Uppsala Master's Thesis in Computing Science 99
Examensarbete DV3
2001-02-19
ISSN 1100-1836

Migration of Service and Customer Data in Intelligent Network Applications

Anders Rosén
Anders.E.Rosen@telia.com

Information Technology
Computing Science Department
Uppsala University
Box 311
S-751 05 Uppsala
Sweden

Supervisor: Lars Planelid
Examiner: Monika Danielsson

Passed:..

Abstract

Intelligent Networks (IN) is the combination of the traditional telecommunications network, new transmission technologies and advanced computing for the provisioning of flexible and innovative new services. Examples of services include Virtual Private Network (VPN), Freephone (known as 800-numbers or 020 numbers) and Premium Rate (known as 900-numbers or 071 numbers), Pre-Paid and Credit Card services.

The area of migration includes the transfer of subscription data and service logic when upgrading to a newer version of a service. The growth in this area and the introduction of new or improved services and new platforms has led to problems in upgrading the services and handling the stored data. Large amounts of data need to be migrated during an upgrade. The operator requires the migration to be carried out as fast as possible, with no loss of data and a minimum of downtime for the system. For the end user, the operator’s customer, the migration must be as transparent as possible.

The Aim of this Master’s Thesis is to investigate the prerequisites of such a migration, to develop a framework for an application and propose an architecture that can handle the requirements to solve the migration problem.

The work has been carried out at Ericsson Telecom in Karlstad, Sweden.

Contents

1.0 Intelligent Networks

1.1 Introduction

Intelligent Network Services is the area in which traditional telecommunications, mobile communications and data communications meet. This is achieved by introducing new services in the telecommunications network. To make this possible, the traditional concept of switching networks must be extended: “Intelligence” must be introduced into the telecommunications network in order to recognize and route calls to the nodes executing the services (containing the actual program logic), and databases are needed to hold the large amount of service and customer data. Service Creation and Management functions must be provided to administrate the Intelligent Network, as well as clear interfaces between the switching network and the Intelligent Network.

The following sections explain the Intelligent Network and the services executing in it. The introduction is by no means meant to be complete. Rather, it outlines the functionality within the Intelligent Network needed to follow the discussion in the following chapters.

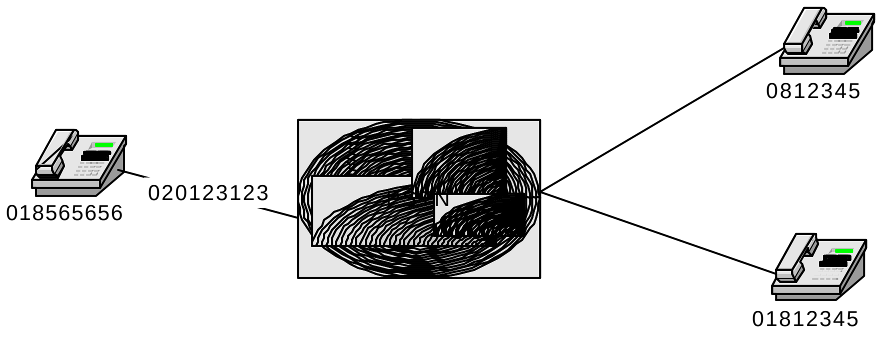


FIGURE 1. A subscriber calls a free phone number.

Imagine a company that wishes to be reached from an universal number. Depending on the caller’s geographical position and the time of day and day in week, the caller can be connected with different numbers, so that an office which is open is always reached. In figure 1 the subscriber calls the universal number, and is routed to one of the destination numbers.

1.2 Why IN?

David Havelin has written a short introduction to Intelligent Networks, [3]. From this paper most of the contents of this section (1.2) is taken.

As a result of the deregulation of economies, telecommunication operators are competing for the first time. Innovative services like call-back and calling cards allow people to use cheaper international carriers in other countries. Cable television companies are now offering telephone services over their cable networks. Clever software allows full-duplex calls to be made over Internet.

The telecommunications companies can no longer justify the high tariffs the levy on basic telephone services. New technology has lowered the actual cost of delivering a telephone call to almost nothing. Thanks to optical fibre, there is now a huge overcapacity in the telephone network. Bandwidth is no longer the scarce resource it was only a few years ago. It costs no more in terms of network resources to call from Stockholm to New York than it does to call from New York to New Jersey. Obviously, the customer is charged a lot more for a long distance call than for a local call. Since most of the telecommunications companies' operating profits come from the long-distance calls they are facing a big problem. They need new sources of revenue.

The answer, at least partly, is to offer subscribers advanced services they are willing to pay a premium for. In the past, adding new functionality to a network involved rewriting core exchange software. Typically, each exchange in the network had to be updated with the new software. A very time-consuming affair.

Intelligent Network (IN) is the solution that allows:

- A telecom operator to design its own, unique services or to adapt existing services to specific customer requirements.
- Impact of installing new services to be limited to a few control nodes.
- Centralized administration of services, thus improving response times and decreasing the human resource overhead required to run the network.
- Customer control of some customer-specific data.

1.3 IN architecture

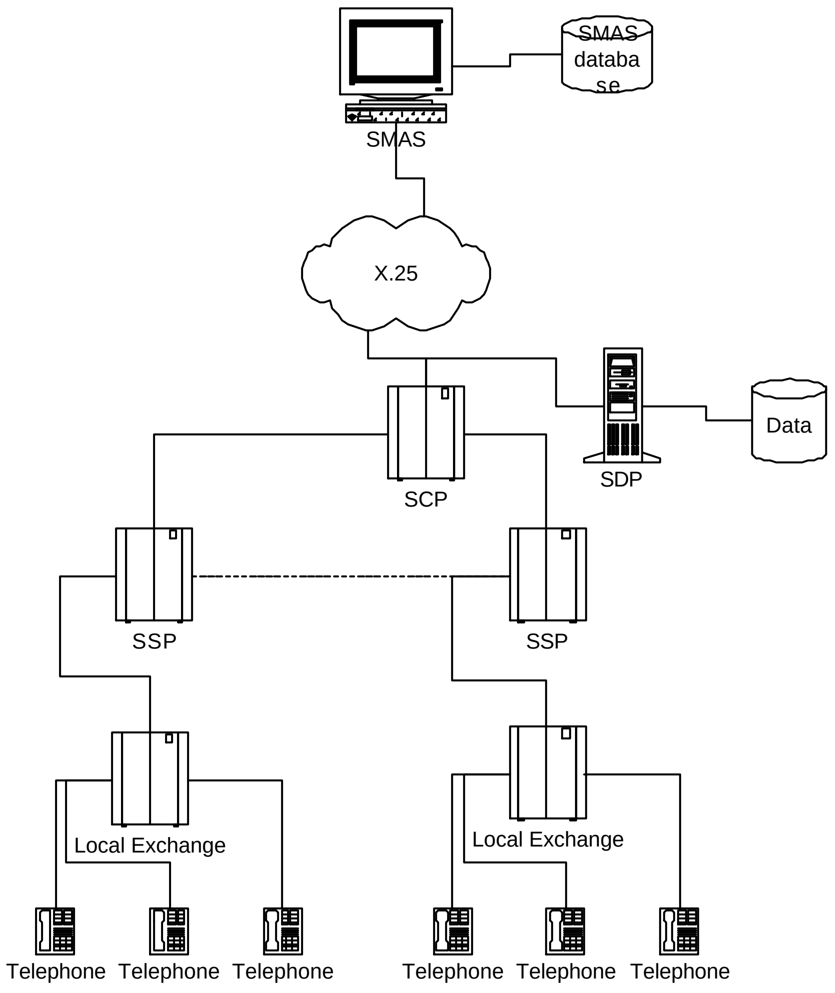


FIGURE 2. A simple Intelligent Network.

When someone makes a call to an IN service, the call must be routed to a special node in the network called the Service Switching Point, SSP. The SSP recognizes the call as needing IN, so all processing of the call stops while the SSP invokes another node, the Service Control Point, SCP. The SCP is the “Intelligence” in the “Intelligent Network”. The SCP controls everything that happens in an IN call, like analysing a dialled number and based on, for example, the time of day finds a corresponding destination number that the call can be routed to. When it has decided what to be done it instructs the SSP to carry out the necessary action. Different protocols are used for different paths in the network.

Non speech paths between Network Elements (such as the Local Exchanges and the SSP, which both are AXE switches) use the C7, or SS7, signalling protocol, while communication between the Network Elements and Management systems (from an AXE to a general purpose machine running Unix or Windows NT) uses X-25 or TCP/IP. An example of the latter is the communication between the SCP and the SMAS System.

What a simple network looks like and what nodes are involved is shown in figure 2. What is considered to be part of the Intelligent Network is a matter of definition, but usually it is the combination of one or several SSPs, SCP and the Management Functions and possibly external databases, such as the Service Data Point, SDP. The SDP is used when very large amounts of customer data are needed.

The IN network has been divided into a number of distinct parts, each with its own particular function and defined interface towards its neighbours.

1.3.1 SCF, Service Control Function

The SCF contains the logic of the services (the programs) and all or part of the customer data. It has complete responsibility for making decisions related to a call. It may run on the AXE platform, which is most common, but since it does not do much switching, it may run on some other system, for example a Unix machine equipped with C7 Interfaces. The node that contains the SCF is called the Service Control Point (SCP).

The SCP contains communication devices that allows it to communicate with the Management Systems. The key interface is the one towards the Service Management System, SMAS. SMAS is used to create the services and contain a full replication of service and customer data the SCP. The physical link between SMAS and the SCP uses the X.25 protocol.

When a service is ready for deployment (installation in the telecom network), this link is used to download the service logic into the SCP. Also, when a user of the IN Service performs changes in the Service (his customer data), the changes will propagate from the SCP up to the SMAS database.

1.3.2 SDF, Service Data Function

The SCF can store the data it needs for the services, for example lists of subscribers. Alternatively, this function can be allocated to the SDF. The SDF will provide the data on demand to the SCF. Typically, the SDF would be a UNIX machine running a commercially available database such as Sybase. The physical node that contains the SDF is called Service Data Point (SDP). The protocol between the SCF and SDF is INAP, IN Application Protocol, over C7 signalling. The INAP primitives that are used are UPDATE and RETRIEVE.

1.3.3 SSF, Service Switching Function

The SSF interprets the instruction sent by the SCF and passes the commands to be executed to the Call Control Function (which is not part of IN but provides IN with information about a call and executes orders that have been sent from the SCF via the SSF). The exchange that contains the SSF is known as the Service Switching Point (SSP).

1.3.4 SMA, Service Management and Administration

The SMA is used for administration of IN services. This includes adding or removing subscriber data and services data. This is typically a client program that is used by the service provider, for example a company like Telia. The SMA is typically run on a UNIX platform or PC, and is tailored for a specific service and/or operator. One example is GSA, Generic Service Adapter, which works towards the SMAS database.

1.3.5 SMAS, Service Management Application System

SMAS provides the operator with tools to define, install, administrate and maintain services and subscriptions in the network. SMAS also includes functions for communication with all SCPs in the network, as well as interfaces towards SDPs. SMAS runs on a UNIX machine and a central part is the database that contains information of the services.

2.0 Migration

2.1 Definition

Migration has been discussed on several workshops within Ericsson. Migrations vary from small functional upgrades to complete replacement of both hardware and software. This study focuses on the case when a service is upgraded from one version to another. Typically, but not necessarily, this also involves changing or upgrading the platform.

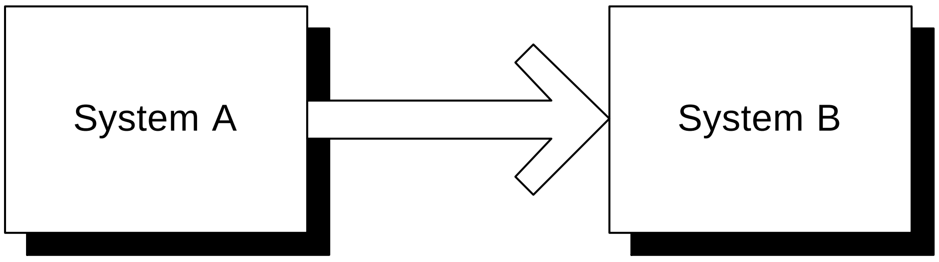


FIGURE 3. Migration

The goal of the workshops has been to outline and try to establish procedures and rules to follow when upgrading an installed service. Not only must the platform itself support migration, services need to be designed with upgrades in mind from the beginning.

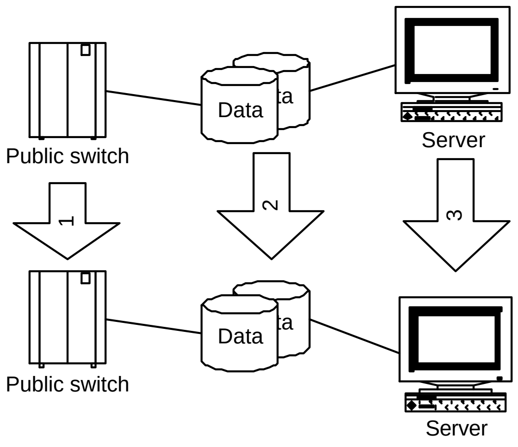


FIGURE 4. Migration paths

The figure shows migration paths and upgrades that are relevant in the case of an IN migration. The switch (AXE) software is upgraded (1), subscriber data is migrated (2), and management and service creation platforms (SMAS) are upgraded (3).

2.2 Identifying Different Migration Scenarios

As discussed in previous section, a number of scenarios are possible:

1. Service upgrade, only.
2. Platform upgrade, only.
3. A combination of service and platform upgrade.

Any of above can be done either on a live system or on a stand-by system.

2.2.1 Service upgrade

As mentioned earlier in this document, a number of criteria must be fulfilled in order to transfer customer data when upgrading a service. This is described below.

2.2.2 Platform upgrade

If compatibility was guaranteed between different platforms, upgrading a service would be no problem. Consider the PC world: backward compatibility is taken for granted – a program designed to run on a 286 will probably run on a Pentium as well.

For most upgrades there is no problems to transfer service data, at least within the same IN family (e.g. SMAS 2.1.1 to 2.1.2), and recommended procedures exist. Only in special cases manual intervention is needed.

It is assumed that if a platform upgrade is part of the total upgrade, this should be done before the service data migration. Conversion of service data is beyond the scope of this document.

2.3 Known Problems

2.3.1 Telecom Characteristics

A complicating factor is the nature of a telecommunications network. It is expected to be fully functional 24 hours a day, 365 day of the year. How is a migration carried out with no or minimal disturbance of the traffic? It is not acceptable to bring the network down for this type of maintenance.

2.3.2 Large Amounts of Data

A realistic scenario is that 100.000 of subscribers are affected by an upgrade. At some stage data needs to be drained from the source system, possibly changed, and then downloaded to the target system.

2.3.3 Limited Capacity

The X.25 interface between SMAS and the SCP has a limited capacity, with a download rate between 5-6 seconds up to several minutes, depending on the traffic and number of parameters, for each subscriber. A real life service contains data that the subscriber can change, for example call forwarding numbers or data set by the network, for example message notifications. If consistency is to be guaranteed, all operations that can change the data must be disabled during an upgrade.

For 100.000 subscribers, this means a six day data freeze in the most optimistic case for a brute force migration.

2.4 Limited Possibilities with Existing Tools

To transfer customer data the only officially supported tool is the *Service Deployment* functionality. This mechanism supports data transfer from one, old, service to another, new, service. However, this mechanism has limitations [2]:

- Only data that is identical in both services will be transferred.
- New data modules will be assigned template data.
- Changes in the data definitions can not be handled.
- Only minor logical changes in the service can be handled.

3.0 An Example IN Service

3.1 Design

The design of a service resembles very much the writing of a program. This “program”, however, is not written in an ordinary programming language like C or C++. Instead it is assembled from a palette of available Control Types, which then are linked together to form the logic of the service, a Service Script.

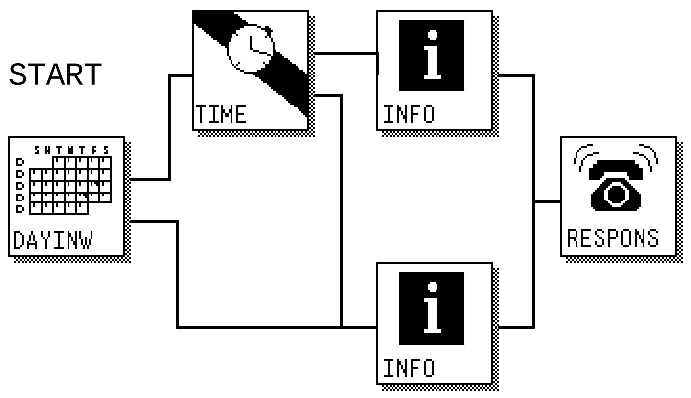


FIGURE 5. Example service.

The Control Types, when invoked, executes one or several predefined operations. This is dependent on the kind of Control Type, and can be anything from a simple branch based on the value of a variable (choosing an outlet for the next step of execution) to complex Control Types that for example maintains a dialogue with other Network Elements (setting up calls, monitoring for events etc.). All Control Types have one inlet and zero, one or more outlets.

The physical node that is responsible for storing and executing Service Scripts is the SCP. The function in the SCP that executes the logic is called the SCF. In the SCF it is the Service Script Interpreter, SSI, that performs the encoding of the Control Types and the interpretation of its data.

The control types are combined to create Service Script Logics, SSL, which are capable of carrying out more complex functions than the Control Types themselves. In an ordinary programming language, the modules would represent the primitives present in order to write a program, and the SSL would represent a procedure or function.

The next step is to collect all SSLs and define a Service Logic, SL, which would represent the main program. As a final step, the data for a Service is defined.

3.1.1 Service Data

The graphical, high-level design of a service logic does not in itself completely define a service. The logic must be complemented with service data.

All data for an IN Service can be said to be contained in what is referred to as *Call Instance Data*. Call Instance Data is a limited block that gets allocated to a call during its initialization, and consists of three or four (depending on platform) data types: Numbers (*Kind of Number*), Integers (*Kind of Variable*) and Long Integers (*Kind of Long Integer*), and String (*Kind of String*) in newer platforms. These are in turn organized in a predefined enumeration, meaning that you have from KoN 0 up to KoN *x*, KoV 0 up to KoV *y* etc. A Ko*X* with a certain index can be seen as place holder for a value, that might or might not be filled in at any time in the execution of a call.

The Call Instance Data block is unique for each call and is allocated when a new call is set up. The *persistent* storage of data is connected to the Control Types. Some Control Types have a *Data Module*. A Data Module can be any of *Local, Global* or *Customer*. A local Data Module is only visible within the service script in which it is defined. Independent of which service customer is executing the script, the values stored in a Local Data Module are always the same. A Global Data Module might be referenced by (connected to) many Control Types, independent of which script the Control Type is defined in, and hence available to all service scripts. A Customer Data Module is like the Local Data Module, but the values are subscriber specific.

3.1.2 Access to Data during Execution

During execution of a service, a mechanism is needed to at any point retrieve the correct data.

When a service is installed, a *Service Administrator, SA*, is created for each SSL in the service. An SA “looks after” a particular service script and contain pointers to the data to use.

There are always two different pointers available, one for pointing out Local and Global data, and one for pointing out Customer data. One is the Service Administrator, SA, pointer, and the pointer for Customer data is called a *Service Customer, SC, pointer*. To identify data within a SA or SC indices are used. The SA and SC pointers are usually looked up in a global access table at the start of execution. The control type that holds such a table is called NRANAX (as in Number Analysis Extended).

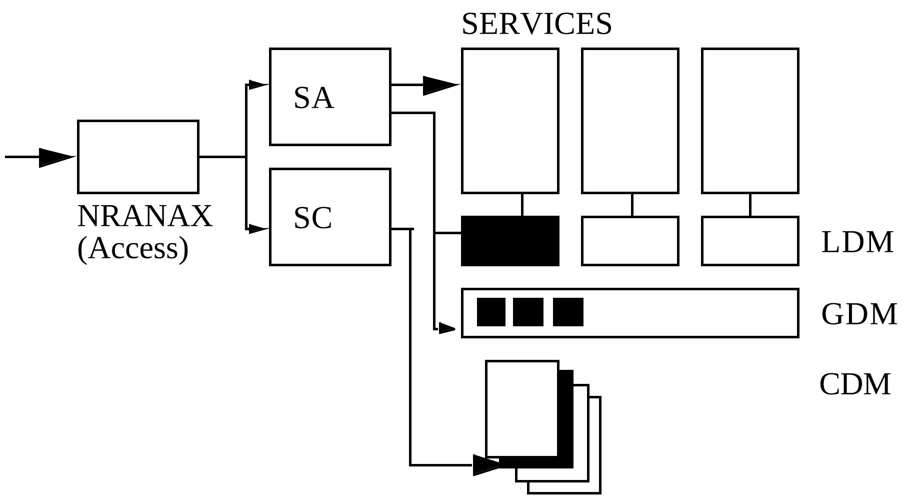


FIGURE 6. Access procedure for service and customer data.

A Local Data Module, LDM, is associated with one and only one Service Administrator, SA.

A Global Data Module, GDM, may be accessed by more than one SA.

A Customer Data Module (CDM), belongs to a service customer (a subscriber) and may be accessed by any SA that need customer-specific data, by using the SC pointer.

To illustrate this, consider the example Service Script Logic. This is a *very* simple routing service that implements, given an access number (known as the B-number, that is, the dialled digits), the possibility to route this call to different destinations depending on the time of day and day in week.

Before this script is executed, some access script has been executed that performs number analysis, and as a result from that we have a SA pointer that points out which script to jump to and what local data to use, and the SC pointer to use from which to get the customer data.

Assume that all subscribers that use this service have ordinary working weeks, that is, Monday to Friday. (Other services may use the same logic to implement versions with other days.) Therefore the first module, the DAYINW, is a local data module. For this service the subscribers have the same opening hours, for example 8-17 during working days, and the same data is also used in other services. Therefore the TIME module is global, and is connected to the predefined

Global Data Module. The INFO modules are used to store the number where to route the call (which is sent to the SSP when reaching the RESPONS module). All customers have their own destination numbers where they want calls to be routed. Therefore the data modules associated with the INFO control types are defined as Service Customer Data. This means that all subscribers will have their own set of INFO data modules.

In a real life service, all data above would be customer specific, and hence customer data modules. This would allow the subscriber to decide the opening hours and days.

When it is decided what modules shall be local, global and customer data is entered. It is possible that the global data module already exist, and in that case it is connected to the control type. Otherwise it is created, and values are entered. The local data are created and values are assigned. In this example a particular outlet should be choosen from Monday to Friday, and another outlet chosen Saturday and Sunday. The customer data modules are special. When they are created, it is not possible to assign values. Either they are assigned template data, or are left blank. The management application used for service provisioning (managing users and subscriptions) would typically contain fields to be filled in when creating a new subscriber. In this case the two INFO modules should contain the numbers where to route a call (known as the C-number).

3.2 Deployment

Now the service is ready for installation into the network. The SMAS application used for this is called Service Deployment. When a service is to be installed into an SCP, the service is selected from the service library and the SCP is specified. The service can be installed immediately or scheduled for future installation.

With some additional work in the public network, like setting routing and triggering information, the service is now ready to use. When customer is interested in the service the operator adds him as a subscriber. Upon installation of the subscriber, he is assigned his own set of customer specific data, which before this is empty or contain template data. These has to be updated with information appropriate for the new subscriber. There is also the possibility that the customer is allowed to control the telephone numbers himself, by allowing Customer Control. In that case he would be given a certain Customer Control access number, and when calling this number it is possible by voice-prompting and a touch-tone telephone to change the data. For customer control a specific script is created that enables access to the customer data modules in the service.

3.3 Functional Upgrade

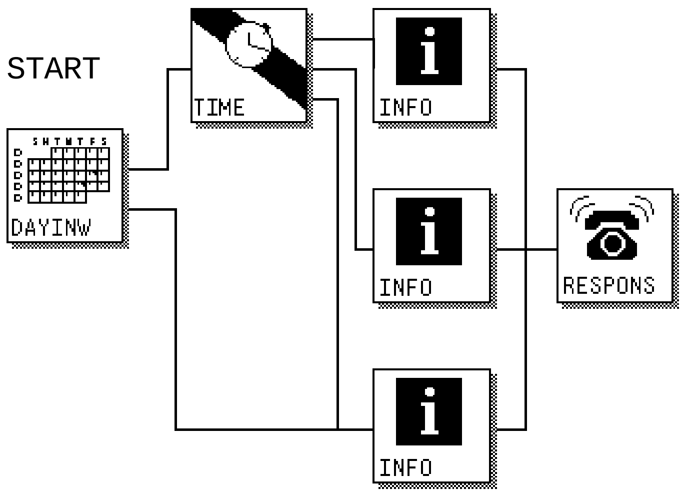


FIGURE 7. Extended version of example service.

The figure above shows an extended version of the example service. This change could be because of new customer requirements. An extra outlet is added to the TIME data module, and a INFO customer data module has been added. The objective could be to route the incoming telephone calls to a different number during lunch hours.

This is a simple case of customer data migration. The data in the earlier version’s customer data modules will be mapped to the corresponding ones in the new version, and the added customer data module will have to be manually updated to contain proper values.

But what happens if SMAS cannot resolve the mapping? There are a number of possibilities. For example, this can happen if a customer data module is removed, and its data is to be distributed into one or several new customer data modules, or if the new service is completely different from the old.

How can this be solved, preventing loss of data, minimizing the amount of manual intervention needed, and ensure fast introduction into the network?

The investigation shows that it might be hard to implement a tool that fulfills all requirements and is generally applicable. Also, the nature of each service has to be considered. As in one known case, the IVPN 2.0 to 2.5 upgrade, very ad hoc actions was taken to migrate the data.

Instead of offering a complete environment that can handle all possible migration scenarios, a *set of tools* can be developed. These tools can then be used in combination with documentation to migrate data.

4.0 Survey of Existing Methods for Migration

4.1 Introduction

A number of service applications are deployed to customers, especially fixed operators. These applications might be changed due to error handling, functional upgrades or platform upgrades (SCP, SMAS). When the changes are implemented, proper procedures must be available for preventing loss of data, ensuring fast introduction etc.

4.2 AXE

For AXE a number of methods exist for functional upgrades. This platform has been designed for maximum up-time and availability, and normally upgrades can be done on the fly. For non-AXE platform, this is not always the case. There is support in, for example, SMAS to aid an upgrade. However, this support cannot handle all the cases that can arise. It expects a number of criterias to be fulfilled in order to work. And if these criterias are not met, a lot of manual intervention might be needed in order to make the new version work. Examples are when a large amount of customer data needs to be migrated, and when the upgraded service differs from its predecessor significantly.

4.2.1 Traditional AXE methods

It is common that the service application will be upgraded simultaneously with the platform, especially when upgrading from one IN generation to another.

In the SCP it is possible to use conventional data conversion functions to perform an upgrade. An existing service application can be converted, including all data, using the AXE *FURAX* data conversion methods. But these methods are designed for AXE switches. An IN service is stored and executed in an AXE, but the SMAS system acts as master, with a database containing a mirror of the data in the SCP. To manually change a service or data associated with a subscriber, SMAS is used, and then the change is downloaded into the SCP. Changes made directly to the SCP, for example using customer control, are propagated to the SMAS database. For this reason there is nothing to gain from issuing an upgrade from the SCP. The upgrade should be initialized from the SMAS system. That is, FURAX can not in a simple way be used when upgrading services and subscriptions.

4.2.2 Service (File) Transfer

The Service Transfer function allows *service data* to be transferred to and from an SMAS system. The function provides a graphic user interface for listing and selecting different types of SMAS objects and for performing the transfer operation for the objects selected. There are four types of SMAS objects that can be transferred using the Service Transfer function:

- Services

- Service Logics
- Service Script Logics
- Global Data Modules

Service Transfer simply extracts the rows from affected database tables to Sybase native format called bcp (bulk copy). The output is stored in a tar archive, containing a file for each table included in the transfer. When loading the file into SMAS, the program is responsible for inserting data in a controlled way, thus ensuring the integrity of the database.

Service Transfer is used to transfer service data and cannot be used to transfer subscriber data.

4.2.3 Service Deployment

The Service Deployment application provides an interface for *installation* of services and handling of different service versions. Service version handling makes it possible to *move subscriptions* from one version of a service to another without losing any subscriber specific data. The following operations can be performed with Service Deployment:

- Look at service/subscriptions/SCP attributes
- Install a service into an SCP
- Activate or deactivate an installed service
- Remove a service from an SCP
- Setup a stand-by service version for one or more subscriptions
- Switch service versions for one or more subscriptions
- Remove the stand-by service version for one or more subscriptions.

This method can be used for service application upgrades that only affect the logical parts, or for upgrades that only concern a limited amount of service data (only a few users) or only a few data modules.

The upgrade is controlled by SMAS, which will update the SCP.

Since this is a manual method, and it requires the source and target data structure to be similar, it is not a solution to transfer thousands of subscribers.

5.0 Requirements for Migration Application

5.1 Thesis Assignment

These are the requirements as originally listed in the Master’s Thesis assignment.

5.1.1 Basic Requirements

The source system is typically

- SMAS and SCP for IN 2.1, 2.2 or 2.3.

The new target system is one of:

- SMAS and SCP for IN 2.2, 2.3 or 3.0
- UNIX SCP

5.1.2 Migration Application

The application shall:

- Read data from the SMAS database or directly from the SCP.
- Produce data file (complete or partial).
- Be able to interpret specified rules to convert migration data to new format.
- Detect errors such as mismatch rules, database and system failures.
- Write data back to the SCP or SMAS database.

5.1.3 Rules

The rules shall include a number of primitives, for example

- Searching in databases and tables.
- Definition of help variables.
- Arithmetic functions.
- Logical functions (including bit operations).
- String functions.
- Routines for output to file.

5.2 Requirements from Workshop

Requirements listed in [5]:

- It shall be possible to transfer many (20000-60000) subscribers fast (flexible amount of data per subscriber) from one version of a service application product to the next without disturbance of the traffic handling.
- A maximum of 15 minutes break is mentioned (the definition of “break” is omitted).

- It shall be possible to transfer a subscriber from one version of a service application product to the next without changes in the access procedure for the subscriber. The access number shall be kept unchanged.
- It shall be possible to process subscriber data (modify, add, delete, change format, to use default data etc.) during transferring.

6.0 Generic Migration Algorithm

6.1 Migration Scenario

To illustrate what is affected, an upgrade of the example service will be used as a reference when a generic migration flow of events is outlined.

6.1.1 Design new service

The first step would be to design the new service that will replace the previous version. If possible, the designer shall keep in mind the mapping of service customer data modules.

Naming conventions should be followed in order to have the possibility to use traditional methods for transferring customer data if desired. For example, the service name must be the same, only the version number is incremented.

In this example it is assumed that the previously designed service is installed with the name OFFHOUR, version 1. Hence the SA name of the SSL will be OFFHOUR01AA. There are three subscriptions (service customers) installed. They are represented as SC Name 000000, 000001 and 000002.

In the management view in SMAS the tables `subscriber` and `sub_subscriber` contains this information:

Subscriber Name	Subscriber Id
Company A	1
Company B	2
Company C	3

TABLE 1. Subset of subscriber table.

Subscriber Id	Telephone number	Admin. info
1	020111111	NULL
2	020222222	Test subscription
3	020333333	NULL

TABLE 2. Contents of sub_subscriber table

The information about the installed subscriptions is stored in a table called `subscription`. It contains information about what service a telephone number is associated with, in what SCP it is installed, and whether stand-by versions exist etc.

It also contains what number analysis module to use for a specific subscription. In this example it is assumed that it is called GLOBAL_NRANAX. The most important fields are:

Telephone number	SA Name	SC Name
020111111	OFFHOUR01AA	000000
020222222	OFFHOUR01AA	000001
020333333	OFFHOUR01AA	000002

TABLE 3. Contents of GLOBAL_NRANAX

6.1.2 Install service into the network

When the service is tested and ready to use, it should be installed into the network. For this, standard SMAS tools are used.

It must be possible to install new subscriptions to the service. Therefore administrative functions must be present. This will include designing new management forms or using SMAS. The new and old service must be possible to administrate in parallel.

6.1.3 Prepare migration process

A migration strategy must be worked out. For the simple example, it is sufficient to migrate the subscribers one by one.

A recommended procedure for real services is to start migrating a test company or subscriber, and extend this to an understanding pilot company or subscriber.

The tools needed in the migration phase must be available and tested.

6.1.4 Start migration loop

If the steps above are finished, the main “loop” of the migration process can start.

1. Add entry to global access DM (NRANAX)

For each subscription to the OFFHOUR service to be migrated, a temporary entry is added in the access table. This can be done with the SMAS tool *Global*

Data Administration. For most real services, this is too time-consuming to be done manually.

The table now contains a new row:

Telephone number	SA Name	SC Name
020111111	OFFHOUR01AA	000000
020222222	OFFHOUR01AA	000001
020333333	OFFHOUR01AA	000002
F020111111	**OFFHOUR02AA**	**000000**

TABLE 4. Updated GLOBAL_NRANAX table.

The SA Name now points to the first SSL in the service OFFHOUR version 2. The telephone number is prefixed with an “F”, to make this a non-accessible number, but still provide the option to install subscriptions to the service. It uses the same Service Customer, SC, pointer.

2. Clone subscription

The above entry in the NRANAX global data module is sufficient for the SCP to recognize the new subscription. But to ensure consistency between the SCP and SMAS, the `subscription` table must be updated.

scp_name	tel_no	srv_name	srv_ver	sc_name	nranax
SMASAXE	020111111	OFF-HOUR	1	000000	GLOBAL_NRANAX
SMASAXE	**F020111111**	**OFF-HOUR**	**2**	**000000**	**GLOBAL_NRANAX**

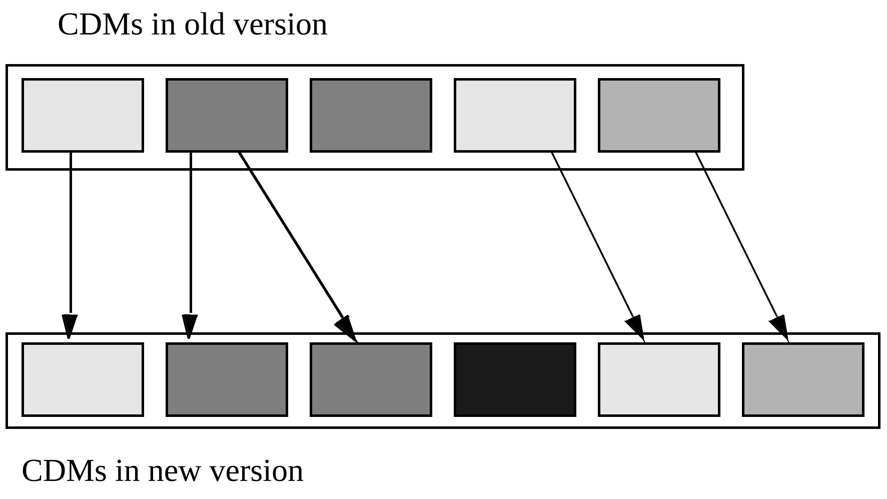
TABLE 5. Updated subscription table (subset).

3. Assign CDM values

The designer and/or the people responsible for the migration of the service must have a clear picture of mapping between the customer data modules in the new and the old version. Identified cases are (this is general, not specific for the example service):

- CDM that are identical in new and old version are not necessary to change, and can be left as is.
- CDM that is present only in old version, but contains data that needs to be distributed among new CDM.
- CDM present only in old version. No action needed.
- CDM only present in new version, but contains inherited data.
- CDM only present in new version.
- CDM mapped between versions, but data changed.

In the last case above, the data need to be defined during the actual switch, that is, after the old subscription is blocked and before the new is taken into traffic. This means the time when it is not possible to use the service for that subscriber increases.



□ 1. CDM same in both versions

▩ 2. CDM only present in old version, but contains data to be distributed to new CDMs

▩ 3. CDM only present in old version

▩ 4. CDM only present in new version, but contains inherited data.

■ 5. CDM only present in new version.

▨ 6. CDM mapped between versions, but data changed.

FIGURE 8. Mapping of CDMs between versions.

The database and the SCP contains one set of CDMs per Service Customer (instances of the template data) and each service has its own set of template data. The transition during a migration for one set of data is shown figure below.

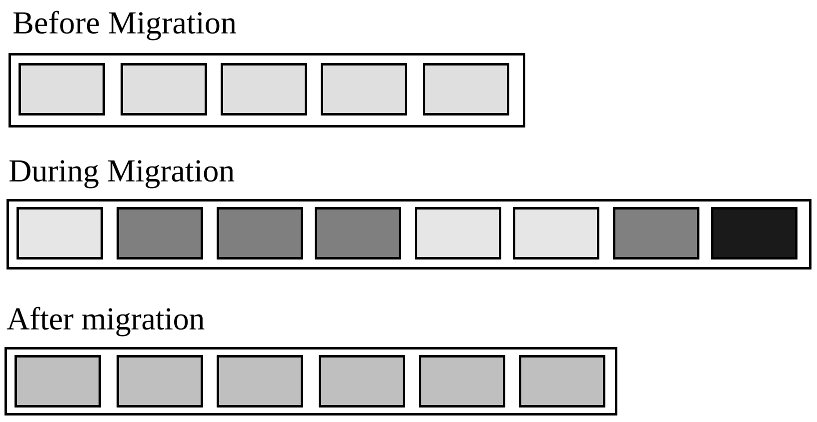


FIGURE 9. CDM situation in different phases of the Migration.

To create new CDMs, the C++ SMAS API can be used. This will only create CDMs in the SMAS database, that is, the CDMs that are used for traffic in the SCP has not yet been created.

New CDMs will be created with the templates as a base. This must then be combined with the possibility to create data according to the model above for each specific service customer.

4. Switch subscription
When all CDMs are assigned proper values, the NRANAX data module is updated, subscriber and subscription tables are updated, and the switch to the new version is done.

6.2 Limitations

The process outlined in this section has a few restrictions. Still, the most important data, the data in the SCP, is not updated. And the API is not supported and perhaps not even compatible between platform versions. The SMAS object *Request Queue* can be used from the API to install the created subscriber data modules in the SCP, thus solving the first issue. But the download time issue remains unsolved, and is not feasible for a large number of subscriptions.

If it is not possible to migrate service customers according to the above scheme, an alternative process is needed. A proposed architecture is described in section 8.0.

7.0 Proposed Methods for Migration

7.1 Entirely Manual

Using built in commands and tools in SMAS, minor upgrades and customer data migrations can be carried out. The procedure is described in the General Migration Algorithm above.

7.2 Extension of Service Transfer

One way of solving the transfer of services is to change the output from Service Transfer before inserting data into the new version of SMAS. The files contained in the tar archives could be changed in a way that, for example, prepares it for insertion in a table in the upgraded system. The problem with this approach is that doing so requires knowledge of all affected database tables and dependencies between them. The database model is not officially documented, and must be reverse engineered. There used to be a public API available, but this is not supported anymore.

7.3 Service Interchangeable Format

To get around the problems described above, a program for this purpose has been developed for internal use at Ericsson. It is called *sifio*, and allows editing a service outside the SMAS database. It works like Service Transfer, but extracts the logic of a service into a single text file (Service Interchangeable Format). This file is editable in any text editor, and can then be reloaded into the SMAS database. A service can be extracted with sifio for one SMAS version, and then loaded in a newer version of SMAS using sifio for the new version.

Sifio can be used for changing service version numbers (e.g. before delivery), change names of components, change row data for data modules etc. It can also be used for migrating service data from one version to another.

Sifio cannot, however, be used to extract service customer data.

7.3.1 Java Based SCE

As a direct result of the thesis work, a new version of SMAS was designed. The new Service Creation Environment was designed in Java based upon requirements collected after the thesis work was completed. A prototype was designed that made development of a SMAS service possible using only one tool, instead of three or four in the SMAS environment. Instead of working directly towards the proprietary SMAS database, the sifio file format was used. See section 9.0 for a short introduction to this application.

7.4 Migration Control Node

An Ericsson product called Internet Gateway, ING, contains components that can be used for accessing and updating data in the SCP in a general and efficient way.

The Internet Gateway is used to build external clients that communicates with the SCP and/or SMAS system. The reason for introducing the Internet Gateway was to enable web access for customer control of service customer data.

A solution for migrating customer data based on the ING will be described in the next section.

8.0 Migration Control Node

8.1 Introduction to Internet Gateway, ING

The ING consists of a UNIX server equipped with hardware to facilitate C7/INAP signalling. With the ING it is possible to communicate with the SCP either directly via its C7 interface, or via GSA-RPC through SMAS. When using the C7 signalling interface a special protocol called Double Update is used, and the application used is called SIG, SCP Interconnection Gateway. The SIG application is a stand-alone part of the ING. Double Update invokes the update service logic in the SCP, which is the same as for customer control. A jump is made based on supplied values to a service specific update script. This script can then read or update values in customer data modules.

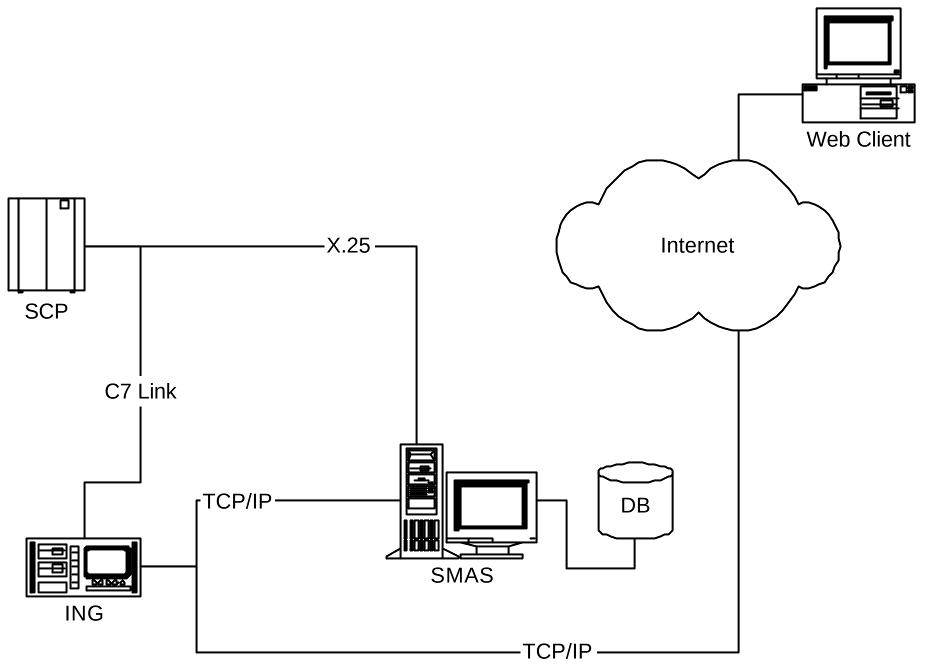


FIGURE 10. ING Typical Physical Environment

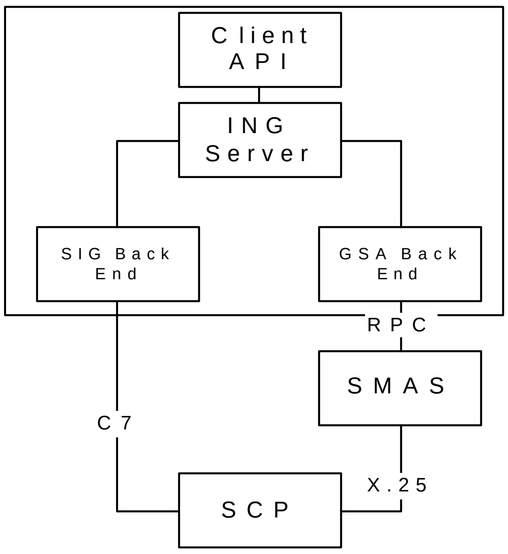


FIGURE 11. ING Architecture Model

The ING server and client parts are written in Java, while the protocol converter (SIG) towards the C7 network is written in C/C++. To develop an application that uses the ING the data model and API must be understood. A service subscriber and corresponding set of customer data is uniquely identified using the service access number. In the ING all parameters belonging to a subscriber is collected into a *profile*. A template profile is defined by means of a *Service Data Definition*, SDD. In the SDD properties for each parameter is defined, like valid values, default values, from which data source to get the parameter, which target to write the parameter etc.

The SDD object is typically created when starting the application, and uses a text file as input. The general syntax of the file is:

```
<parameter_name>.<property>=<value>
```

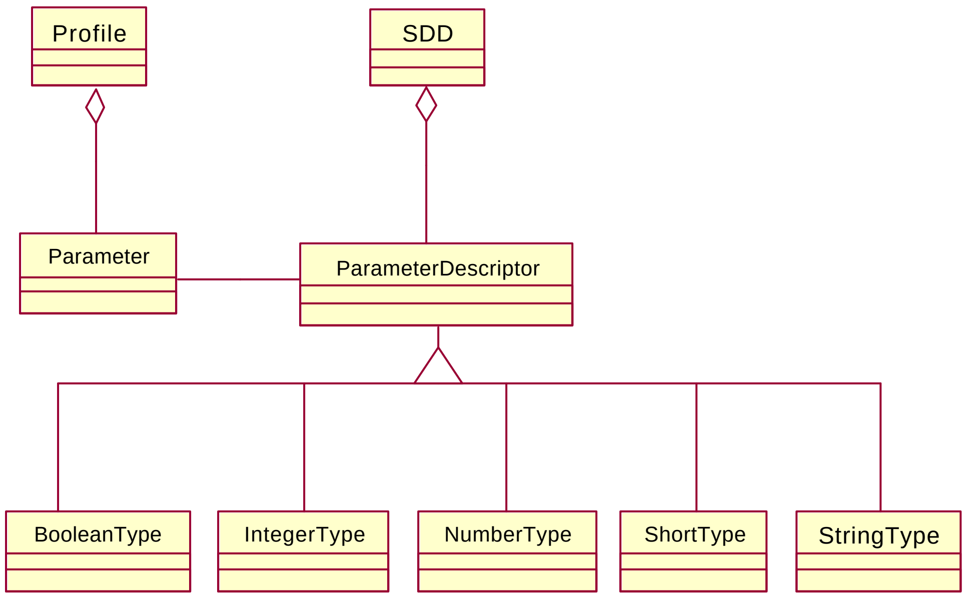


FIGURE 12. Object Model, subset of ING

An example SDD, based on the previously described service is shown below. Before migration, the service contained two subscriber specific data entries, the numbers where to route incoming calls depending on time of day and day in week.

```
NUMBER1.addParameter=NumberType
NUMBER1.get=GSA
NUMBER1.GSAConfigFile=/opt/in_services/ing/gsacfg
NUMBER1.set=SIG
NUMBER1.SIGConfigFile=/opt/in_services/ing/sigcfg
NUMBER2.addParameter=NumberType
NUMBER2.get=GSA
NUMBER2.GSAConfigFile=/opt/in_services/ing/gsacfg
NUMBER2.set=SIG
NUMBER2.SIGConfigFile=/opt/in_services/ing/sigcfg
```

FIGURE 13. SDD (template profile) for the Example Service

8.2 Using the ING for Migration

The migration application can be used either to convert one profile into a new format on the fly, reading from one source system and applying the changed profile towards the target system. Or it can be used to “drain” a source system for all subscribers to be migrated, store the profiles, for example by using object serialization towards the filesystem, and then later read back, converted and applied towards the target system.

For each back end, there is also the option to implement an interface called *ParameterTranslator*.

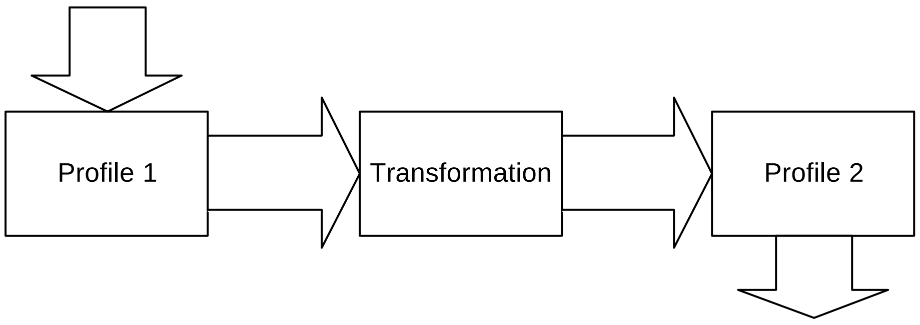


FIGURE 14. Transformation of an ING Profile

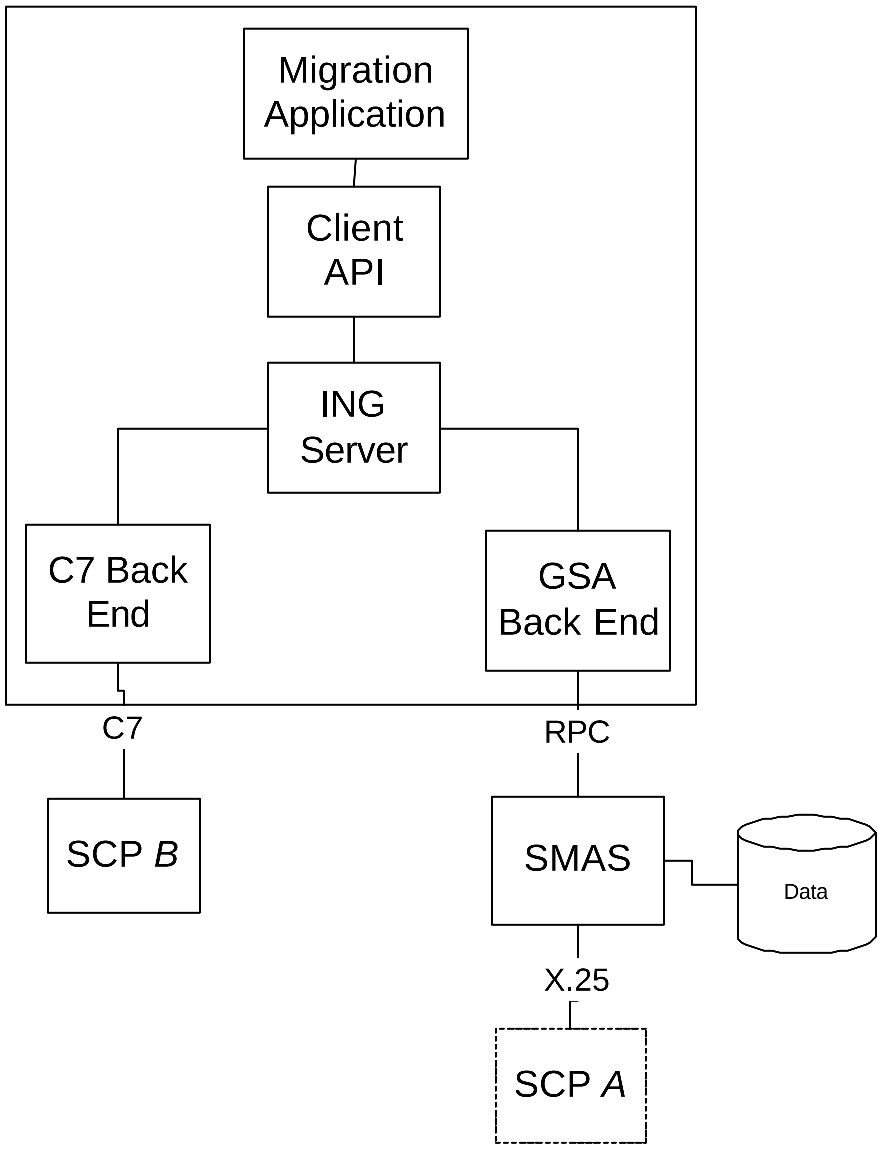


FIGURE 15. Migration Control Node

This solution centres around having an external migration application built on top of the ING client API. Since the service customer data in the SCP is mirrored in the SMAS system the SMAS database will be the *source system*. To get around the low bandwidth link between SMAS and the SCP, the ING C7 interface is used for loading data into the *target system*. The target SCP can be either the same SCP to which the SMAS system is connected (SCP A), or a another SCP, possibly running another version of IN.

The connection between SMAS and SCP is mainly used for provisioning applications, such as Subscriber Management. This document mentions X.25 as the protocol used, but later implementations might use an optional protocol called INM, which uses TCP/IP. With X.25 the data is sent in an ASCII format called MML (Machine-Machine Language), while INM is a binary format. INM offers improved performance over MML with a factor of three to four.

The management traffic is also limited compared to traffic using C7 signalling. This is to ensure enough capacity for call related processing in the AXE.

Still the synchronization problem exist. While the migration capacity is high, it is on expense of the SMAS database update in the target system. When a C7 Update operation finishes, and if service customer data is changed, a synchronization operation will be sent from the SCP to the SMAS database. This is optional, and is controlled by a soft “switch” in the SCP. Default in ordinary traffic is SMAS “update on”. However, capacity is increased [12] when SMAS update is switched off. Two alternative solutions for this problem exists: either the performance penalty having SMAS update switched on is accepted, or the SMAS update is switched off, and the SMAS database is updated in another way. The first alternative can be motivated with the argument that the most important data, the traffic data, is updated first. Updating the SMAS database can then take its time.The second alternative might be used when a new system is to be taken into operation at a certain time, and customer control or any other activities that affects the consistency of data in the SCP and SMAS are disabled during migration.

8.3 Applied Algorithm for Migration

1. Design, verify and install new version of service.
2. Install necessary administration tools, e.g. GSA, to administrate new service.
3. For new subscribers, add them to new service.

For data to be migrated:

1. Install stand-by subscription, “dummy”, for the service customer to be migrated. Use the same access number as in old version.
2. Create a profile object for the subscriber
3. Update data modules that differ from template. This can be, for example, data specific for the subscription not present in old version. Data can be retrieved from database or equivalent.

4. Copy data from SMAS that is subscriber specific but not possible for the customer to change via customer control into profile object.
5. If present, disable Customer Control for the subscriber.
6. Copy data from SMAS that are under Customer Control.
7. Block subscription.
8. If present, data from SMAS associated with "in traffic" use (statistical counters etc.) are copied to their corresponding new CDMs in the profile object.
9. Write profile object to SCP using the C7 interface.
10. Update access procedure for subscriber to make it possible for subscriber to reach subscription in the same way as before.
11. Take new subscription into traffic.

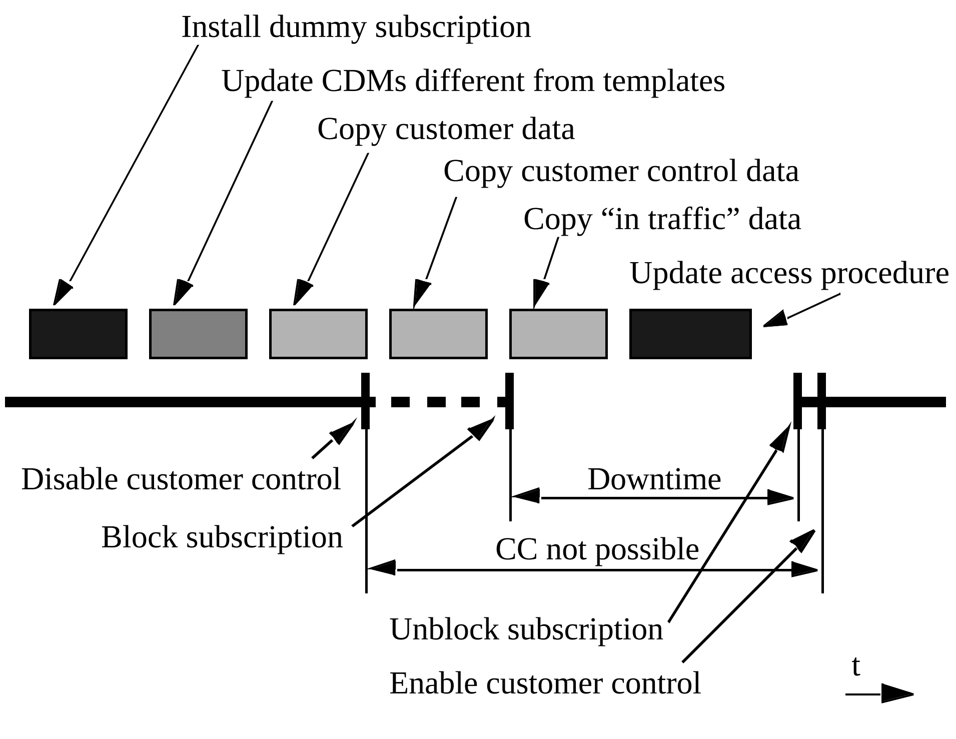


FIGURE 16. Service Customer Data Migration

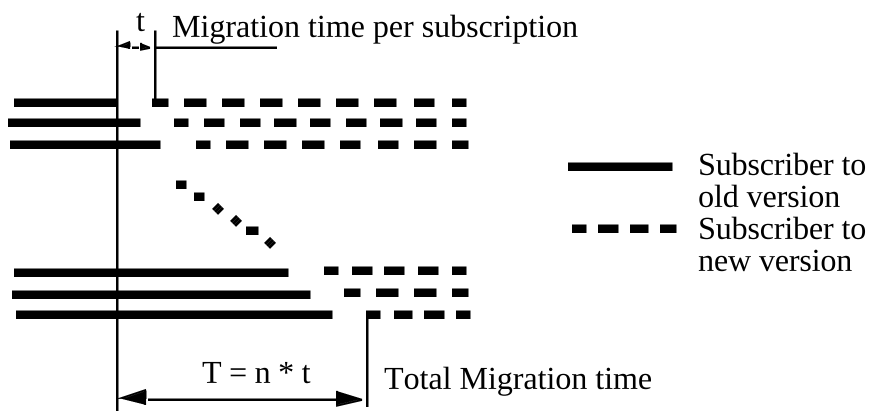

t
Migration time per subscription
Subscriber to old version
Subscriber to new version
T = n * t
Total Migration time

FIGURE 17. Migration of a set of subscriptions.

Still, the time it takes to migrate a large amount of subscribers is significant. One factor that affect the migration is customer control. If it is not possible for the subscriber to change the data during the migration the block and unblock operations in the above algorithm are unnecessary.

9.0 Sade, Java Based SCE

9.1 Introduction

This section presents the *Service Application Design and Editing, Sade,* tool. The Sade application was developed as a direct result of the thesis work, after identifying the weaknesses in the SMAS development environment. The Sade application was later used as input when the new version of SMAS was developed.

9.2 Background

The Service development tools provided by the SMAS design environment depends on working directly towards the SMAS database and on the same machine where SMAS is installed. Also, to design a Service, a number of applications need to be started, each performing a certain part of the necessary steps to complete the design. Also, the on-line help is poor, as well as the consistency between the different applications.

Using Sade, most parts of Service Design is embedded into one single application. The platform dependency is solved by, instead of working directly towards the SMAS database, using the *sifio* format as intermediate format and implementing the application in Java, hence making it possible to run on any machine supporting the Java Virtual Machine (for example Solaris or Windows NT).

9.3 Implementation Model

The objects identified are in most cases a direct mapping from SMAS; Controltypes, Services, Service Customer, Service Administrator etc. The Controltype class is subclassed to represent different controltypes, like NRANAX, BRDIR etc. For compatibility with the SIF format, a container class called SIFFile have been introduced. The meaning of this is to hold the same information as in a SIF file. Each object that can be a member of a SIF file have the knowledge how to dump itself in SIF format.

A basic GUI have also been designed. An outline of what menu items the application shall consist of exist, but many of them are just stubs. For more information about the classes and their methods and fields, please see the *javadoc* (automatically generated) documentation.

9.4 Requirements

As a first step, the application need to have enough functionality to make it possible to demonstrate and evaluate it as an alternative/complement to SMAS. I have (below) made a list of possible things that are needed.

1. Add drawing/editing functionality like insert, delete, move, select. This should be implemented in an intuitive way, that is like “ordinary” drawing tools work, not like

SMAS today. For example, it shall of course be possible to select a number of controltypes for moving or deletion.

2. Algorithm for selecting a controltype on the graphical panel.
3. Outlet management. This is one place where SMAS has poor and inflexible functionality
4. Functionality to allow editing multiple Services/Scripts residing in different buffers.
5. Management of Services and Service Logics. It shall be possible to make or edit a Service by grouping a number of SSLs into a Service Logic.
6. A way of separating Logic Module information from Data Modules, making it possible to use the same SSL with different data settings (much in the same way as SMAS works today)
7. Behavior for different Logic Modules, that is IR Parameter descriptions, if datamodules are allowed, Branch or Register mode, etc.
8. Adding and grouping Logic Modules to represent different IN versions.
9. Default name for a Logic Module. Should comply to the way SMAS works, that is the Controltype name (or in some cases an abbreviation of this) followed by a sequence number.
10. SMAS version handling.
11. Service/SSL version handling.
12. Multiple Undo/Redo facilities. Analyze if it is feasible, or if just one level is much easier to implement.
13. Dynamic drawing area resize.
14. Connection between Logic Modules and Data Modules.
15. Data Module Management.
16. Algorithm for SA assignment.
17. Integrating the Report Generator (for inclusion in design documents) within Sade.
18. HTML backend.
19. Printing functionality.

9.5 Possible Extensions

1. Connecting to the SMAS database via JDBC/ODBC, possibly Sybase's *jConnect* driver to retrieve information about Services, SSLs etc.
2. Given the database connectivity, implement a socket interface that reads its input data from a *sifio server* residing on the SMAS boxes.
3. Simulated mode, executing the controltypes and scripts according to the pseudo code in the IN Application Information. This would in effect be the same as implementing the SSI in Java.
4. Possibility to view generated INAP operations.
5. Backend that generates MML commands.
6. An overview window, representing a Feature. Each item in this window represent a Service, consisting of multiple SSLs. This can be used both for overview and navigation, as well as for connecting Services and SSLs into Services/Features.

7. A Macro Generation Functionality. Collect items into a macro, that can be inserted for example like a controltype in an SSL.

9.6 What Sade is

“Sade” is a platform independent Service Application Design and Editing Tool, implemented in Java.

Instead of working directly towards the SMAS database, it uses the Service Interchange Format (sifio) as a layer between the Creation Environment and SMAS. Sade can read and write sif-files. The tool that is used to extract and load sif-files from and to SMAS is called Sifio. Sifio is a command line based program that will extract SSLs, GDMs, and services from SMAS. The output is a human readable, modifiable text file. Sifio can also read this file and copy an SSL, Service, and GDM back into SMAS.

One of the main objectives is to have an application that provides easier and more intuitive editing, and more support for the design process than the standard SMAS tools does today. It should also be possible to provide an integrated environment for Service Development: Service Script Logic design, Service Logic Administration, Global Data Module administration and Service Definition. But it could also provide debugging and testing tools. Also the generation of different service objects documentation could be supported (e.g. like the Report Generator is used today).

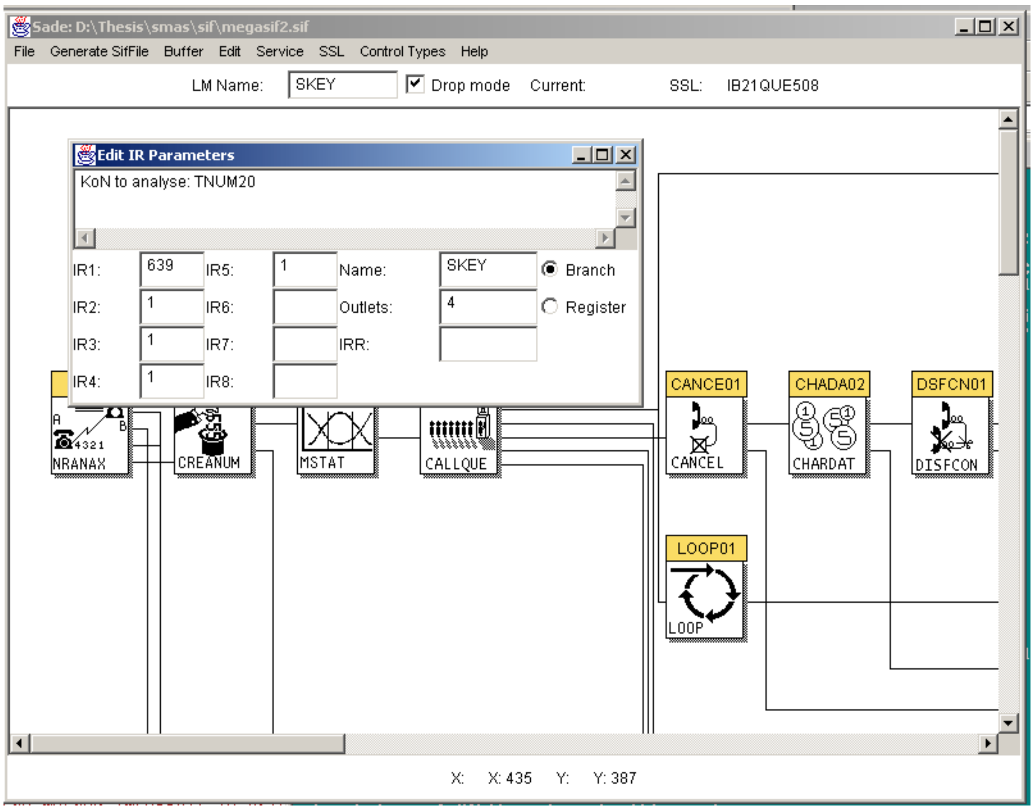


FIGURE 18. Java Based Service Creation Environment, snapshot

9.6.1 General

As a first step towards an integrated creation environment, a quite simple SSL editor was designed and implemented.

An object oriented approach is well suited for this kind of an application: the “world” to model is made of objects – logic modules, data modules, service scripts etc. The objects are easily identified, and the software objects will simply reflect these external objects.

9.7 Service Creation

9.7.1 Service Script Logic

Each Controltype is viewed as an object with a set of operations defined.

A colour coded scheme is adhered to for easy overview, e.g. a uninitiated Controltype is white, a local Data Module is blue, a global red and a customer yellow.

By double-clicking on a controltype a window appears containing the parameters for the controltype.

The number of outlets and connecting controltypes is managed in a more easy way than in SMAS. As long as the maximum number of outlets are not succeeded, controltypes are connected by just right clicking on the "from" controltype and releasing the mouse button on the "to" controltype.

9.7.2 Service Logic Administration

What a Service Administrator does is simply "bundling" a set of Service Script Logics into a package that can then be used as a Service. It wraps it up and provides unique names to access different Service Scripts.

9.7.3 Service Definition

The objectives of having a common environment for service script definition and service data definition is probably the same as those for not having them in the same.

9.7.4 Service Transfer

Sifio itself completely replaces the need of Service Transfer as it is defined today in SMAS.

10.0 Conclusions

10.1 Performance

Operations towards the SCP involves connection overhead and structuring parameters into packets of a fixed size.

- To achieve better utilization it is better if whole blocks of data is read or written at the same time. That is, several subscribers.
- To avoid expensive I/O the intermediate result (the profiles) might be stored in a database.
- To avoid using the slow X.25 connection between SMAS and SCP, the ING RPC interface is used to update the SMAS database, and the C7 interface is used to update the SCP. SMAS update is switched off in the target SCP.
- The result is also the bottom line of this thesis: A migration should be performed with “smart” tools, with customer control switched off, and using high capacity interfaces, avoiding traditional bottlenecks in the system.

10.2 Data Synchronization

From SMAS it is possible to carry out a consistency check, Audit, between the SCP and the SMAS database. It is recommended to perform an audit after a migration, but before the service is taken into traffic.

10.3 Customer Control

If customer control must be working during a migration the data in the target system might differ from its source. Changes made after data has been drained will not be migrated. To allow for this, the data file that is sent from the SCP to SMAS for database update can be used to track changes. After all data is migrated customer control is disabled (this will be easier to control and during a shorter time frame), the delta between the migrated data and the source system is collected using time stamps and the data file.

11.0 Evaluation

The thesis work has involved a lot of studies of the field of migration, as well as learning how Intelligent Networks works. It was hard to grasp the problem in the beginning, since the area of IN was new to me. Also, Java and C++ programming was new to me.

Both using the ING for migration and using an application similar to Sade as replacement for SMAS was work that happened after the Thesis was first written, but in many parts related directly to the thesis work. These sections were added in the second revision of this document.

The result of the study did not end with a working application for migration, but my hope is that it will serve as valuable input for further work in the area. Test programs have been written, that might serve as building blocks or templates for an application. Especially for subscription and customer data module handling.

12.0 Glossary

AP	Application Program
API	Application Programming Interface
DB	Database
DBMS	Database Management System
FP	Functional Processor. Hardware used in the SDP node.
HLS	High-Level Structure. The HLS consists of HLS Access and HLS Update, and provides a structured method for accessing services in the same SCP.
IN	Intelligent Network. SS7 network containing network elements with distributed functionality.
ING	Internet Gateway
INAP	Intelligent Network Application Protocol. Ericsson defined protocol for interaction between IN network elements.
LE	Local Exchange. The exchange in the network that is closest to the subscriber.
PBX, PABX	Private (Automatic) Branch Exchange. An office telephone exchange, such as Ericsson's MD110.
SA	Service Application
SA	Service Administrator
SADE	Service Application and Design and Editing
SMAS	Service Manager Application System
SDP	Service Data Point. IN network element that provides access to subscriber data.
SC	Service Customer
SCP	Service Control Point. IN network element that executes service logic.
SC-ptr	Service Customer pointer. Identify a subscription in SMAS. Used as a pointer in the service.
SDF	Service Data Function. Major functionality of a Service Data Point.
SCE	Service Creation Environment
SIG	SCP Interconnection Gateway
SL	Service Logic
SMS	Service Management System
SIU	SS7 Interface Unit. Hardware unit that supports the SS7 links.
SNAP	SNS Application Platform. Provides high availability and other common functions for a UNIX-based network element.

SNS	Service Node Subsystem. The subsystem that implements the functionality of the SDP.
SSCP	Service Switching Control Point. Ericsson defined IN network element combining the functions of an SCP and an SSP.
SS7	Signalling system No. 7. CCITT defined protocol for common channel signalling (same as CCS-7).
SSL	Service Script Logic
SSP	Service Switching Point. In network element associated with the end office.
TCAP	Transaction Capabilities Application Part. Transaction protocol used over SS7.
TE	Transit Exchange.
TMOS	Telecommunications Management Operating System. Ericsson/HP platform for operational support systems.

13.0 References

[1] Generation Handling Workshop, EPK Karlskrona Januari 31, 1996. Arne Gustavsson

[2] Quickstudy concerning Generation Handling of Service Applications (OX-95:0220). Ulf Seijsing

[3] The Bluffer’s Guide to Intelligent Networks. David Havelin. 1995.

[4] Implementation Proposal (ETO/X/M-95:0286 Uen). J.L. Sakariassen

[5] Common Architecture & Functionality for SA Products (ETX/X/US-95:281). Martin Silvemark

[6] Second Workshop in Generation Handling of Service Applications. (NE-96:0076). Arne Gustavsson.

[7] Service Feature Analysis Report of Migration IVPN 2.5 (5/1597-FCPN 209 024 Uen). ETM/RPK ERBL

[8] Migration Plan of IVPN 2.5 (2/155 17 FAY 103 97 Uen)

[9] OpenClient DB-Library/C Reference Manual

[10] Man pages for bcp

[11] TMOS Users Guide

[12] ING Network Integration Test Report