different, but the zone number is not equal to zero, the node can be used as a direct access terminal for the zone listed in column 3. Column 4 contains a consolidation group code. All nodes sharing the same consolidation group code belong to the same group and share the same consolidation potential. The fith column contains the port area, that determines the scale factor that will be used for ferry and vessel times. The sixth column in nodes.txt contains a domestic dummy. This is used to determine which logistic chains will be available. The logistic network outside Sweden is much less detailed, missing most consolidation or distribution possibilities. To avoid underestimation of consolidated flows to and from Sweden, direct access by consolidated heavy lorries is allowed for the non-domestic zones, indicated with value '0' in column 4. Columns 7 and 8 contain cost and time efficiency factors at the terminal, representative for the technology level of transfer facilities. These technology factors are used as multipliers upon the loading costs (see equations 3 and 4). If the factor is changed from 1.0 to 0.75, the loading costs in the transport costs function are decreased by 25%. Column 9 to 11 contain maximum load constraints. Modifying these values can affect the generation of logistic chains, in particular for logistic chains for flows with larger volumes. If a terminal has no maximum load constraint the value '99999' is used as a maximum. If the load constraint is not relevant (e.g. because the node is a zone) the value '0' is used. Column 12 to 14 contain exogenous (observed) output data for transfer terminals (from the port statistics). The outgoing link flows that are calculated in the logistic module are weighed by this exogenous factor to determine the ranking and level of consolidation (this is described in the Method Report). This is not a policy variable in the model but derived from observed data.

All other input files containing the node variables have the same tab-delimited format. The first column contains the node number, the second column the node name. After that 16 columns follow, one for each commodity. If the dummy variable has the value TRUE, the cell contains the commodity number. If the dummy variable has value FALSE the cell is empty.

The file `pilotfees.txt` contains the pilotfees at the nodes in the network for specific vehicle types. The first line in the file can be used for comments. The second line contains a file header that specifies the vessel types. The structure of the remainder of the file is shown in Table 4-7. The pilot fee parameters are used as input to the transport costs functions: c_{vt}^{pilot} in equations 3 and 4. Increasing the pilotfee at a node for a specific vehicle type, will increase the transport costs for each outgoing leg from this node for goods carried with that specific vehicle type. This increases the transport costs for these outgoing links and will affect the identification of optimal transport chains.

Table 4-7: Contents of `pilotfees.txt`:

Column	Parameter	Description
1	Terminal nr	code of zone with sea terminal
2	vehicle type from file header	Pilot fee in SEK per vehicle for applicable vessel type
..		
18	vehicle type from file header	Pilot fee in SEK per vehicle for applicable vessel type

[13] This procedure is implemented to allow for an efficient data handling.

CHAPTER 5 ChainChoice procedure

5.1 Introduction

The `ChainChoice` procedure simulates the logistic decisions and optimizes the logistic costs. In doing so, the shipment size is determined and a choice is made for a transport chain. This Chapter first presents an overview of the input and output files. Next, the procedure is outlined. Next, it is discussed how alternative options are set for this procedure and how these influences the results.

5.2 Input files

The `ChainChoice` uses a large number of input files. Some of these input files are used by the BuildChain procedure or calculated in that procedure. Below an overview is provided of all input files that are used.

Table 5-1: Overview of ChainChoi input files

Folder	Filename	Description
CHAINCHOI\	Chainchoi.ctl	Control file for running the BuildChain procedure, containing common settings for all commodities. Structure of file is explained in Section 4.5
CHAINCHOI\	Chainchoi1.ctl	control file for running the ChainChoice procedure for commodity type 1. Structure of file is explained in Section 4.5
CHAINCHOI\	…	
CHAINCHOI\	Chainchoi16.ctl	control file for running the ChainChoice procedure for commodity type 16. Structure of file is explained in Section 4.5
INPUT\COST\	cargo.txt	contains value, holding costs and order costs for all commodities. For a description of the content of the file, and the units of parameters see Section 4.5.
INPUT\COST\	PilotFees.txt	specifies the fees at each terminal [SEK per vehicle]
INPUT\COST\	VHCLS_COM01.TXT	contains commodity 1 parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to Vhcls_gen_cargo.txt described in Section 4.5.
INPUT\COST\	..	...
INPUT\COST\	VHCLS_COM16.TXT	contains commodity 16 parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to Vhcls_gen_cargo.txt described in Section 4.5.
INPUT\LOS\	FreqAir.314	frequency for air mode [#transports/week]
INPUT\LOS\	FreqCombi.314	frequency for combi mode [#transports/week]
INPUT\LOS\	FreqContainerVessel.314	frequency for container vessels [#transports/week]
INPUT\LOS\	FreqLorry.314	frequency for lorries [#transports/week]
INPUT\LOS\	FreqOtherVessel.314	frequency for other vessels [#transports/week]
INPUT\LOS\	FreqRailFerry.314	frequency for rail ferry [#transports/week]
INPUT\LOS\	FreqRoadFerry.314	frequency for road ferry [#transports/week]
INPUT\LOS\	FreqRoRoVessel.314	frequency for RoRo vessel [#transports/week]
INPUT\LOS\	FreqSystem.314	frequency for system [#transports/week]
INPUT\LOS\	FreqWaggonload.314	frequency for wagonload [#transports/week]
INPUT\LOS\	v101_dist.314	distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_ddist.314	Domestic distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_timeh.314	time matrix for vehicle type 101 [hour]
INPUT\LOS\	v101_xkr.314	extra costs matrix for vehicle type 101 [kSEK]
…	…	…
INPUT\LOS\	v401_dist.314	distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_ddist.314	Domestic distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_timeh.314	time matrix for vehicle type 401 [hour]
INPUT\LOS\	v401_xkr.314	extra costs matrix for vehicle type 401 [kSEK]
INPUT\GENERAL\	CalibrationParameters.txt	Scale factors for sea and ferry times per port area and STAN-group
INPUT\GENERAL\	MAXCAPANDCONSOLEXCEPT.DAT	Vehicle type availability exceptions and exogenous/inherited consolidations rates to be used by the model, see Section 5.5.
INPUT\GENERAL\	LBD_Ratio.dat	Sub mode specific threshold values for the ratio between the EOQ and the vehicle capacity, to restrict the availability of large vehicle types, see Section 5.5.
INPUT\NODES\	nodes.txt	contains general node information for all commodities. For a description of the content of a node file and the units of parameters see Section 4.5
INPUT\NODES\	transferroadroad.txt	Contains a dummy variable indicating whether or not road-road transfers are allowed
INPUT\NODES\	transferroadtrain.txt	Contains a dummy variable indicating whether or not road-train transfers are allowed
INPUT\NODES\	transferroadsea.txt	Contains a dummy variable indicating whether or not road-sea transfers are allowed
INPUT\NODES\	transferroadcombi.txt	Contains a dummy variable indicating whether or not road-combi transfers are allowed

Folder	Filename	Description
CHAINCHOI\	Chainchoi.ctl	Control file for running the BuildChain procedure, containing common settings for all commodities. Structure of file is explained in Section 4.5
CHAINCHOI\	Chainchoi1.ctl	control file for running the ChainChoice procedure for commodity type 1. Structure of file is explained in Section 4.5
CHAINCHOI\	…	
CHAINCHOI\	Chainchoi16.ctl	control file for running the ChainChoice procedure for commodity type 16. Structure of file is explained in Section 4.5
INPUT\COST\	cargo.txt	contains value, holding costs and order costs for all commodities. For a description of the content of the file, and the units of parameters see Section 4.5.
INPUT\COST\	PilotFees.txt	specifies the fees at each terminal [SEK per vehicle]
INPUT\COST\	VHCLS_COM01.TXT	contains commodity 1 parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to Vhcls_gen_cargo.txt described in Section 4.5.
INPUT\COST\	..	...
INPUT\COST\	VHCLS_COM16.TXT	contains commodity 16 parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to Vhcls_gen_cargo.txt described in Section 4.5.
INPUT\LOS\	FreqAir.314	frequency for air mode [#transports/week]
INPUT\LOS\	FreqCombi.314	frequency for combi mode [#transports/week]
INPUT\LOS\	FreqContainerVessel.314	frequency for container vessels [#transports/week]
INPUT\LOS\	FreqLorry.314	frequency for lorries [#transports/week]
INPUT\LOS\	FreqOtherVessel.314	frequency for other vessels [#transports/week]
INPUT\LOS\	FreqRailFerry.314	frequency for rail ferry [#transports/week]
INPUT\LOS\	FreqRoadFerry.314	frequency for road ferry [#transports/week]
INPUT\LOS\	FreqRoRoVessel.314	frequency for RoRo vessel [#transports/week]
INPUT\LOS\	FreqSystem.314	frequency for system [#transports/week]
INPUT\LOS\	FreqWaggonload.314	frequency for wagonload [#transports/week]
INPUT\LOS\	v101_dist.314	distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_ddist.314	Domestic distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_timeh.314	time matrix for vehicle type 101 [hour]
INPUT\LOS\	v101_xkr.314	extra costs matrix for vehicle type 101 [kSEK]
…	…	…
INPUT\LOS\	v401_dist.314	distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_ddist.314	Domestic distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_timeh.314	time matrix for vehicle type 401 [hour]
INPUT\LOS\	v401_xkr.314	extra costs matrix for vehicle type 401 [kSEK]
INPUT\NODES\	transferroadair.txt	Contains a dummy variable indicating whether or not road-air transfers are allowed
INPUT\NODES\	transferroadroadferry.txt	Contains a dummy variable indicating whether or not road-road-ferry transfers are allowed
INPUT\NODES\	Transferseasea.txt	Contains a dummy variable indicating whether or not sea-sea transfers are allowed
INPUT\NODES\	transfercombisea.txt	Contains a dummy variable indicating whether or not Combi-sea transfers are allowed
INPUT\NODES\	transferwagonloadrailferry.txt	Contains a dummy variable indicating whether or not wagonload-railferry transfers are allowed
INPUT\NODES\	transferwagonloadsea.txt	Contains a dummy variable indicating whether or not wagonload-sea transfers are allowed
INPUT\NODES\	transferfeedertrainwagonload.txt	Contains a dummy variable indicating whether or not feeder train-wagonload transfers are allowed
INPUT\NODES\	transfersystemtrainsea.txt	Contains a dummy variable indicating whether or not system train-sea transfers are allowed
INPUT\NODES\	Directsea.txt	Contains a dummy variable indicating whether or not direct sea access is allowed

Folder	Filename	Description
CHAINCHOI\	Chainchoi.ctl	Control file for running the BuildChain procedure, containing common settings for all commodities. Structure of file is explained in Section 4.5
CHAINCHOI\	Chainchoi1.ctl	control file for running the ChainChoice procedure for commodity type 1. Structure of file is explained in Section 4.5
CHAINCHOI\	…	
CHAINCHOI\	Chainchoi16.ctl	control file for running the ChainChoice procedure for commodity type 16. Structure of file is explained in Section 4.5
INPUT\COST\	cargo.txt	contains value, holding costs and order costs for all commodities. For a description of the content of the file, and the units of parameters see Section 4.5.
INPUT\COST\	PilotFees.txt	specifies the fees at each terminal [SEK per vehicle]
INPUT\COST\	VHCLS_COM01.TXT	contains commodity 1 parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to Vhcls_gen_cargo.txt described in Section 4.5.
INPUT\COST\	..	...
INPUT\COST\	VHCLS_COM16.TXT	contains commodity 16 parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to Vhcls_gen_cargo.txt described in Section 4.5.
INPUT\LOS\	FreqAir.314	frequency for air mode [#transports/week]
INPUT\LOS\	FreqCombi.314	frequency for combi mode [#transports/week]
INPUT\LOS\	FreqContainerVessel.314	frequency for container vessels [#transports/week]
INPUT\LOS\	FreqLorry.314	frequency for lorries [#transports/week]
INPUT\LOS\	FreqOtherVessel.314	frequency for other vessels [#transports/week]
INPUT\LOS\	FreqRailFerry.314	frequency for rail ferry [#transports/week]
INPUT\LOS\	FreqRoadFerry.314	frequency for road ferry [#transports/week]
INPUT\LOS\	FreqRoRoVessel.314	frequency for RoRo vessel [#transports/week]
INPUT\LOS\	FreqSystem.314	frequency for system [#transports/week]
INPUT\LOS\	FreqWaggonload.314	frequency for wagonload [#transports/week]
INPUT\LOS\	v101_dist.314	distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_ddist.314	Domestic distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_timeh.314	time matrix for vehicle type 101 [hour]
INPUT\LOS\	v101_xkr.314	extra costs matrix for vehicle type 101 [kSEK]
…	…	…
INPUT\LOS\	v401_dist.314	distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_ddist.314	Domestic distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_timeh.314	time matrix for vehicle type 401 [hour]
INPUT\LOS\	v401_xkr.314	extra costs matrix for vehicle type 401 [kSEK]
INPUT\NODES\	directsystemtrain.txt	Contains a dummy variable indicating whether or not direct system train access is allowed
INPUT\NODES\	directfeedertrain.txt	Contains a dummy variable indicating whether or not direct feeder train access is allowed
INPUT\NODES\	directwagonload.txt	Contains a dummy variable indicating whether or not direct wagonload access is allowed
INPUT\NODES\	containerhandling.txt	Contains a dummy variable indicating whether or not direct containers can be handled
INPUT\LOCKED\	Locked.dat	Contains list of locked transport solutions
INPUT\PWC\	pwc_01.txt	producer demand matrices, with the annual demand (flows from producers or wholesales to consumer between zones) for commodity type 1. [annual demand in tonnes]
INPUT\PWC\	…	…
INPUT\PWC\	pwc_16.txt	producer demand matrices, with the annual demand (flows from producers or wholesales to consumer between zones) for commodity type 16 [annual demand in tonnes]
Buildchain\OUTPUT\	Chains1.dat	logistic chains that were generated for commodity type 1 in the BuildChain program. The structure of this file is explained in Section 4.3

Folder	Filename	Description
CHAINCHOI\	Chainchoi.ctl	Control file for running the BuildChain procedure, containing common settings for all commodities. Structure of file is explained in Section 4.5
CHAINCHOI\	Chainchoi1.ctl	control file for running the ChainChoice procedure for commodity type 1. Structure of file is explained in Section 4.5
CHAINCHOI\	…	
CHAINCHOI\	Chainchoi16.ctl	control file for running the ChainChoice procedure for commodity type 16. Structure of file is explained in Section 4.5
INPUT\COST\	cargo.txt	contains value, holding costs and order costs for all commodities. For a description of the content of the file, and the units of parameters see Section 4.5.
INPUT\COST\	PilotFees.txt	specifies the fees at each terminal [SEK per vehicle]
INPUT\COST\	VHCLS_COM01.TXT	contains commodity 1 parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to Vhcls_gen_cargo.txt described in Section 4.5.
INPUT\COST\	..	...
INPUT\COST\	VHCLS_COM16.TXT	contains commodity 16 parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to Vhcls_gen_cargo.txt described in Section 4.5.
INPUT\LOS\	FreqAir.314	frequency for air mode [#transports/week]
INPUT\LOS\	FreqCombi.314	frequency for combi mode [#transports/week]
INPUT\LOS\	FreqContainerVessel.314	frequency for container vessels [#transports/week]
INPUT\LOS\	FreqLorry.314	frequency for lorries [#transports/week]
INPUT\LOS\	FreqOtherVessel.314	frequency for other vessels [#transports/week]
INPUT\LOS\	FreqRailFerry.314	frequency for rail ferry [#transports/week]
INPUT\LOS\	FreqRoadFerry.314	frequency for road ferry [#transports/week]
INPUT\LOS\	FreqRoRoVessel.314	frequency for RoRo vessel [#transports/week]
INPUT\LOS\	FreqSystem.314	frequency for system [#transports/week]
INPUT\LOS\	FreqWaggonload.314	frequency for wagonload [#transports/week]
INPUT\LOS\	v101_dist.314	distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_ddist.314	Domestic distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_timeh.314	time matrix for vehicle type 101 [hour]
INPUT\LOS\	v101_xkr.314	extra costs matrix for vehicle type 101 [kSEK]
…	…	…
INPUT\LOS\	v401_dist.314	distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_ddist.314	Domestic distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_timeh.314	time matrix for vehicle type 401 [hour]
INPUT\LOS\	v401_xkr.314	extra costs matrix for vehicle type 401 [kSEK]
Buildchain\OUTPUT\	…	…
Buildchain\OUTPUT\	Chains16.dat	logistic chains that were generated for commodity type 16 in the BuildChain program. The structure of this file is explained in Section 4.3

5.3 Output files

The prime result of the `ChainChoice` procedure is logistic chains for commodity flows between an OD pair. These chains include detailed information on the vehicle types that are used and the shipment sizes. Below an overview is given of the output files that are generated for each commodity group. These output data contain disaggregate results. For analysis of aggregate data the post-processing procedure `Extract` is available, that will be described in Chapter 6.

Table 5-2: Overview of ChainChoi output files

DIR	Filename	Description
ChainChoi\OUTPUT\	ChainChoi01STD.out	contains the best chains for each f2f flow for commodity type 1. T content of the file is illustrated in Figure 5-1
ChainChoi\OUTPUT\	…	…
ChainChoi\OUTPUT\	ChainChoi16STD.out	contains the best chains for each f2f flow for commodity type 16. content of the file is illustrated in Figure 5-1
ChainChoi\OUTPUT\	ChaiChoi01data01STD.out	contains additional data (DirectAccess indicator) for t best chains each f2f flow for commodity type 1.
ChainChoi\OUTPUT\	…	
ChainChoi\OUTPUT\	ChaiChoi16data01STD.out	contains additional data (DirectAccess indicator) for the best cha each f2f flow for commodity type 16.
ChainChoi\OUTPUT\	ChaiChoi01data02STD.out	contains additional data (LoadingCosts) for the best chains for ea flow for commodity type 1.
ChainChoi\OUTPUT\	…	
ChainChoi\OUTPUT\	ChaiChoi16data02STD.out	contains additional data (LoadingCosts) for the best chains for ea flow for commodity type 16.
ChainChoi\OUTPUT\	ChaiChoi01data03STD.out	contains additional data (WaitTimeCosts) for the best chains for flow for commodity type 1.
ChainChoi\OUTPUT\	…	
ChainChoi\OUTPUT\	ChaiChoi16data03STD.out	contains additional data (WaitTimeCosts) for the best chains for flow for commodity type 16.
ChainChoi\OUTPUT\	ChaiChoi01data04STD.out	contains additional data (Main[14] Utilization Rate) for the best cha each f2f flow for commodity type 1.
ChainChoi\OUTPUT\	…	
ChainChoi\OUTPUT\	ChaiChoi16data04STD.out	contains additional data (Main Utilization Rate) for the best chain each f2f flow for commodity type 16.
ChainChoi\OUTPUT\	ChaiChoi01data05STD.out	contains additional data (NrVhcls main leg) for the best chain for flow for commodity type 1.
ChainChoi\OUTPUT\	…	
ChainChoi\OUTPUT\	ChaiChoi16data05STD.out	contains additional data (NrVhcls main leg) for the best chains fo f2f flow for commodity type 16.
ChainChoi\OUTPUT\	ChaiChoi01data08STD.out	contains information for the best chains on the conditions that ar satisfied in the avail-function for commodity type 1.
ChainChoi\OUTPUT\	…	
ChainChoi\OUTPUT\	ChaiChoi16data08STD.out	contains information for the best chains on the conditions that ar satisfied in the avail-function for commodity type 16.
ChainChoi\OUTPUT\	Locked01.log	Contains a list of keys, corresponding to relations with fixed trans solutions, for commodity 1
LOG\	Log01.log	logfile with description of progress for commodity type 1
LOG\	…	…
LOG\	Log16.log	logfile with description of progress for commodity type 16
CHAINCHOI\OUTPUT\	ChaiChoi01STD.rep	contains an aggregate report for commodity 1
CHAINCHOI\OUTPUT\	…	…
CHAINCHOI\OUTPUT\	ChaiChoi16STD.rep	contains an aggregate report for commodity 16
CHAINCHOI\OUTPUT\	VhclRep01STD.rep	contains an aggregate costs report for commodity 1
CHAINCHOI\OUTPUT\	…	…
CHAINCHOI\OUTPUT\	VhclRep16STD.rep	contains an aggregate costs report for commodity 16
CHAINCHOI\OUTPUT\COVO	Consol01_D	contains the level of consolidation of commodity type 1 between pairs for transport mode D
CHAINCHOI\OUTPUT\COVO	…	
CHAINCHOI\OUTPUT\COVO	…	
CHAINCHOI\OUTPUT\COVO	consol16_R	contains the level of consolidation of commodity type 16 betwee pairs for transport mode R

[14] The main leg refers to the longest leg of a chain

DIR	Filename	Description
CHAINCHOI\OUTPUT\COVO	Volume01_D	contains the total volume of commodity type 1 between r transport mode D
CHAINCHOI\OUTPUT\COVO	…	
CHAINCHOI\OUTPUT\COVO	…	
CHAINCHOI\OUTPUT\COVO	Volume 16_R	contains the total volume of commodity type 16 between transport mode R

Below the output is illustrated for a sample from an output. Again, the chosen chains are illustrated for origin/destination pair 711400 and 716000. The output file shows 4 f2f commodity flows. The second column shows the number of firms in that specific firm size relation group (3x3). In the first group there are 3 f2f flows. Each flow has a size of 1.4191 tonnes. The Prob-column has been added with the introduction of cost variation in version 1.1.3. If no cost variation is assumed (as in prior versions), the Prob-column will always be equal to 1.0. If there is some cost variation, the Prob-column will contain a value between 0 and 1.0. The remaining fraction will be assigned to the second best transport chain solution that can be found in the ChainChoi17_02.out file. The best frequency that was chosen in the optimization proofs to be 2 shipments. Next the transport and total costs are specified. The chosen chain type is 'A' (=Lorry). The best vehicle type within this transport mode is 104 (=heavy lorry 25-40 ton).

Figure 5-1: Sample of results in output file ChainChoi17_01.out, with header description

Key	Nrelat	Ton2 reciev. [tonne]	Prob	Best freq [#] [SEK]	Best transp. costs [SEK]	Best cost total	Best chain type	orig node	dest node	Best vehicle type	N of vehicles [#vehicles]
33	3	1.4191	1.0	2	1559.4	2978.1	A	711400	716000	104	1
34	4	1.1753	1.0	2	1550.5	2772.9	A	711400	716000	104	1
35	3	0.9743	1.0	2	1543.2	2603.8	A	711400	716000	104	1
36	2	1.6139	1.0	2	1566.4	3142.0	A	711400	716000	104	1

Each line in the output file represents one chose logistic chain. Please note that all chosen chains in Figure 5-1 only contain one leg. If a chosen chain consists of multiple legs, the output file has extra columns for a transfer node node, the best vehicle type, and N of vehicles, for each additional link.

The report files (*.rep) contain aggregate reports for a commodity type. It reports statistics such as number of shipments and vehicles average load factors, and average distances. Statistics are aggregated to vehicle type, and to chain types. The structure of the file cannot be illustrated in a figure, but the report file contains descriptions that explain the labels of the attributed listed in the files.

The log files (*.log) contain warnings that occurred during execution of the module for the specific commodity type. A warning is written when no chains are available for an OD pair but for which the PWC matrix does specify an OD flow.

5.4 Description

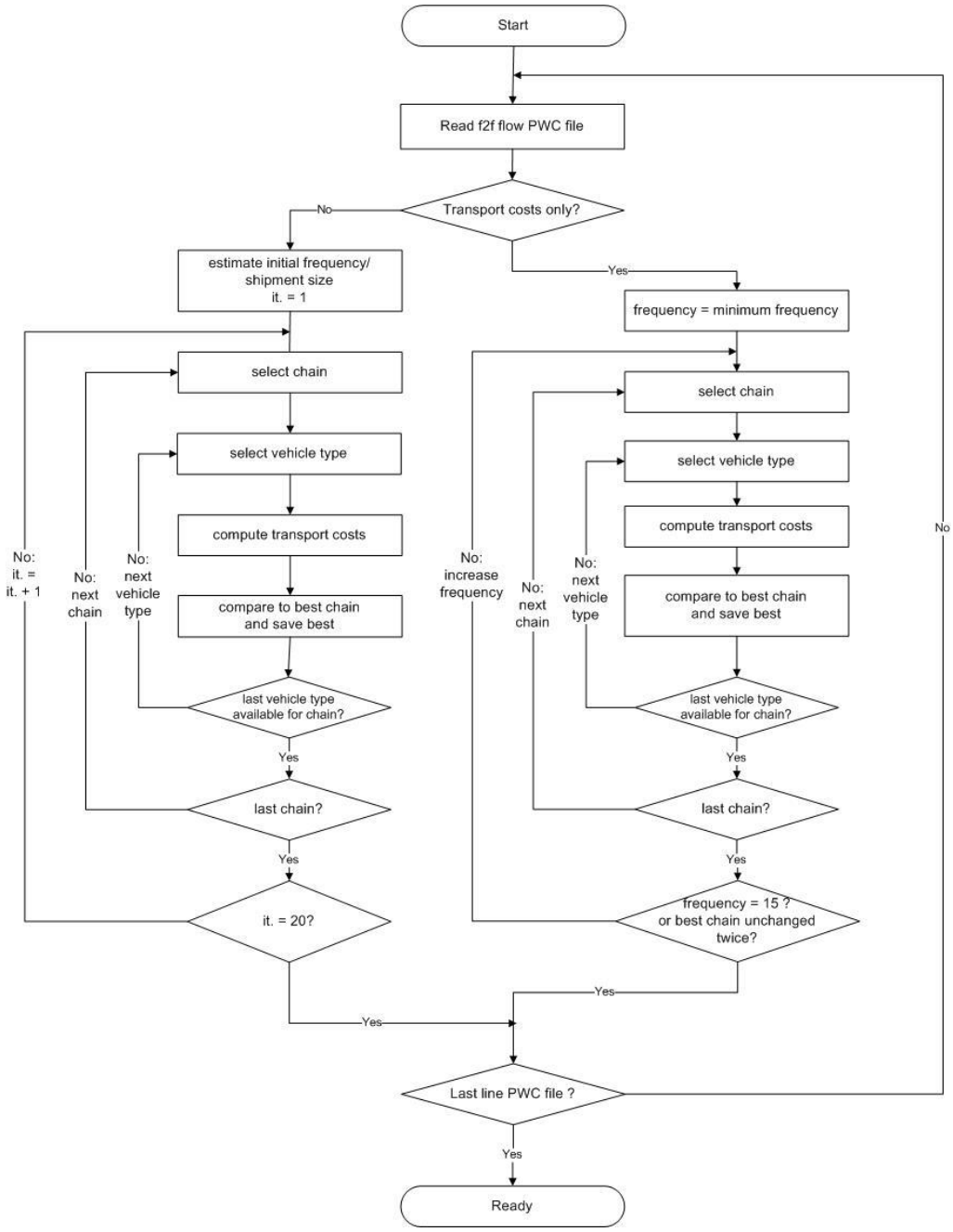
The `ChainChoice` procedure optimizes the logistic costs. This optimization is based on a choice for one of the available transport chains, and a choice for shipment sizes. The optimal shipment sizes are determined by optimizing the

consolidation and distribution of commodity flows at transfer points. This optimization is reached by an iterative procedure as described in Chapter 3 and visualised in Figure 3-2.

In the first iteration of the model, that also includes generating the available transport chains, we use a consolidation factor of 75% (this is just a starting point): for all consolidated legs of a transport chains (that is legs coming after a consolidation centre) we assume that 75% of the vehicle capacity is used, and the shipment studied only has to pay a costs proportional to its share in this total load. This is needed to calculate the total logistics cost of transport chains that use rail, sea, airplane or consolidated road vehicles. In the second iteration, the consolidation factor is based on the potential for consolidation and observed terminal throughput data, as described below. In the third iteration, we use the OD legs from the previous model run as starting point for the level of consolidation between terminal pairs.

The flowchart of the procedure is illustrated in Figure 5-2. For a more detailed description of the `ChainChoice` procedure see the Method report on the logistics module (Significance, 2010).

Figure 5-2: Flowchart ChainChoice procedure



5.5 Controlling the ChainChoice procedure

The parameters and input files for the ChainChoice procedure are controlled with commodity specific control files. These control files are case sensitive, because the mode identifiers used within the logistic are case sensitive (for example, mode D and d are different combi train modes). This section describes the usage of this control file and gives an overview the input files. Below an example is given for the control file for commodity 1.

Figure 5-3: Example of a `ChainChoice` control file: [`Chainchoi1.ctl`]

```
INCL=ChainChoi_Common.ctl
COMMODITY=01
OPTIP=0
OPTIW=0
MFREQ=1
DIRACC=1,0,0,1,0,0,1,1,1,1
CSTVAR=0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1
PWC=..\LogMod\Input\PWC\2016\PWC_01.txt
CHAINS=..\BuildChain\OUTPUT\Chains01.dat
VHCL=.\Cost\VHCLS_COM01.TXT
VHCLA=104,105
VHCLB=101,102,103
VHCLC=104,105
VHCLc=106
VHCLD=201
VHCLd=210
VHCLE=202
VHCLF=207,208
VHCLG=202
VHCLH=207,208
VHCLh=212
VHCLI=204
VHCLi=211
VHCLM=305,306,307,308,309,310,311,312,313,314,315,316,317
VHCLN=315,316
VHCLO=317
VHCLP=318,319,320
VHCLQ=321
VHCLR=401
VHCLT=205
VHCLU=206
VHCLV=322
VHCLW=322
VHCLf=212
VHCLX=106
LOG=..\log\log01.log
CHAINSRCM=..\BuildChain\OUTPUT\Chains01_RCM.dat
LOCKED=..\Input\Locked\Locked_2017.dat
LOCKEDLOG=..\ChainChoi\OUTPUT\Locked01.log
```

Explanation of parameters:

INCL	: reference to the control file that contains the common settings for all commodities. An example of this file is given in Figure 5-4.
COMMODITY	: commodity code
OPTIP	: sets the producer costs in the optimization procedure. 0 = include all costs, 1 = include transport costs only
OPTIW	: sets the wholesale costs in the optimization procedure. 0 = include all costs, 1 = include transport costs only
MFREQ	: minimum frequency that is used when the default frequency is not available in the vehicle costs file (path to this file is specified in parameter VHCL). These frequencies are used as an initial value in the shipment size optimization procedure. [#transports/week]
DIRACC	: switches to make direct access available for specific f2f sizeclass combinations. 0= unavailable, 1 = available. The first character is for singular flows. The second character for small to small firms. The last character for large to large firms
CSTVAR	: Parameter used to calculate the cost variance per sub cell.
PWC	: path to file with producer demand matrices for the specific commodity.
CHAINS	: path to file with logistic chains for each chain type available to the respective commodity type. This file is generated by the BuildChain procedure. For a description of this file see section 4.3.
VHCL	: path to file with vehicle data for the specific commodity type. Parameters to the logistic cost function (equation 1). This input file is described in detail in section 4.5.
VHCLA	: available vehicle type numbers for mode A.
…	
VHCLX	: available vehicle type numbers for mode X.
LOCKEDLOG	: Path for output file containing a list of keys, corresponding to relations with locked transport solution
LOG	: path for the logfile.

Figure 5-4: Example of a common `ChainChoice` control file: [`Chainchoi.ctl`]

```
INCL=..\Input\General\OtherCostMatters.ctl
LOGCTL=1
LOGFLS=1
LOGCST=1
STDLST=5,2
RCMLST=2,2
STDDATA=6,7
RCMDATA=6
CONSOL=0.05,0.95
CONSOLA=0.40,0.60
CONSOLB=0.40,0.60
CONSOLc=0.40,0.60
CONSOLC=0.40,0.60
CONSOLS=0.10,0.35
CONSOLX=0.40,0.60
CONSOLd=0.60,0.95
CONSOLD=0.60,0.95
CONSOLE=0.60,0.95
CONSOLF=0.60,0.95
CONSOLf=0.60,0.95
CONSOLG=0.60,0.95
CONSOLh=0.60,0.95
CONSOLH=0.60,0.95
CONSOLi=0.66,0.98
CONSOLI=0.66,0.98
CONSOLT=0.66,1.00
CONSOLU=0.66,1.00
CONSOLJ=0.05,0.60
CONSOLK=0.05,0.60
CONSOLL=0.05,0.60
CONSOLM=0.05,0.60
CONSOLN=0.05,0.60
CONSOLO=0.05,0.60
CONSOLP=0.15,0.90
CONSOLQ=0.15,0.90
CONSOLV=0.05,0.60
CONSOLW=0.05,0.60
CONSOLR=0.15,0.80
TYPES=..\Input\chaintype.lis
NODES=.\Nodes
PILOTFEES=.\Cost\pilotfees.txt
CARGO=.\Cost\cargo.txt
OUTDIR=.\OUTPUT
CONSOLDIR=..\ChainChoi\OUTPUT\CoVo
LOSDIR=.\LOS
LOSFAC=..\Input\General\CalibrationParameters.txt
LBD_Ratios=..\Input\General\LBD_Ratio.dat
Cap_Except=..\Input\General\MaxCapANDConsolExcept.dat
EVALALL=1000
WRTLOS=0
ALL_LORRY_TYPE_CONSOL=1
MINIMUM_ANNUAL_TONNE_DEMAND_4_FREQ_OPTIMIZE=100.0
INP=..\Input
ASEKINP=..\Input_CBA
```

Explanation of parameters:

INCL	: reference to the control file that contains the common settings for different programs within the logistic model. An example of this file is given in Figure 4-7.
LOGCTL	: Indicator (0/1) that determines whether or not CTL file settings will be logged in the log file.
LOGFLS	: Indicator (0/1) that determines whether or not input file information will be logged in the log file.
LOGCST	: Indicator (0/1) that determines whether or not cost parameters will be logged in the log file.
STDLST	: sets the number of solutions (best, second best, …) included in the output file and data output files respectively, for a run with the standard logistic model
RCMLST	: sets the number of solutions (best, second best, …) included in the output file and data output files respectively, for a run with the RCM model
STDDATA	: comma separated list of the data output files generated for a run with the standard logistic model
RCMDATA	: comma separated list of the data output files generated for a run with the RCM model
STUFF	: costs for stuffing and stripping of containers at the origin and destination of a chain [SEK per tonne]
INTEREST	: Interest rate used in cost calculations [%/year]
CONSOL	: gives the default upper and lower bounds for the consolidation factors ranking output
CONSOL<mode>	: gives the default upper and lower bounds for the consolidation factors ranking output for mode <mode>
ALL_LORRY_TYPE_CONSOL	: 0/1 switch that determines whether or not consolidation is allowed for all lorry types[15]
MINIMUM_ANNUAL_TONNE_DEMAND_4_FREQ_OPTIMIZE	: Minimum demand in annual tonnes on a firm-to-firm relation, required for shipment frequency optimization
TYPES	: Path to file containing all available chain types

[15] Regardless of this setting a separate mode S (consolidated heavy lorry) is distinguished in the mode. Because this mode is only allowed between terminals and on international relations, a consolidation factor range may be used that differs from the other lorry modes (A,B,C).

NODES	: Path to directory containing the nodes input files. Each nodes file contains a single node variable for all commodities. The file names of the nodes files are fixed.
PILOTFEES	: path to file with pilot fees. Parameters to the logistic cost function (equation 1). For a description see below.
CARGO	: path to file with cargo values, holding costs and order costs. Parameters to the logistic cost function (equation 1). This input file is described in detail in section 4.5.
LOSDIR	:Path to directory containing the Level of service files. The level of service files have fixed names, referring to the applicable vehicle type.
LOSFAC	: Path to input file containing scale factors for ferry and vessel times, per port area and STAN-group
LBD_Ratios	: path to input file containing sub mode specific threshold values for the ratio between the EOQ and the vehicle capacity, as to restrict the availability of large vehicle types
Cap_Except	: Path to input file containing a list with vehicle type availability exceptions and the exogenous / inherited consolidations rates to be used within the model
EVALLALL	: Threshold value for the annual transport volume above which all shipment frequencies are evaluated
LOCKED	: Path for input file containing a list of locked transport solutions
OUTDIR	:Path to output directory. All output files have fixed names.
CONSOLDIR	: Path to consolidation output directory. All output files have fixed names.
SELECT	: path to input file containing the collection of relations that should be included in the `detailed cost log`.

Explanation of input files

Except for the three files specified below, all input files were already discussed in Chapter 4. For this description see section 4.5.

LOCKED

The input file with locked transport solutions contains a header line, followed by one line for each locked solution, containing the following columns:

Table 5-3: Contents of locked solutions file:

Column	Parameter	Description
1	Commodity	Commodity for which this locked solution applies
2	From node	Origin of locked transport solution
3	To Node	Destination of locked transport solution
4	SubCell	Subcell for which this locked solution applies
5	Freq	Shipment frequency for the locked transport solution, or -1 if the model should determine the shipment frequency
6	Mode	Mode of the locked transport solution
7	NNodes	Number of nodes in route locked transport solution
8	Node *1*	First node in route locked transport solution
…	…	…
	Node *NNodes*	: Last node in route locked transport solution

LBD_Ratios

The input file with threshold values for the ratios between the EOQ and the vehicle type capacities contains a header line, followed by one line for each sub mode, containing the following columns:

Column	Parameter	Description
1	Sub mode	Sub mode for which the threshold value applies
2	Com 1 threshold	Threshold value for commodity 1
…	…	…
17	Com 16 threshold	Threshold value for commodity 16

Cap_Except

The input file with vehicle type availability exceptions and exogenous/inherited consolidations rates contains a header line, followed by one line for each exception, containing the following columns:

Column	Parameter	Description
1	Orig	Origin node number
2	Dest	Destination node number
3	SubMode	Sub mode for which the exception applies
4	CMD_SLCT	Selection of commodities for which the exception applies. A value of 0 indicates all commodities are selected. To select per

		commodity a string of 16 “0” or “1” characters should be given, corresponding to the 16 commodities.
5	VEHCAP	Threshold value for the vehicle type availability. All vehicle types with a capacity below the threshold are available.
6	CONSOLRATE	Consolidation rate to be used

CHAPTER 6 Extract procedure

6.1 Introduction

The `Extract` procedure generates vehicle demand matrices. This chapter describes the procedure and gives an overview of the input files it uses and what output files are produced. Please note that the output reports are different in the Swedish and Norwegian implementation. The Swedish vehicle matrices are post processed outside of the logistic module. To avoid double processing some output reports were excluded from the Swedish implementation.

6.2 Input files

The extract procedure loops through all vehicle types. During its execution it uses the chosen logistic chains from the `ChainChoice` procedure, the LOS files for each vehicle type and a list of all nodes in the network. Below an overview is presented of all files that are used in the `Extract` procedure.

Table 6-1: Overview of Extract input files

Folder	Filename	Description
\EXTRACT\	emptyfrac.dat	Emty vehicle fractions per vehicle type
\EXTRACT\	ExtractSTD.ctl	control file for running the extract procedure for the standard logistics model. Structure of file is explained in Section 0
INPUT\NODES\	AllNodes.dat	contains a list of all nodes with their ID's and names. Contains no additional information on transfer options.
INPUT\LOS\	v101_dist.314	distance matrix for vehicle type 101 [km]
…	…	…
INPUT\LOS\	v401_dist.314	distance matrix for vehicle type 401 [km]
CHAINCHOI\OUTPUT\	Chainchoi1STD.out	contains the best chains for each f2f flow for commodity type 1. The content of the file is illustrated in Figure 5-1
…	…	…
CHAINCHOI\ OUTPUT\	Chainchoi16STD.out	contains the best chains for each f2f flow for commodity type 16. The content of the file is illustrated in Figure 5-1

6.3 Output files

The Extract procedure generates vehicle matrices (see overview below).

Table 6-2: Overview of Extract output files

Folder	Filename	Description
\EXTRACT\ OUTPUT\	ODVhcl101.314	Vehicle matrix for vehicle type 101. Structure of file is explained below
\EXTRACT\ OUTPUT\	…	…
\EXTRACT\ OUTPUT\	ODVhcl401.314	Vehicle matrix for vehicle type 401. Structure of file is explained below

The vehicle demand matrices are in EMME/2 format. The file specifies the location, date and time of execution, and the control file that was used. The fourth line in the file gives the matrix ID code (in the example below: mf39). A data line contains three pieces of information: origin node, destination node, and number of vehicles. The content of a typical output file is illustrated in Figure 6-1.

Figure 6-1: Sample of output file `od_vhcl101.314` **the vehicle matrices for vehicle type 101.**

```
c D:\SIKA\EXTRACT\Extract.exe 06/12/2007 10:35:02
c Control file: extract101.ctl
t matrices
a matrix=mf11 start 0
711400 711401:27.0000
711400 712000:664.4220
711400 712500:40.5000
711400 712600:165.0000
711400 712700:328.5000
711400 712800:197.6600
711400 713600:1971.5742
711400 714000:224.5102
711400 716000:6.0000
711400 718000:1545.0000
711400 718100:1286.8432
711400 718101:38.0000
711400 718200:24.0000
711400 718400:148.5000
711400 718600:4.5000
711400 718700:6.0000
711400 718800:5385.9671
711400 719200:966.0669
711400 730500:1.5000
……
```

6.4 Controlling the Extract procedure

The extract procedure is controlled with control files. These control files are used to set the parameters and paths to the necessary input and output files. Below the control file for a run of the standard logistics model is illustrated. The chosen chains for all commodity types are all included in the vehicle matrices for a specific vehicle type.

Figure 6-2: Example of an Extract control file: [`Extract.ctl`**]**

```
RUN=STD
EMPTY=1
```

```
ERRLOG=..\LOG\Error.log
EMPTYFR=emptyfrac.dat
NFLOWS=16
NODES=..\Input\Nodes
LOSDIR=..\Input\LOS
OUTDIR=.\OUTPUT
FLOW1=..\chainchoi\OUTPUT\chainchoi01STD.out
FLOW2=..\chainchoi\OUTPUT\chainchoi02STD.out
FLOW3=..\chainchoi\OUTPUT\chainchoi03STD.out
FLOW4=..\chainchoi\OUTPUT\chainchoi04STD.out
FLOW5=..\chainchoi\OUTPUT\chainchoi05STD.out
FLOW6=..\chainchoi\OUTPUT\chainchoi06STD.out
FLOW7=..\chainchoi\OUTPUT\chainchoi07STD.out
FLOW8=..\chainchoi\OUTPUT\chainchoi08STD.out
FLOW9=..\chainchoi\OUTPUT\chainchoi09STD.out
FLOW10=..\chainchoi\OUTPUT\chainchoi10STD.out
FLOW11=..\chainchoi\OUTPUT\chainchoi11STD.out
FLOW12=..\chainchoi\OUTPUT\chainchoi12STD.out
FLOW13=..\chainchoi\OUTPUT\chainchoi13STD.out
FLOW14=..\chainchoi\OUTPUT\chainchoi14STD.out
FLOW15=..\chainchoi\OUTPUT\chainchoi15STD.out
FLOW16=..\chainchoi\OUTPUT\chainchoi16STD.out
LOG=..\LOG\Extract_STD.log
```

The parameters from the control file are described below.

`RUN`	: label added at the tail of the output file names. It is used to indicate the type of model application (STD/RCM)
`EMPTY`	: switch for activating/deactivating the calculation of empty vehicles. 0 = empty vehicles are not included, 1 = empty vehicles are derived. The procedure for determining the number of empty vehicles is described below.
`NFLOWS`	: the number of commodity flows that is analysed for vehicle flows. In a standard application this parameter is set to 34, and all commodity flows that are implemented in the current Swedish model are analysed. If a user might decide to determine vehicle matrices for one specific commodity type, the parameter `NFLOWS` is set to 1. In this setting only one chosen chain path has to be specified: parameter `FLOW1`. The paths `FLOW2` to `FLOW34` are abundant in such a case.
`NODES`	: path to file with node information on all nodes. This input file is similar to the node file described in detail in section 4.5.
`EMPTYFR`	: path to file with empty vehicle fractions.
`LOSDIR`	: path to the directory containing the level of service files, specifically the vehicle specific distance files
`FLOW1`	: path to file with chosen logistic chains for commodity type 1. This input file is described in detail in section 5.3.
…	…
`FLOW16`	: path to file with chosen logistic chains for commodity type 16. This input file is described in detail in section 5.3.

`OUTDIR` : path to the directory where the output files are written

`LOG` : path to the log file

Explanation of input files:

The emptyfrac.dat file contains the distance dependent empty vehicle fractions per vehicle type:

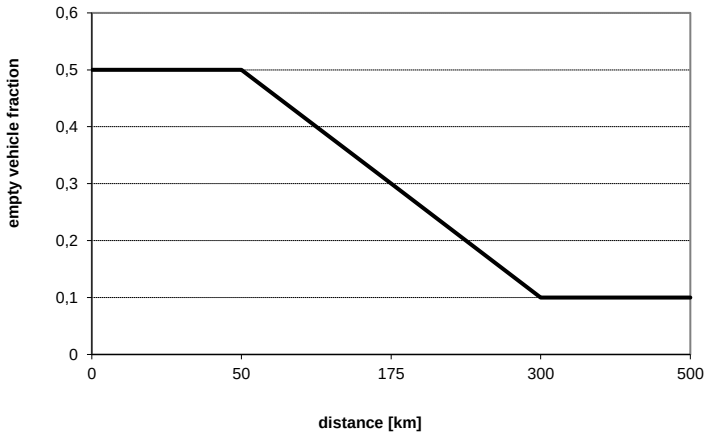
Table 6-1: Contents of `emptyfrac.dat`:

Column	Parameter	Description
1	Vehicle Type	Vehicle type number
2	Distance	Distance threshold, see explanation below
3	Fraction	Empty vehicle fraction

Below the lowest distance threshold and above the highest distance threshold the empty vehicle fractions are constant. In between two distance thresholds a linear interpolation is used. For example, the following lines in emptyfrac.dat:

101 50 0.5
101 300 0.1

imply that the empty vehicle fraction for vehicle type 101equals 0.5 below 50 km and 0.1 above 300 km. In between 50 km and 300 km there is a linear decrease of the empty vehicle fraction from 0.5 to 0.1. This distance function is visualised below.



Empty vehicles

Below the ASYM threshold the empty vehicle flow is calculated by multiplying the empty vehicle fraction (*E*) with the loaded vehicle flow in the reverse direction:

Empties(o,d) = *E* * Loaded(d,o)

Above the ASYM threshold the empty vehicle flow consists of two components. First of all, vehicles may return empty because the reverse flow is unbalanced. On top of that, a fraction of the vehicles will always return empty, even if a flow is perfectly balanced with its reverse flow. This fraction is specified in the emptyfrac.dat file:

Empties(o,d) = *E* * Min(Loaded(o,d),Loaded(d,o)) + Max(Loaded(d,o)-Loaded(o,d),0)

References

Henrik Edwards, Christer Anderstig, Daniel Pettersson och Adriana Huelsz-Prince (2019) Samgods PWC-matriser 2016 och 2040; Sweco Society och WSP, Stockholm.

RAND Europe and SITMA (2005) The development of a logistics module in the Norwegian and Swedish national model systems, deliverable 4: final progress report on model development, PM-1968-SIKA, RAND Europe, Leiden.

RAND Europe and SITMA (2006) Documentation and clarification of deliverable 4 and the associated program delivery for a logistics module in the Norwegian and Swedish national model systems, deliverable 4a, PM-2055-SIKA, RAND Europe, Leiden.

Significance (2007) Technical report on the further development of a logistics module in the Norwegian and Swedish national freight model systems, Deliverable 5 for the Samgods group and the working group for transport analysis in the Norwegian national plan, Significance, Leiden.

Significance (2013) Method Report – Logistics Model in the Swedish National Freight Model System (Version 2.1), Deliverable 6B for Trafikverket, Significance, The Hague.

Appendix A: Rail Capacity Management

A.1 Introduction

A transport demand is defined by a certain amount of goods needing to be transported from an origin to a destination. The purpose of the logistic model is to find the most economic transport chain for each transport demand within Sweden as well as from Sweden to destinations abroad, and to Sweden from origins abroad.

The standard logistic model reads all transport demands from PWC matrices. In these matrices all transport volumes, origins and destinations are given. In a first step the program BuildChain finds all possible transport chains. In the next step the program ChainChoi optimizes each of the candidate chains, and selects the most economic one. It also lists the best alternative transport chains.

One limitation of the standard logistic model is that it does not take into account the capacity limitations of rail transport chains. For this purpose the logistic model for rail capacity management, LogMod4RCM, has been developed. At the end of the rail capacity modelling process the new partial solutions needs to be merged with the original solutions from the standard logistic model, and all statistics recalculated. That is the task for the LP2CC program.

A.2 LogMod4RCM

Program purpose

A transport demand is defined by a certain amount of goods needing to be transported from an origin to a destination. The purpose of the logistic model is to find the most economic transport chain for each transport demand within Sweden as well as from Sweden to destinations abroad, and to Sweden from origins abroad.

The standard logistic model reads all transport demands from PWC matrices. In these matrices all transport volumes, origins and destinations are given. In a first step the program BuildChain uses Dijkstras algorithm to find possible transport chains. In the next step the program ChainChoi optimizes each of the candidate chains, and selects the most economic one. It also lists the best alternative transport chains.

One limitation of the standard logistic model is that it does not take into account the capacity limitations of rail transport chains. For this purpose the logistic model for rail capacity management, LogMod4RCM, has been developed to support a linear programming model generating optimal combination of solutions that satisfy the rail capacity constraints.

Rail capacity management

The LogMod4RCM model uses the same BuildChain and ChainChoi executables as the standard logistic model, but running in a special RCM-mode that is being activated by passing a /RCM argument on the command line. Within this RCM-mode the programs find the most economic transport chain in much the same

way as the standard logistic model, with two major differences: it only calculates the cost for those transport demands that uses rail in the solutions from the standard logistic model, and it adds a marginal cost, extracted from an LP-model by the Java program MPS, to the cost for rail transport links facing capacity constraints (capacity bottle necks). The result from these two programs is an additional set of solutions options (columns in the LP-model) facing the marginal costs of the capacity constraints. These are added to the LP-model to provide an improved set of options enabling the model to identify better solutions.

Program flow

1. BuildChain finds a set of cost-minimizing transport chains (within predefined limits)
2. ChainChoi calculates the most economic alternatives
3. The LP-model calculates the marginal cost for congested links (these are extracted by MPS)
4. BuildChain4RCM finds a set of cost-minimizing transport chains for rail given marginal costs on constraining capacities
5. ChainChoi4RCM calculates the most economic alternative

Function

LogMod4RCM are basically the same programs as the standard logistic model, with a few exceptions, to calculate costs and find the most economic transport chain. The difference lies in the selection of transport demands, and in the added marginal cost for certain rail links. In the standard logistic model the transport demands are read from PWC matrix files. In LogMod4RCM loops over all transport chains from the standard LogMod output that uses rail. These are specified in the file named by the JLIST control file parameter. The transport demands are then read from an output file from the standard logistic model solution, ChainChoiXX_data_06.out. However, the demand might as well have been read from the PWC-matrices.

Program control

Command line parameter

Both BuildChain and ChainChoi in RCM-mode read the control file name from the command line. The parameter RDVOLUMES=YES indicates that volumes and consolidation factors are read from the CONSOLXX_Y.314 and VOLUMEXX_Y.314 files written by standard LogMod, like the final standard ChainChoi iteration.

Bat file example:

```
cd buildchain
time /t
BuildChain4.exe buildchain01.ctl /RCM
cd..
cd ChainChoi
time /t
ChainChoi.exe chainchoi01.ctl /RDVOLUMES=YES /RCM
cd..
time /t
```

Control file parameters

The programs uses the ctl files from standard LogMod with a few new parameters added. These parameters are used by both BuildChain and ChainChoi.

CHAINSRCM specifies the BuildChain output chains file name in RCM-mode, also used as input for ChainChoi.
JLIST specifies the JLISTA.dat file, containing indexes and keys for all chains to recalculate.

Example:

CHAINSRCM=OUTPUT\Chains01_RCM.dat
JLIST=..\RCM\JLISTA.DAT

I/O

Input files

Nodes, costs and vehicle data are read from the same files that standard LogMod uses.

Output files

LogMod4RCM generates the same set of files as standard LogMod, with the label RCM in the file name before the file extension, as in ChainChoi01RCM.out. These files only contain the recalculated chains. The exception is that only one chain is generated for each PWC demand.

A.3 LP2CC

Program purpose

At the end of the rail capacity modelling process the new partial solutions needs to be merged with the original solutions from the standard logistic model, and all statistics recalculated. That is the task for the LP2CC program. It generates a complete set of LogMod output files where the chains that are recalculated in RCM replaces the original chains.

Optionally the program can also be used to generate output for cost benefit analysis, CBA.

Function

LP2CC uses the transport solutions found by the standard logistic model and the alternative solutions calculated by the rail capacity modelling process. It then recalculates all costs and statistics, and generates a complete new set of output files for the merged solution. The chains are read from the ChainChoiXXLPX.out files, where the optimal LP-solution columns are inserted. Some additional data are also read from ChainChoiXX data06.out. These data are then entered into the ChainChoi optimization process as the best chain, and recalculated using the calculation routines from ChainChoi.

Program control

Command line parameter

The first command line parameter is the control file name. Optionally the program also takes the command line parameter /CBA. This parameter tells the program to generate output for cost benefit analysis. In this mode the program uses ASEK input data, as explained below.

Command file example

cd ChainChoi

LP2CC.exe chainchoi01.ctl

cd..

or

cd ChainChoi

LP2CC.exe chainchoi01.ctl /CBA

cd..

Control file parameters

The LP2CC program uses the ctl files also being used by CHAINCHOI. Compared to a run with the standard logistic model a few parameters are replaced by special LP2CC parameters:

ChainChoi parameter	LP2CC parameter
STDLST	XTDLST
STDDATA	DATA

Within CBA-mode a few additional parameters are replaced by special CBA parameters;

Standard model parameter	CBA model parameter
INTEREST	INTERESTCBA
STUFF	STUFFCBA
OUTDIR	CBAOUT
INPDIR	ASEKINP

I/O

Input files

Nodes, cost and vehicle data are read from the same files as ChainChoi uses

Input for chain calculation is read from ChaiChoiXXLPX.out and ChainChoiXXdata06.out. For the CBA analysis from ChaiChoiXXFIN.out and ChainChoiXXdata06FIN.out are used instead.

Output files

The output generated is the full set of output from LogMod, with the label FIN before the file name extension. The exception is that only one chain is generated for each PWC demand.

For CBA mode the output is written in an output folder named OutputCBA. This output directory is placed next to the original output folder …\ChainChoi\OUTPUT.

Appendix B: Dimensions in the model

Table A-1: Overview of commodity types:

Nr	Products from agriculture, forestry and fishing
1	Coal, crude oil and natural gas
2	Ore, other products of extraction
3	Food, beverages and tobacco
4	Textiles, clothing, leather and leather goods
5	Wood and articles of wood and cork (excl. Furniture), pulp, paper and paper products, printed matter
6	Coal and refined petroleum products
7	Chemicals, chemical products, synthetic fibers, rubber and plastic products and nuclear fuel
8	Other non-metallic mineral products
9	Metal products excluding machinery and equipment
10	Machinery and instruments
11	Transport equipment
12	Other manufacturing ex furniture
13	Household waste, other waste and return raw material
14	Round timber
15	Air transport goods
16	Products from agriculture, forestry and fishing

Table A-2: Overview of vehicle type numbers, and aggregate modes for container transport and non-container transport.

Aggregate mode		ModeNr	VhclNr	Vehicle type
Containers	Heavy lorry	A	104	Lorry HGV 25-40 ton
			105	Lorry HGV 25-60 ton
	Extra heavy lorry	X	106	Lorry HGV 74 ton
	Kombi train	D	201	Kombi train
	Long kombi train	d	210	Long kombi train
	Feeder train	E	202	Feeder/shunt train
	Wagonload train	F	207	Short wagon load train
			208	Medium wagonload train
			209	Long wagonload train
	Long Wagonload train	f	212	Long wagonload train
	Direct Sea	J	301	Container vessel 5 300 dwt[1]
			302	Container vessel 16 000 dwt[1]
			303	Container vessel 27 200 dwt[1]
			304	Container vessel 100 000 dwt[1]
			305	Other vessel 1 000 dwt
			306	Other vessel 2 500 dwt
			307	Other vessel 3 500 dwt
			308	Other vessel 5 000 dwt
			309	Other vessel 10 000 dwt
			310	Other vessel 20 000 dwt
			311	Other vessel 40 000 dwt
			312	Other vessel 80 000 dwt
			313	Other vessel 100 000 dwt
			314	Other vessel 250 000 dwt
			315	Ro/ro vessel 3 600 dwt
			316	Ro/ro vessel 6 300 dwt
			317	Ro/ro vessel 10 000 dwt
	Feeder vessel	K	301	Container vessel 5 300 dwt
			315	Ro/ro vessel 3 600 dwt
			316	Ro/ro vessel 6 300 dwt
	Long-Haul vessel	L	303	Container vessel 27 200 dwt
			304	Container vessel 100 000 dwt
			317	Ro/ro vessel 10 000 dwt
	IWW	V	322	IWW-vessel
Non-Containers	Light Lorry	B	101	Lorry light LGV, ≤ 3,5 ton

			102	Lorry medium 3,5-16 ton
			103	Lorry medium16-24 ton
	Heavy lorry	C / S[16]	104	Lorry HGV 25-40 ton
			105	Lorry HGV 25-60 ton
	Extra heavy lorry	c	106	Lorry HGV 74 ton
	Feeder train	G	202	Feeder/shunt train
	Wagonload train	H	207	Short wagonload train
			208	Medium wagonload train
	Long wagonload train	h	212	Long wagonload train
	System train	I	204	System train STAX 22,5
		T	205	System train STAX 25
		U	206	System train STAX 30
		i	211	Long system train
	Direct Sea	M	305	Other vessel 1 000 dwt
			306	Other vessel 2 500 dwt
			307	Other vessel 3 500 dwt
			308	Other vessel 5 000 dwt
			309	Other vessel 10 000 dwt
			310	Other vessel 20 000 dwt
			311	Other vessel 40 000 dwt
			312	Other vessel 80 000 dwt
			313	Other vessel 100 000 dwt
			314	Other vessel 250 000 dwt
			315	Ro/ro vessel 3 600 dwt
			316	Ro/ro vessel 6 300 dwt
			317	Ro/ro vessel 10 000 dwt
	Feeder vessel	N	315	Ro/ro vessel 3 600 dwt
			316	Ro/ro vessel 6 300 dwt
	Long-Haul vessel	O	317	Ro/ro vessel 10 000 dwt
	IWW	W	322	IWW-vessel
	Road Ferry	P	318	Road ferry 2 500 dwt
			319	Road ferry 5 000 dwt
			320	Road ferry 7 500 dwt
	Rail Ferry	Q	321	Rail ferry 5 000 dwt
	Plane	R	401	Freight airplane

[16] Consolidated heavy lorry is coded as mode S in the chains file. Consolidation in heavy lorries is only available on an intermediate leg in a chain with at least three legs.

Table A-3: Transport chains used for Sweden

1	A		41	GHG		81	VA		121	hM
2	ADA		42	GHM		82	XdX		122	Mh
3	AdA		43	GHMI		83	XdJA		123	CMi
4	ADJA		44	GHMT		84	AJdX		124	GHMi
5	ADJDA		45	GHMU		85	cB		125	iMC
6	ADKL		46	GHQH		86	cS		126	iMHG
7	AJ		47	HC		87	cC		127	Mi
8	AJA		48	hC		88	cH		128	cGH
9	AJDA		49	HG		89	Bc		129	cGHc
10	AKL		50	hG		90	XA		130	cGHM
11	APA		51	HGC		91	AX		131	cHG
12	AV		52	I		92	WB		132	cHGc
13	AVA		53	i		93	HQH		133	cM
14	B		54	IM		94	CHM		134	cMc
15	BR		55	iM		95	HM		135	cMI
16	BRB		56	IMC		96	MH		136	cMT
17	BS		57	IMHG		97	AdJA		137	cMU
18	BSB		58	J		98	AdJdA		138	cPc
19	C		59	JA		99	AdKL		139	cUM
20	c		60	KL		100	AJdA		140	GHc
21	CGH		61	LK		101	LKdA		141	Hc
22	CGHC		62	LKA		102	CGh		142	hc
23	CGHM		63	LKDA		103	CGhC		143	HGc
24	CH		64	M		104	CGhM		144	IMc
25	Ch		65	MC		105	ChG		145	Mc
26	ch		66	MHG		106	ChGC		146	MHGc
27	CHG		67	MHGC		107	GhC		147	TMc
28	CHGC		68	MI		108	GhG		148	UMc
29	CM		69	MT		109	GhM		149	cHM
30	CMC		70	MU		110	GhMI		150	cGh
31	CMI		71	RB		111	GhMT		151	cGhc
32	CMT		72	SB		112	GhMU		152	cGhM
33	CMU		73	T		113	GhQh		153	chG
34	CPC		74	TM		114	hGC		154	chGc
35	CUM		75	TMC		115	IMhG		155	Ghc
36	CWC		76	TMGH		116	MhG		156	hGc
37	cWc		77	U		117	MhGC		157	MhGc
38	GH		78	UM		118	UMGh		158	chM
39	Gh		79	UMC		119	hQh		159	cMi
40	GHC		80	UMGH		120	ChM		160	iMc

161	cGHC		201	XVA
162	cHGC		202	XdJdA
163	cMC		203	XJdA
164	cWC		204	ADX
165	cGhC		205	AdX
166	chGC		206	ADJX
167	CGHc		207	ADJDX
168	CHGc		208	AJX
169	CMc		209	AJDX
170	CWc		210	APX
171	CGhc		211	AVX
172	ChGc		212	AdJX
173	X		213	AdJdX
174	XDX			
175	XDJX			
176	XDJDX			
177	XDKL			
178	XJ			
179	XJX			
180	XJDX			
181	XKL			
182	XPX			
183	XV			
184	XVX			
185	JX			
186	LKX			
187	LKDX			
188	VX			
189	XdJX			
190	XdJdX			
191	XdKL			
192	XJdX			
193	LKdX			
194	XDA			
195	XdA			
196	XDJA			
197	XDJDA			
198	XJA			
199	XJDA			
200	XPA			