Informatica

Informatica® SSA-NAME3(EXTN)
10.5

Service Group Generation and Testing Guide

Informatica SSA-NAME3(EXTN) Service Group Generation and Testing Guide
10.5
September 2022

Publication Date: 2023-09-28

Table of Contents

Preface

Welcome to the Informatica SSA-NAME3 Service Group Generation and Testing Guide.

This manual describes the mechanics of producing a Callable SSA-NAME3 module or library (called a "Service Group") by generating the customizable Definition files and combining them with the supplied core modules. It also describes how to test the Callable Service Group using the SSA-NAME3 Test-bed.

The Callable Service Group is the module which provides the run-time Key Building, Search Strategy and Match services to user applications.

Learning About Informatica SSA-NAME3(Extn)

This section describes the various manuals that make up the SSA-NAME3 Extensions for 1.8 Users documentation set. That is, these manuals should be used to generate an SSA-NAME3 Service Group.

Introduction to SSA-NAME3

Provides an overview of SSA-NAME3. It is written in a way that can be read by someone who has no prior experience of the product and wants a general overview of SSA-NAME3. It explains the problems SSA-NAME3 overcomes and provides an overview of how this is done. One chapter is dedicated to providing an overview for Application Programmers.

Getting Started

This manual is intended to be the first technical material a new developer or designer reads before installing or using the SSA-NAME3 software, regardless of the platform or environment. Its goal is to help a new user get the software installed and produce a working prototype application that calls SSA-NAME3 and executes searches against their own data.

To achieve this it provides a "script" to follow which includes pointers to pertinent sections of the other manuals.

Definition & Customization Guide

The SSA-NAME3 software is customizable via modifications to various tables called Definition Files. This manual describes the contents and syntax of the Definition Files and provides tips & techniques for their customization.

Generation & Testing Guide

This manual describes the mechanics of producing a Callable SSA-NAME3 module or library (called a "Service Group") by generating the customizable Definition files and combining them with the supplied core modules. It also describes how to test the Callable Service Group using the SSA-NAME3 Test-bed.

The Callable Service Group is the module which provides the run-time Key Building, Search Strategy and Match services to user applications.

Application Reference

The ultimate goal of an SSA-NAME3 implementation is for application programs to be able to Call on its Services to build keys, effect searches and drive matching.

This manual describes in detail how an Application Program invokes the various SSA-NAME3 Services via the Callable Service Group. It describes the parameters required by these Services, what goes on within a Service, the information that is returned, and what the Application should then do with that information. The manual also contains program pseudo code and topics covering System & Database Design considerations.

Installation Guide

The SSA-NAME3 Installation guide provides information on how to install the product on Windows and UNIX.

Release Notes

The Release Notes contain information about What's New in SSA-NAME3 Extensions for 1.8 Users. It is also used to summarize any documentation updates as they are published.

What Do I Read If. . .

I am. . .

. . . a manager

The INTRODUCTION TO SSA-NAME3 will address questions such as "Why have we got SSANAME3?"

I am. . .

. . . installing SSA-NAME3

Before attempting to install SSA-NAME3 Generation software you should read the GETTING STARTED document. This will tell you about pre-requisites and help you plan the installation and implementation of SSA-NAME3 Service Groups.

The actual installation steps for your platform are documented in the separate SSA-NAME3 Extensions for 1.8 Users 9.5 manual's INSTALLATION GUIDE.

I am. . .

. . . an Analyst or Application Programmer

A high-level overview is provided specifically for Application Programmers in the INTRODUCTION TO SSA-NAME3 SERVICE GROUPS manual. Before attempting to develop programs that interface with SSA-NAME3 Service Groups you should also read the GETTING STARTED manual.

When designing and developing the application program(s), use the APPLICATION REFERENCE FOR SSA-NAME3 SERVICE GROUPS manual as your main guide. This describes the Service Calling conventions, required parameters and provides pseudo code examples.

Working sample programs in various languages can also be found on the SSA-NAME3 CD.

I am. . .

. . . customizing the SSA-NAME3 Definition Files

The DEFINITION & CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS provides all the information required to customize the definition files used by SSA-NAME3. Having done this you will need to generate the actual run-time modules required to link with your application. This is done by performing a process known as Generation. Generation is described in the GENERATION & TESTING GUIDE FOR SSA-NAME3 SERVICE GROUPS.

I want to know. . .

. . . what SSA-NAME3 does

The INTRODUCTION TO SSA-NAME3 manual gives an overview of what SSA-NAME3 does and how it does it.

I want to know. . .

. . . how to develop an Algorithm

Refer to the DEFINITION & CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS in a section called Tips on Customizing an Algorithm.

I want to know. . .

. . . how to develop an Edit-list

Refer to the DEFINITION & CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS in a section called Tips on Building an Edit-list.

I want to know. . .

. . . how to develop a Matching Scheme

Refer to the DEFINITION & CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS in a section called Tips on Developing a Matching Scheme.

I want to know. . .

. . . how to perform Generation

Refer to the Generation Section of the GENERATION & TESTING GUIDE FOR SSA-NAME3 SERVICE GROUPS.

I want to know. . .

. . . where to find the error messages

The non-zero Response Codes which can be returned to Calling application programs are documented in the Response Codes section of the APPLICATION REFERENCE FOR SSA-NAME3 SERVICE GROUPS manual.

Generation messages are documented in the GENERATION & CUSTOMIZATION GUIDE FOR SSANAME3 SERVICE GROUPS.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Service Group Generation and Testing

This chapter includes the following topics:

Overview

This guide describes the mechanics of producing a Callable SSA-NAME3 Service Group module from the supplied core modules, the customizable Definition files, and the user's own data (being the data that will be used for searching and matching). It assumes you have read both the *INTRODUCTION* and *GETTING STARTED* guides. It does not describe the internals of the Definition Files - an in-depth description of each of these can be found in the *DEFINITION and CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS*.

What is Generation?

Once the SSA-NAME3 Definition Files have been customized, they, and an extracted analysis of the user's data, are turned into a Service Group Data File (a `ysg` file).

The Service Group Data File can be immediately used by IIR and DCE. If SSA-NAME3 will be called by a user written application then it will be necessary to generate a callable library (on MS Windows) or shared object (on unix). This final step requires a C compiler.

Note that Unix file systems are case sensitive and the Service Group Data File name should always be kept lower case.

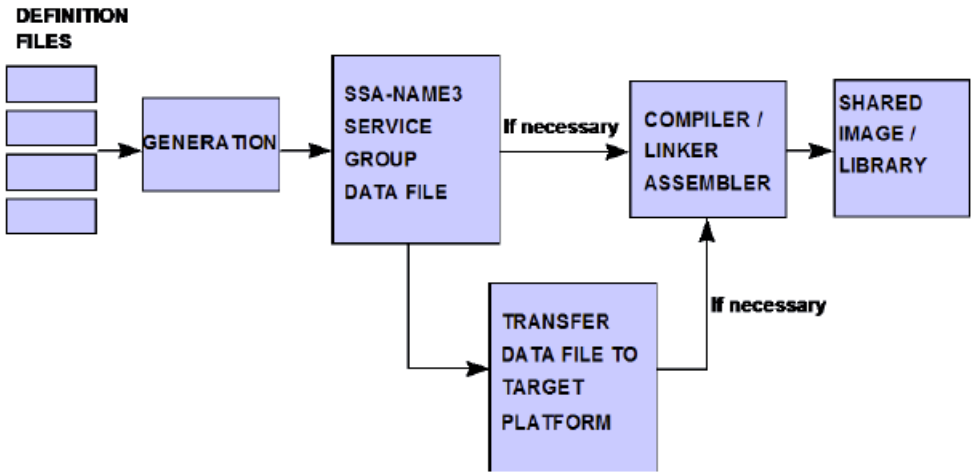
Generation is comprised of a set of utilities which syntax check the Definition Files and create modules from them and from the extracted user data. The Generation steps must be performed on a Windows computer.

When the target platform (i.e. the platform where the applications which will Call SSA-NAME3 execute) is not Windows, the Service Group Data File must be transferred from the computer where the generation was performed to the target environment (the data file is ASCII text).

Generation is performed using Command Line utilities.

Once the Service Group Data File has been produced and stored on the target platform, it can be invoked by IIR, DCE or Testbed. A re-entrant module or shared image/library can be created for linking to your application. An ANSI standard C compiler must be used for this step.

The following diagram is a flow chart showing the low-level steps involved in Generation.



What Gets Generated?

Following are the entities that need to be generated. The Definition files are: the Service Group, which includes both the Service Group and Algorithm Definitions, the Edit-lists, and the Matching Schemes.

These Definition Files should be customized, and the sample user data prepared, before Generation is done.

- Edit-list,
- Sample User Data,
- Algorithms,
- Matching Schemes,
- Service Group

One Edit-list and one file of sample user data are related to one Algorithm. There may be more than one Algorithm, which needs to be generated, depending on the number of population types that need to be searched. There is normally only one Matching Schemes Definition and one Service Group Definition.

Each of the above Definition Files is described fully in the *DEFINITION and CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS*. Following is a brief description of each and an explanation of what is meant by Sample User Data.

Edit-list

An Edit-list is a look-up table containing rules about the commonly occurring words and phrases in a population of data (e.g. nick-names, noise words, abbreviations)

Sample User Data

Sample User Data is used to generate a module called the Frequency Table and a report called the Word Frequency Report. The sample data file must be extracted by the user and be truly representative of the production names or addresses in the system; otherwise, optimum performance will be lost.

You will need one file for each population of names to be searched or matched using SSA-NAME3 – generally, one for each Algorithm being used. If you intend to search on person names as distinct from organization names then you will need a sample file for each. For more information on files of mixed person and company names, see the System Design Notes section in the *APPLICATION REFERENCE FOR SSA-NAME3 SERVICE GROUPS* guide. If you intend to search or match on addresses, a file of these will also be required.

The file need only contain a maximum of 250,000 records as long as these are RANDOMLY selected from the data. The entire file can be used if selecting a random sample is not possible or difficult; however, this will require more disk space and, as the sample file or files should be kept permanently available for future Generation runs, this could have a bearing on the decision.

The Frequency Table, which is generated from the sample user data, consists of the common major and minor words in the data and is used to control the key building and the search logic for a specific Algorithm.

An optional Scaler Frequency Table can also be generated from either the user's sample data or the user's SSA-NAME3 Keys. It consists of SSA-NAME3 Keys and is used to enhance the accuracy of the Scale values returned in the NAMESET Search table ranges.

The Word Frequency Report analyses the Sample User Data to show the common words, phrases and patterns in the population of data. It is used to assist with the customization of the Edit-list and Algorithm.

For some systems where quality is a critical issue, and where the user is prepared to put more time into customizing the Edit-list(s), it may be advantageous to use the entire population of data as the sample file. This is so that the more uncommon variations of words used in Edit-list rules can be discovered.

The sample user file should contain records which are carriage-return, line-feed terminated (as is standard with MS-DOS text files) and can be variable or fixed length. In either case, the records should not be longer than the NAME-LENGTH= parameter that you specified in the Algorithm Definition.

Each record in the file should consist of one field containing the unedited full name or address. If you are building Algorithms for both names and addresses, you will need two sample files. If you are building Algorithms for company names and person names, these two name types should also be in separate files.

For example, for a person name sample file, each name record should consist of one field containing the unedited full name in the sequence specified in the Algorithm definition. For example, if you have NAME-FORMAT=R, then the order should be given-names/family-name and should include titles and suffixes, as it would appear in an address. If you have NAME-FORMAT=L, then the order should be family-name/given names.

The following is an extract from a typical person names file where NAME-FORMAT=R,,

```
A Andrews
A Baker
A Batten
A Bernard
A. Sewter
A. Van West
A.A. Auto Tracey
Aaron Campbell
Aaron Lobb
Azle Lycke
B Andres
B Barrenkatt
Balkenhal. Dempsey
Bama Furseth
Buddy Hilty
```

It is up to you to extract the names into the proper file. Use any means at your disposal, just ensure that the records are in the proper format and that you are using a truly random sample.

If you will be searching on addresses, then it is preferable that whatever data the keys will be built for, is the same data that is provided in the sample file (this is also discussed in the *Algorithm Definition/Tips on Customizing and Algorithm* section of the *DEFINITION and CUSTOMIZATION GUIDE SSA-NAME3 SERVICE GROUPS*.)

For example, if your address entity is structured so that street address (apartment/floor/suite/building, house number, street name) is in a separate field or fields from suburb/town/city, state and postal/zip code, then you may only want to use the street address for your searching. The sample file would therefore contain street addresses only. For example:

```
13 Jamison St.
5/6 Main St
1st Floor Thompson Building 65 General Square
Suite 8a, 12-16 Tryon Road
23 Park Rd
5 The Bouldvarde
Unit 6, Block C, Wan Tan Rd, Symtec Complex
```

Algorithms

An 'Algorithm' contains customizable rules, which control how the various SSA-NAME3 Services operate. Among other things, it defines the specific Edit-list and Frequency-table that will be used. An Algorithm must be authorized before it can be used. Authorization produces an Authorization module, which holds the signatures of each module used by the Algorithm.

Matching Schemes

Matching Schemes define the views and rules, which allow two records to be compared for the purpose of matching.

Service Group

The Service Group groups together all of the SSA-NAME3 Services and Algorithms to be used by a system.

Naming Conventions

SSA-NAME3 has some naming conventions used both internally and externally. There is a set of mnemonics used to identify each type of definition file or module. Those relevant to Generation are shown in the following table:

Mnemonic	Type
MA	Matching Schemes
USG	User Service Group
SG	Same as USG
EL	Edit-list
AU	Authorization Module

Mnemonic	Type
FTB	Frequency Table
TB	Same as FTB
SC	Scaler Frequency Table

These mnemonics are used mainly in file names, for example `n3elus.def` is a definition file for an Edit-list.

What Does Generation Produce?

The purpose of Generation is to produce a Service Group Data File. This data file contains the search and match algorithm options which have been customized within the definition files.

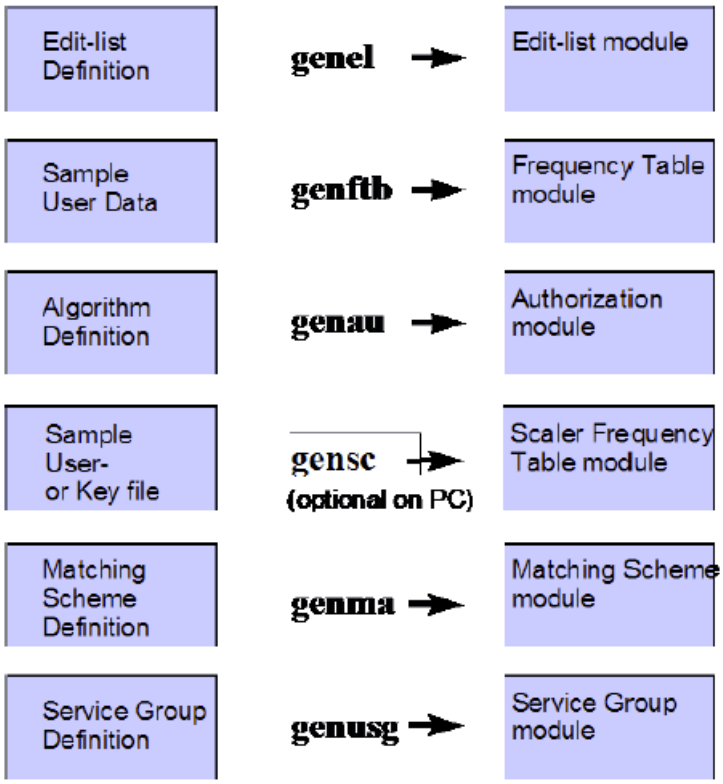
Optionally it can produce a generic SSA-NAME3 shared object / image library for use in an application. This is simply a stub routine that in turn makes use of the Service Group Data File.

The shared object will use the directory set in the environment variable, *SSAUSGDIR* , to detect the Service Group Data File. If this is not set it will refer to its own directory.

The Generation Jobs

While Generation is the process of creating a Service Group Data File from Definition entities, the supplied Generation Jobs take the process to its logical completion.

To complete the picture in the previous section, the following diagram shows the names of the low-level Generation jobs applicable to each of the Definition entities. These jobs are supplied as batch scripts.



The following table gives a description of each job and an indication of the order in which these jobs would normally be run.

Step	Job	Description
1	genel	Edit-list Generation This step parses an Edit-list Definition file and creates an Edit-list module.
2	genftb	Frequency Table Generation This step takes a file of sample user data and creates a Frequency Table module.
3	genau	Authorization Generation This step takes an Algorithm Definition and creates an Authorization module.
4	gensc (optional)	Scaler Frequency Table Generation This step takes a file of the user's sample data or the user's SSA-NAME3 Keys and creates a Scaler Frequency Table module. N.B. Initially, this step must be run after `genau`, but thereafter can be run at any time. Generally, it should be run whenever the SSA-NAME3 Keys are reloaded into the database.
5	genma	Matching Scheme Generation This step takes a Matching Schemes Definition and creates a Matching Schemes module.

Step	Job	Description
6	genusg	Service Group Generation This step takes a Service Group Definition and creates a Service Group Data File.
7	gendll	Dynamic Library Generation This step builds a generic Dynamic Link Library (dll) or Shared Object containing the API for the Service Group.

What do I do After Generation?

The steps taken after Generation differ according to where the target platform is.

If The Target Platform Is Not Where Generation Was Run

If the application resides on a target platform then you need to transfer the Service Group Data File to your target platform. This file is ASCII text so make sure this is considered if using FTP.

Running `gendll` on the target system will create the shared image/library. This will become the callable program which, in turn, makes use of the Service Group Data File.

If The Generation and Target Platform are the Same Machine

This means that the Service Group Data File is to be used on Microsoft Windows. In this case, run `gendll` and the libraries and modules are all ready as a shared image/library.

Verifying Changes Using the Test-bed

Once the Service Group Data File has been prepared, it is a good idea to test your changes with the SSA-NAME3 Test-bed.

If using command level generation, and for other target environments, this involves running the `testbed` executable which in turn will dynamically access the algorithm options in the Service Group Data File. This is described in more detail in the *Test-bed* section.

If you have a different target machine, it is advisable to run the Test-bed on both computers and ensure that the results are the same.

Building or Re-building Your SSA-NAME3 Keys

If this your first Generation after installing SSA-NAME3 (even if it's just a new version of SSA-NAME3) you will need to build and load the SSA-NAME3 Keys for your database file.

If this is a subsequent Generation, you may need to re-build the SSA-NAME3 Keys in your database if changes have been made which affect any of the following modules: Frequency Tables, Edit-lists, Algorithms, Service Group.

CHAPTER 2

Generation

This chapter includes the following topics:

Overview

The information in this chapter assumes that you have already read the *Overview* section or are otherwise familiar with what Generation is.

The following table gives a brief description of each of the low-level Generation jobs and an indication of the order these jobs would normally be run in.

Step	Job	Description
1	genel	Edit-list Generation This step parses an Edit-list Definition file and creates an Edit-list module.
2	genftb	Frequency Table Generation This step takes a file of sample user data and creates a Frequency Table module.
3	genau	Authorization Generation This step takes an Algorithm Definition and creates an Authorization module.

Step	Job	Description
4	gensc (optional)	Scaler Frequency Table Generation This step takes a file of the user's sample data or the user's SSA-NAME3 Keys and creates a Scaler Frequency Table module. N.B. Initially, this step must be run after `genau`, but thereafter can be run at any time. Generally, it should be run whenever the SSA-NAME3 Keys are reloaded into the database.
5	genma	Matching Scheme Generation This step takes a Matching Schemes Definition and creates a Matching Schemes module.
6	genusg	Service Group Generation This step takes a Service Group Definition and creates a Service Group Data File.
7	gendll	Dynamic Library Generation This step builds a generic Dynamic Link Library (dll) or Shared Object containing the API for the Service Group.

The Order of Generation Jobs

The order in which the Generation Jobs are run is important, however, it is not always necessary to run all of the jobs.

The order of the jobs in the above table is roughly the order in which you would perform them if you were starting with a new system and intended to perform the entire Generation process. In this case, you would run Steps 1-3 (and optionally Step 4) for each Algorithm you are using, then Steps 5 and 6.

Usually, however, you will only be modifying an Edit-list, Algorithm parameter or Matching Scheme, in which case you will only need to perform a subset of the jobs. The following table lists the common customization changes that are made to Definition entities and the jobs, which must be run as a result of those changes.

If You Have Changed	Then You Need to Run Job(s). . .
Matching Scheme	`genma`
Edit-list	`genel`, `genftb`, `genau` for the Algorithm affected, `gensc` (if NAMESET's Scale used) and then `genusg`
Sample User Data	`genftb`, `genau` for the Algorithm affected, `gensc` (if NAMESET's Scale used) then `genusg`
Algorithm parameter	`genau` for the Algorithm affected, then `genusg`
Service Group parameter	`genusg`

The Generation Environment

Generation is performed at the command prompt. Use the shortcut to the Generation Window provided in the **Start** > **Program(s)** Menu. Details of the Generation jobs provided in this chapter relate to the syntax of the command level utilities.

If you have followed the *GETTING STARTED* guide, you will have created a user directory containing copies of the Definition Files you will be working with, and one or more sample user data files containing random samples of names or addresses from your production database. It is the members in this directory and this input file which form the primary input to the Generation jobs. Generation jobs should be run in this directory.

Before you can perform Generation, you must always set up the *SSA-NAME3* Environment Variables. This is done automatically if you start the SSA-Name3 Generation Window through the **Windows Start** > **Program(s)** Menu. These environment variables should have been set up by the Installation process in a file called `ssaenv.bat`. If using the command line, run this file to set up the environment before beginning.

Note: Note that where required the Definition File names are specified to the Generation jobs without the `.def` extension.

One environment variable is used to control some global settings for the Generation Jobs. This environment variable is called *SSAGENOPTS* and responds to the following flags:

`-v` will turn on verbose messages (showing the progress of the job)

`-c` will turn on commenting in the generated code.

For Example:

```
SET SSAGENOPTS=-v -c
```

will turn on verbose messages and commenting in the generated code. Other more localized settings for `SSAGENOPTS` are discussed in the sections describing each Generation job in detail.

A provided script, `build.bat`, will complete the necessary generation steps. Otherwise the individual steps described in this manual can be followed.

To get a NM3 Extensions environment you have to do the following:

1. Open a command line window
2. Run the following scripts:

   ```
   C:\InformaticaIR\env\nm3c
   C:\InformaticaIR\setups\defsets
   ```
3. Set the following environment variables:

   ```
   set SSALIB=%SSATOP%\lib
   set SSAINC=%SSATOP%\h
   ```
4. Now set the following environment variable depending on your System Architecture,

   ```
   set SSA_HAVE_X86_TSC=1
   ```

   or

   ```
   set SSA_ARCH_AMD64=1
   ```
5. Run the following setup script to use Microsoft VS2008 compiler, `C:\InformaticaIR\setups\setcmsvs2008`

The compiler flags now need adjusting

```
set SSACCFLAGS=%SSACCFLAGS% -I%SSATOP%\ssaname\h
```

Now either set the *SSAWORKDIR* environment variable to the current directory (".") or explicitly to the working diretory where the `YSG` file is located.

```
set SSAWORKDIR=%cd%
```

This will give you an environment set for the MSVS2008 compiler.

To create a DLL, perform the following:

1. Copy the YSG to the current directory,
2. Run the command,

```
%SSABIN%\gendll <ysg name>
E,g,
%SSABIN%\gendll n3sgus
```

A DLL `n3sgus.dll` is created.

Edit-list Generation

This step can only be run when the Algorithm is marked `GENERATING` or `UNCONTROLLED`. See the *DEFINITION and CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS* for details.

If any errors are detected, the job will fail. Possible errors are incorrect record layout, wrong category type, the same category name defined twice or conflicting categories for the same word. Most messages are accompanied by a line number where the error was detected or by the offending word. Note the messages and check your definition file before re-running the job.

This command invokes the `genel` executable in `%SSABIN%`.

```
genel [SGname] [ALGname] [Edit-list] [-d]
```

where,

[SGname]

The name of the Service Group you are working with.

[ALGname]

The name of the Algorithm.

[Edit-list]

The name of the Edit-list. Must be the same as that defined for this Algorithm in the Service Group Definition file with the `EDIT-LIST=` directive.

-d

An optional switch that will cause the deletion of temporary files as soon as they are not needed. Useful if you are short of disk space.

For example,

```
genel n3sgus PERSON n3elusp
```

If any errors are detected the job will fail. Appropriate messages will appear in the files, n3e1.err and n3e2.err.

After the job has completed successfully, you can discard the intermediate files, n3en.dat and n3en.srt (where `n` is an integer).

Words Changed by Cleaning

If a word in the Edit-list contains a character or characters which would be changed by the Cleaning routine before the word is stored in or looked up in the generated Edit-list, a message will be displayed and the Edit-list generation terminated. The message will read:

```
W044701> Cleaning changed n words.
```

For example, the following rule would normally generate such a message:

```
*C RR R Replacement Word
RR T/AS>TRADINGAS<
```

because Cleaning would remove the "/" before the word reached the Formatting phase.
A list of the Edit-list words that were changed by the Cleaning routine can be obtained by setting the environment variable *SSAGENOPTS*,

```
SET SSAGENOPTS=-showcln
```

This will cause the `n3e1.err` file created by the Edit-list generation to hold a list of words that were changed by the Cleaning.

You should fix or remove the offending character(s), or, instead define the rule as a Pre-Cleaning definition. Using the above example, the following Cleaning Editing definition could be used:

```
*S >T/AS<BA
*W >TRADINGAS<
```

For more information, see the Edit-list Definition/Cleaning Editing Definitions section in the *DEFINITION and CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS*.

Frequency Table Generation

This step can only be run when the Algorithm is marked `GENERATING` or `UNCONTROLLED`. See the *DEFINITION and CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS* for details.

Generation Phase 1 is a large sort. Each successive phase is shorter. A message telling you how many names received a non-zero response code is displayed at the end. If the number is high then you should consider checking your data file. The most common non-zero response codes are for an empty name (020046), or a Formatting loop (070146). For example,

```
W070400> 3 errors 02 found

W070500> 1 errors 07 found
```

This command invokes the `genftb` executable in `%SSABIN%`.

```
genftb [SGname] [ALGname] [namesfile-filename] [FTBname] [-d] [-yOffset] [-zLength]
```

where,

[SGname]

The name of the Service Group you are working with.

[ALGname]

The name of the Algorithm.

[namesfile-filename]

The full filename of your sample data.

[FTBname]

The name of the Frequency Table module to be created. Must be the same as that defined for this Algorithm in the Service Group Definition file with the `FREQUENCY-TABLE=` directive.

-d

An optional switch that will cause the deletion of temporary files as soon as they are not needed. Useful if you are short of disk space.

-yOffset -zLength

Optional switches which allow the specification of the column offset and length where the name or address input field(s) can be found in the input text file. The default offset is 0 and the default length is the whole record length up to the maximum of 256 characters.

For example,

```
genftb n3sgus PERSON names.dat n3tbusp
```

The files, `n3g1.dat` and `n3g1.srt` will be about twice the size of the input file `names.dat` for people names, and three times as large for company names. Both files can be deleted after the job has run successfully.

List of Names with errors

A list of the names which caused errors can be obtained by setting the environment variable *SSAGENOPTS* ,

```
SET SSAGENOPTS=-vv
```

The names in error can be found in the file `n3g1.err` when the job has finished.

Authorization

This step can only be run when the Algorithm is marked `GENERATING` or `UNCONTROLLED`. See the *DEFINITION and CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS* for details.

This command invokes the `genau` executable in `%SSABIN%`.

```
genau [SGname] [ALGname] [AUth]
```

where,

[SGname]

The name of the Service Group you are working with.

[ALGname]

The name of the Algorithm to be Authorized.

[AUth]

The name of the Authorization module to be created. Must be the same as that defined for this Algorithm in the Service Group Definition file with the `AUTHORIZED=` directive.

For example,

```
genau n3sgus PERSON n3auusp%
```

Scaler Frequency Table Generation

This step can only be run when the Algorithm is marked `GENERATING` or `UNCONTROLLED`. See the *DEFINITION and CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS* for details.

The input file may either contain the user's sample data or the user's SSA-NAME3 Keys (specified via the `-k` switch).

If the input file contains sample user data, it initially processes the file as for Frequency Table Generation. A message telling you how many names received a non-zero response code is displayed at the end. If the number is high then you should consider checking your data file. The most common non-zero response codes are for an empty name (020046), or a Formatting loop (070146). For example,

```
W070400> 3 errors 02 found

W070500> 1 errors 07 found
```

A temporary file of SSA-NAME3 Keys is then created, sorted and loaded into the Scaler Frequency Table.

If the input file contains the user's SSA-NAME3 Keys, the keys are sorted and loaded into the Scaler Frequency Table. An error message will be given for keys, which are not recognized as valid SSA-NAME3 Keys.

Initially, Scaler Frequency Table generation must be run after Authorization (`genau`), but thereafter can be run at any time. Generally, it should be run whenever the SSA-NAME3 Keys are reloaded into the database.

This command invokes the `gensc` executable in `%SSABIN%`

```
gensc [SGname] [ALGname] [namesfile|keysfile-filename] [SCname]

[-d] [-yOffset] [-zLength] [-cCount]
[-nFrequency] [-kFormat]
```

where,

[SGname]

The name of the Service Group you are working with.

[ALGname]

The name of the Algorithm.

[namesfile|keysfile-filename]

The full file name of your sample data or SSA-NAME3 Keys.

[SCname]

The name of the Scaler Frequency Table module to be created. Must be the same as that defined for this Algorithm in the Service Group Definition file with the `SCALER-TABLE=` directive.

-d

An optional switch that will cause the deletion of temporary files as soon as they are not needed. Useful if you are short of disk space.

-yOffset -zLength

Optional switches, used for input namesfile of sample user data only, which allow the specification of the column offset and length where the name or address input field(s) can be found in the input text file. The default offset is 0 and the default length is the whole record length up to the maximum of 256 characters.

-cCount

Optional switch, which specifies how many of the most common keys (in the input being analysed) should be included in the output Scaler Frequency Table.

-nFrequency

Optional switch which specifies the minimum frequency of the keys (in the input being analysed) to be included in the output Scaler Frequency Table. The default frequency is 10. This switch should be used to restrict the size of the Scaler Frequency Table.

-kFormat

Optional switch, which specifies the input file containing SSA-NAME3 Keys. The Format is: `KeyOffset`, `KeyLength`, `KeyFormat`, `CountOffset`, `CountLength`, `CountFormat`, `RecSize` where,

- `KeyFormat = b` (binary)
- `CountLength =` the actual Count value specified in this field in the input record is the number of occurrences of this SSA-NAME3 key.
- `CountFormat = d` (decimal, ie. Numeric string) or `n` (no count, ie. `CountOffset` and `CountLength` are ignored).

For example, using sample user data,

```
gensc n3sgus PERSON names.dat n3scusp
```

The files, `n3g1.dat` and `n3g1.srt` will be about twice the size of the input file `names.dat` for people names, and three times as large for company names. Both files can be deleted after the job has run successfully.

Another example, using an input file of SSA-NAME3 Keys,

```
gensc n3sgus PERSON keys.dat n3scusp -n50 -k0,5,b,0,0,n,5
```

In this example, only keys which occur 50 or more times will be included in the Scaler Frequency Table. In the input key file the key offset is 0, the key length is 5 bytes binary, no count field is used, and the record is 5 bytes long.

List of Names with errors

A list of the names which caused errors can be obtained by setting the environment variable *SSAGENOPTS* ,

```
SET SSAGENOPTS=-vv
```

The names in error can be found in the file `n3g1.err` when the job has finished.

Matching Scheme Generation

This command invokes the `genma` executable in `%SSABIN%`.

```
genma [Match] [SGname]
```

where,

[Match]

The name of your Matching Scheme definition file.

[SGname]

The name of the Service Group you are working with

For example,

```
genma n3maus n3sgus
```

Note: The module `SSASCRM` must exist in your working directory for this step. If you have previously executed the BUILD.BAT script in this directory, `SSASCRM` will already have been copied in. If not, it will need to be copied from the `%SSANAME%\H` directory.

Service Group Data File (ysg) Generation

If the Algorithm Definition is marked as `GENERATING` then this should now be made to read `NOT-GENERATING`. If the Algorithm Definition is marked `UNCONTROLLED` it may be left unchanged.

This command invokes the `genusg` executable in `%SSABIN%`.

```
genusg [SGname] [options]
```

where,

[SGname]

The name of the Service Group you wish to generate.

[options]

`-x` will generate the `ysg` file in unobfuscated format for use within DCE or IIR.

All the Algorithms defined in `[SGname].def` that are not marked as `GENERATING` will be included in the new Service Group module.

For example,

```
genusg n3sgus
```

CHAPTER 3

Using the Service Group Data File

This chapter includes the following topic:

Overview

Once the Definition entities have been generated, the result will be a Service Group Data File. The next step is to be able to programmatically call SSA-NAME3 and access the algorithm options within the Data File.

The Service Group Data File will have a name of the format `n3sgxx.ysg` where `xx` is the 2 character county identifier.

On Microsoft Windows

In order to call SSA-NAME3 a DLL must first be generated.
The syntax of the command to generate a 32-bit DLL is as follows:

```
gendll [SGname]
```

where,
[SGname]

The name of the Service Group.

This creates a DLL with the name `[SGname].dll`. For example,

```
gendll n3sgus
```

Will create `n3sgus.dll`. The Service Group Data File will be located at runtime (when using the DLL generated by `GENDLL`) using the environment variable *SSAUSGDIR*. Therefore, *SSAUSGDIR* should be set and point to the directory containing the Service Group Data File.

On Unix

Transfer the Service Group Data File from the Windows computer. Set the Environment Variable *SSAUSGDIR* at its location.

Next, create the shared library using the following command:

```
$SSABIN/gendll [Sgname]
```

where,

[SGname]

The name of the Service Group.

This creates a shared library with the name `[SGname].so`. For example,

```
$SSABIN/gendll n3sgus
```

Will create `n3sgus.so`. The Service Group Data File will be located at runtime (when using the shared library generated by `GENDLL`) using the environment variable *SSAUSGDIR*. Therefore, *SSAUSGDIR* should be set and point to the directory containing the Service Group Data File.

`$SSABIN` will need to be added to `LD_LIBRARY_PATH`. However, there are some platform dependencies associated with this. For example, on AIX it will need to be added to `LIBPATH`.

Note: Unix file systems are case sensitive and the Service Group Data File name should always be kept lower case.

CHAPTER 4

Test-bed

This chapter includes the following topics:

Overview

Changes to the Definition Files should always be checked by using the SSA-NAME3 Test-bed, before the new Service Group is used by the application. This allows you to verify that the changes made are actually having the desired effect.

The Test-bed is a utility, which will accept parameters from an input file or the screen and display the results.

The Test-bed is also useful for application programmers to visualize and check the parameters of Service Calls.

Enabling the Test-bed

This section describes on how to enable the Test-bed.

On Windows

The Test-bed can be run either as a command line program.

- An executable program, called `testbed.exe` is supplied in the `%SSABIN%` directory. Either your current working directory must be the location of your Service Group Data File or you will need to set the *SSAUSGDIR* environment variable to refer to its location.

On Unix

On Unix, the Test-bed can be run either as an interactive or batch program.

- As on Windows, an executable program, called `testbed` is supplied in the `$SSABIN` directory. Either your current working directory must be the location of your Service Group Data File or you will need to set the *SSAUSGDIR* environment variable to refer to its location.

Executing the Test-bed

This section provides information on how to execute the test-bed.

On Windows

The Test-bed is run a Command Line command program.

1. Command Line Command

   Using the Command Line Test-bed, execution can be either Interactive or Batch. In batch mode the input data is provided using the redirection features of the Command Line. For example,

   ```
   %SSABIN%\testbed <testin
   ```

2. This will pass the commands in testin to the Test-bed as though they were typed at the terminal. Optionally the output can be written into a file, for example,

   ```
   %SSABIN%\testbed <testin >testout
   ```

3. This is useful if there will be many lines of output because the testout file can then be browsed. In interactive mode, the Test-bed is invoked as follows,

   ```
   %SSABIN%\testbed -i
   ```

4. This will prompt you to enter the data. This version of the Test-bed also accepts the following commandline switches. To list this, type

   ```
   %SSABIN%\testbed -?
   ```

   This results to output similar to the following:

```
testbed v1.8.2.07

SSA-NAME3 Testbed

usage: testbed [options]

Available options are:
-i[nnn] = interactive; nnn = screen page size (default 25 lines)
-v      = verbose
-q      = quiet
-c      = force compatibility with MVS input length limit
          (truncate input lines that are longer than 80 characters)
-b      = put a newline after each testbed output
-f      = flush the buffer after each output
-stw    = show words in TRACE output
-sg     = show names of linked service groups
          (automatically enables interactive mode -i and selects the
          service group if there is only one available)
-l      = show input line count
-le     = print input line count into stderr
-li     = long input enabled
-rb     = raw browse output
-rc=RC  = only explain the extended response code
          RC = at least the first 6 digits from the 10 digit response code
-t      = show execution time
```

```
-ifIFIL = read input from file 'IFIL'
-ofOFIL = write output to file 'OFIL'
```

On Unix

Execution of the test-bed on Unix is the same as described for Windows Command Line mode in the above section, except that `%SSABIN%` should be replaced with `$SSABIN`.

Test-bed Parameters

The first line in any batch input file must identify the Service Group, for example

```
TEST=n3sgus
```

This will test the Service Group program `n3sgus`.

Using the Interactive version of the Command Line Test-bed, the TEST= can be omitted and just the Service Group name typed.

The TEST= parameter can be followed by any number of *TEST-BED-OPTIONs* as described below. The next required parameter is the name of a Service to be tested. This should be followed by the parameters that this particular Service requires. For more information on Services and their parameters, refer to the *APPLICATION REFERENCE FOR SSA-NAME3 SERVICE GROUPS* guide.

The last parameter for a service should be followed by an escape character (this is normally the = sign, but can be changed through a TEST-BED-OPTION). For example,

```
TEST=n3sgus<= Service Group name
NAMESETP<= NAMESET Service Name
*FINE*<= NAMESET Function
Mike Taylor<= Input name
=
```

Test-bed Options

Test-bed Options can be placed wherever a SERVICE name would normally be expected on the input. The syntax for the option setting command is as follows,

```
TEST-BED-OPTION=[option]
```

where `[option]` is one of the following,

[NO-]REGISTER-CHECK

MVS only. Tests registers before and after a call to a SSA-NAME3 service. If any registers have changed then this is reported in the Test-bed output.

[NO-]TRANSLATE-DBCS

This will translate output names and words from the Test-bed to DBCS format.

[NO-]BOUNDS-CHECK

MVS only. This tests parameters to a service for leading and trailing overwrites, if this switch is specified the Test-bed will report how much of a field was used.

[NO-]LONG-INPUT

Allow lines longer than 80 characters to be entered. If this switch is set then long lines can be accepted by the Test-bed as follows; if a non-blank character is found in position 72 then the input continues on the next line in column 1. The continuation character can be any non-blank character and is not treated as part of the input field.

[NO-]ESCAPE-CHAR[=[echar]]

This option is actually two commands in one and can be used to perform one (or both) of two functions, these being, to change the way blank lines are handled, the [NO-]ESCAPE-CHAR part, and to set the character to be used as an escape character, the `[=[echar]]` part.

- Change Blank Line Handling.

  The NO-ESCAPE-CHAR/ESCAPE-CHAR option allows or disallows blank lines to be entered in the test data. The default behavior of the Test-bed program is to treat a blank line as a break. Specifying `NO-ESCAPE-CHAR` overrides this and allows a blank line to be entered as input, for example,

  ```
  NAMESETP
  **

  Mike Taylor
  =
  ```

  This input to the Test-bed would normally produce an `UNDEFINED SERVICE` message for the Mike Taylor input line because the blank line would have caused the Test-bed to "pop" up one level and expect a Service name on the input stream. If the line,

  ```
  TEST-BED-OPTION=NO-ESCAPE-CHAR
  ```

  is added before the service name (`NAMESETP` in the example) then the blank line will be accepted as input for that service. The command,

  ```
  TEST-BED-OPTION=ESCAPE-CHAR=[echar]
  ```

  enables the blank line escape feature.

- Changing the Escape Character

  To change the escape character from the default of '=', use the command

  ```
  TEST-BED-OPTION=ESCAPE-CHAR=[echar]
  ```

  For example,

  ```
  TEST-BED-OPTION=ESCAPE-CHAR=*
  ```

will define * as the escape character.

Option	Description
LINE-COUNT	This option will cause Test-bed to print the line numbers in the Test-bed output. Each test will be preceded by the line number of the input file where the test data was read from. Numbering starts from 1 and can be reset (during the test) to any number by embedding the command [echar]%Lnnnnnnnn within the input data. **Note:** That nnnnnnnn must be an 8 digit number with leading zeroes. For example, TEST=n3sgus TEST-BED-OPTION=LINE-COUNT NAMESETP *FINE* Mike Taylor =
NEW-OPTIONS	This option allows the new Matching Method option syntax to be specified in a Call to the DEBUG service. The new method options are specified in the DEBUG call after the REPEAT Count. A blank line terminates the specification of one set of Options. Another blank line terminates the specification of all Options. The Options and Values should be specified in alphabetical order. For example, TEST=n3sgus =%C Test DEBUG Call using New Matching Method options TEST-BED-OPTION=NEW-OPTIONS DEBUG 1 LL LCOMP =%C METHOD=LL,EP=N3SCL,ALGNAME= 00000001 00320000 =%C GOPTS=REFMIN+LENGTH*50 00BE0080 =%C LOPTS=CONC+CINITA+NOORDER+NOSTD+I100+INITLOW 00000000 =%C XOPTS= 00000000 00000001 =%C New Options: FLAGS EXACTINI 1 EXACTWRD 1 SCORES INIT 10 =

Adding Comments to the Test-bed Input

Comments may be added to the input scripts in two ways,

```
[echar]%*xxxxxx
```

or

```
[echar]%Cxxxxxx,
```

where `xxxxxx` is the text of the comment and `[echar]` is the escape character (ie. normally the = sign). The first form will not be output by the Test-bed, so it is useful for including internal notes about the test script; perhaps its purpose and expected results. The second form will be echoed by Test-bed and is used to add comments to your output.

Allowing Hexadecimal Input Names

Hexadecimal input names, DBCS for example, can be passed to Test-bed by preceding the hexadecimal name with the following characters:

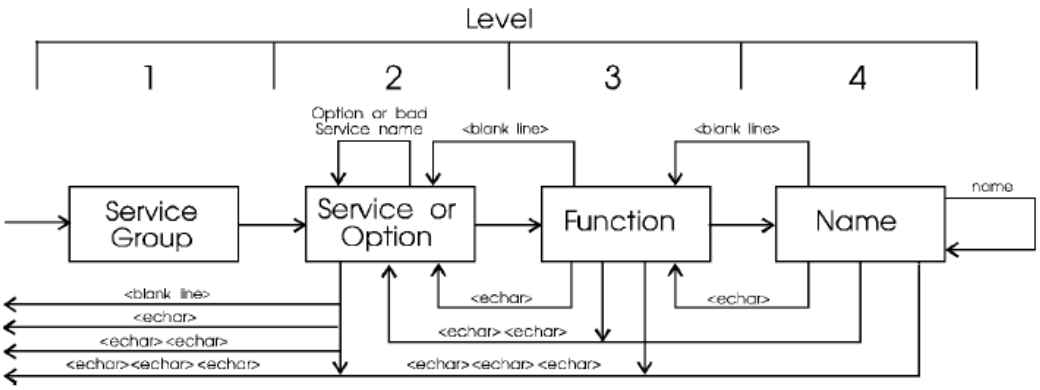
```
[echar]%Xnnnnnn
```

where `nnnnnn` is the hexadecimal name and `[echar]` is the escape character (i.e. normally the = sign).

Hierarchical Levels

The Test-bed works with a hierarchical input system, which is to say that at any time it is expecting input data of a type suitable for a given level. For example at the top level the Test-bed expects the name of a Service Group, once that is entered it is looking for the name of a service etc.

The following diagram shows the flow through a Test-bed session with the `NAMESET` Service, note that levels deeper than 3 depend on the Service being used.



A single `[echar]` entered as a command will return you to the Service level, `[echar][echar]` will go back one level, and `[echar][echar][echar]` will escape the Test-bed completely.

Parameters to Services

This section describes on Parameters to Services.

1. As mentioned the Test-bed reads lines from the input file, each line will be a Service name, a parameter required by the previous Service name or an escape character. The following shows the data expected on the input lines following the Service name using the default escape character (=),

   ```
   TEST=n3sgus
   NAMESETP
   **
   MR JOE BLOGGS
   ROB GRAY
   =
   ```

   This is a typical input file to the Test-bed, each line is described in the following section.

   ```
   TEST=n3sgus
   ```

2. Run the test using the `n3sgus` Service Group.

   ```
   NAMESETP
   ```

3. Call the `NAMESETP` Service.
4. Use the default `NAMESET` Function.

   ```
   MR JOE BLOGGS
   ```

5. Pass this name to the Service, print results.

   ```
   ROB GRAY
   ```

6. Pass this name to the Service, print results.

   ```
   =
   ```

   Terminate this call, this may either be the end of the file or preparation to call another Service. The parameters required by each Service and the level of each parameter is shown in the following table.

Service	Lvl	Parameter name	Example
NAMESET	3	Function	`*FINE*`
	4	Name	
MATCH	3	Function	`*SCORE-ONLY*`
	4	Scheme name	RANK
	5	Search record	Bill Bloggs
	6	File record	William Bloggs
	4	Algorithm name	`PERSON`
	3	Function	2
	3	Function	6
DEBUG	3	Function	1
	4	Scheme name	
	5	Method name	
	6	Weight	An 8 digit hex number
	7	Global options	An 8 digit hex number
	8	Local options	An 8 digit hex number
	9	Field offset	An 8 digit hex number
	10	Repeat count	An 8 digit hex number

Service	Lvl	Parameter name	Example
			Lines 5-10 can be repeated if more than one method is used in the Scheme. A final line with all blanks must follow to indicate the end of the Scheme definition.
	3	Function = 2	
	4	Algorithm name	
	5	Parameter	
INFO	2	Table	ALGSIG
	3	Function	1
	4	Key1, Key2, Key3	PERSON
SSAPHO	3	Function	1
	4	Input Word	Bloggs
SSAMAJ	3	Function	1
	4	Name	Bill Bloggs
SSACLN	3	Name	Bill Bloggs
SSAFMT	3	Name	Bill Bloggs
SSASTD	3	Word	Bloggs

Example Test-bed Output

The following shows an example of the output to expect from Test-bed. It shows basically what an Application program would get returned if it called that Service.

```
PGM:   TESTBED TST   1.8.2.07MSVC60
Aug 22 2002

DONE:  TESTBED --DONE--
Sep 03 2002

USG:   S\_MDT  MDT    1.8.2.0800255YLN5NY
SERVICE GROUP NAME: n3sgau
NOTE: THIS PROGRAM CONTAINS CONFIDENTIAL INFORMATION AND
 IS THE SUBJECT OF COPYRIGHT, AND ITS ONLY PERMITTED USE
 IS GOVERNED BY THE TERMS OF AN AGREEMENT WITH
 INFORMATICA CORPORATION, OR ITS SUBLICENSORS.
 ANY USE THAT DEPARTS FROM THOSE TERMS, OR
 BREACHES CONFIDENTIALITY OR COPYRIGHT MAY EXPOSE THE
USER TO LEGAL LIABILITY.
SERVICE: NAMESETP
TYPE:    NAMESET
ALG:     PERSON
```

```
WSIZE:   4373     SSIZE: 2080
RESP:    00
FUNC:    **
INPUT:   JOHN SMITH
OUTPUT:  JOHN SMITH
STACK:   02
  1 JOHN              Y OO J JOHN
  2 SMITH             M Y
S   SNAT

KEYS:    02
 1 C79A437E76 A1
 2 F061C00000 F4
STAB:    F061C00000
 1 F061C00000 F061FFFFFF 60 04 20 F4 C 00
 2 F05D800000 F0627FFFFF 75 12 11 D5 C 00
 3 F040000000 F07FFFFFFF 80 20 10 D6 C 00
 4 EA80000000 F3BFFFFFFF 95 31 01 A8 C 00
 5 0000000000 FFFFFFFFFF 00 40 00 A9 C 00

CATS: OO.
WORK: F4                                    0000310000000000310000

SSA-NAME3 TESTBED PROCESSING COMPLETE
```

CHAPTER 5

Utilities

This chapter includes the following topics:

Word Frequency Report

The Word Frequency Report is produced by processing your sample file of names or addresses to determine the common names, common words and common word patterns in your Population.

It is strongly recommended that this utility is run to assist with the customization of the Edit-List tables, Customset rules, and Account-Name rules. This step should be run using different input files for as many Algorithms as are being used.

When you have the sample file, you are ready to run the Word Frequency Report using the command line tool, `genrep`.

This step can only be run when the Algorithm is marked `GENERATING` or `UNCONTROLLED`. See the *DEFINITION and CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS* for details.

`GENREP` has the following syntax:

The format of the Command Line genrep command is described below.

```
genrep [SGname] [ALGname] [namesfile-filename] [Switches]
```

where

[SGname]

The name of the Service Group you are working with.

[ALGname]

The name of the Algorithm.

[namesfile-filename]

The full filename of your extracted names.

[Switches]

A set of optional switches which control the behavior of various steps. These are listed and described below.

Switches

The following table lists the switches in alphabetical order along with a one-line description of each option. The table is immediately followed by a more detailed description of the switches.

Switch	Short Description
-a	Analyze account name patterns.
-bXXX	Report all occurrences of the word XXX (after cleaning).
-d	Causes the deletion of temporary files.
-dbcs	Specifies that the input file is double-byte character set data.
-e[n]	Strings of consecutive consonants are reported.
-fmt	Words selected only after cleaning & formatting.
-g[n][noinit]	Analyze groups of n words, ignoring initials.
-g[n][noword]	Analyze groups of n initials, ignoring words.
-gwhole	Analyze whole names.
-gwholeasis	Analyze whole names (no cleaning or formatting)
-hn	Report all n-word names (after formatting).
-init	Causes initials to be included in the report.
-iXXX	Report all occurrences of the sub-string XXX (after cleaning).
-j	Show the word length on the final report.
-k	Include the stabilized form of the word in the report.
-ln	Causes only words of length n to be reported.
-ns	Sort the final report by name.
-oXXX	Names the final report file as XXX.
-px[-y]	Report prefixes of length x to y (inclusive).
-rn	Limit the size of the report to n lines.
-sel	Produce a NAMESET search selectivity report.
-std	Stabilizes the word before doing the frequency analysis.
sx[-y]	Report suffixes of length x to y (inclusive).
-t<types>	Select only words with specified Word-type(s).
-uTitle	User specified report title.

Switch	Short Description
-w	Analyze word and initial patterns.
-word<digit>	Exclude the positionally selected word from the report.
-x	Produces a shortened form of the reported word output.
-yOffset -zLength	Specifies the column offset and length where the name or address input field can be found in the input text file.

The following pages give a more detailed description of each of the switches. The switches are categorized into the following groupings:

- Name, size and sequence of the report
- Pre-selection options
- Reporting words
- Reporting phrases
- Reporting initials
- Reporting sub-strings
- Reporting patterns
- File formats
- Temporary files

Name, size and sequence of the report

The following parameters can be used:

-oXXX

Names the final report file as XXX. If this parameter is not specified, the report's name defaults to [ALGname].REP.

-ns

Sort the final report by name. The default sort order is by frequency as the major key, followed by name as a minor key.

-rn

Limit the size of the report to n lines. If not specified, the number of lines in the report file is determined by the REPORT-SIZE= parameter specified in the Algorithm definition file.

-uTitle

Allows a user specified title to be inserted as the first line in the report. If the title contains blanks, specify it inside double quotes.

Pre-selection options

Option	Description
-fmt	Will cause words to be selected only after Cleaning & Formatting. The default is to not Format the name first. For a description of Formatting refer to the *APPLICATION REFERENCE* guide. When you specify `-fmt`, you must have already defined and generated an Edit-List because the Formatting process will refer to it. This would generally just be a very small Edit-list containing, say, the very common noise words which you do not wish to get in the way when reporting phrases or patterns.
-t<types>	Only words with a type listed in `<types>` will be selected from the word stack. Valid types are M (major), Y (select), N (maj-code), C (code), S (skip) and I (initial, but only if the `-init` switch is also specified). Example, `-tmcs` will select major, code and skip words. The `-t` switch requires the `-fmt` switch also to be specified. For a description of Word-types, refer the *APPLICATION REFERENCE* guide. **Note:** For Codes to be picked up, `FORMATTING-OPTIONS #1` (and optionally #2) in the Algorithm Definition must be set to C.
-word<digit>	Where `<digit>` is one or more digits in the range 1 through 9. The words from the cleaned name are counted in left to right order and the word corresponding in position to the digit(s) specified is/are included in the frequency report. For example, if it is known that all names start with a salutation, then instead of building a small Edit-list with the salutations as delete words and then using the -fmt option, this option can remove all first words by specifying `word23456789`. This is mutually exclusive with the `-fmt` switch.

Reporting words

Option	Description
`-bXXX`	Will find all occurrences of the word `XXX` (after cleaning) and report them to the file `n3r1.err`. This file is created in addition to the frequency report file.
`-h1`	Report 1-word `names` (after formatting). Names which qualify are written to `n3r1.err`. This file is created in addition to the frequency report file. This switch requires the `-fmt` switch to be specified.
`-j`	Show the word length on the final report. When used in conjunction with `-std`, shows the stabilized word length rather than the original word length.
`-k`	Include the stabilized form of the words as an attribute in the report
`-ln`	Will cause only words of length n to be reported. This parameter is mutually exclusive with the `-p` and `-s` switches.
`-std`	Stabilizes the words first and then reports on the frequencies of the stabilized words.

Reporting phrases

Option	Description
-g[n][noinit]	Analyze and count the frequency of groups of words from the cleaned or formatted name. n is used to specify the number of words in a group and defaults to 2. This can be used to find common pairings of words. By default, initials will be considered to be a 'word' but this behavior can be turned off by specifying noinit. If it is desired to use the formatted name, -fmt must also be specified.
-gwhole	Analyze and count the frequency of whole names. The names will have been passed through Cleaning only. To use the name after Cleaning and Formatting, specify the -fmt switch. To use the name before Cleaning, see the -gwholeasis switch.
-gwholeasis	Analyze and count the frequency of whole names. The names will be as they are encountered in the input file, that is, not Cleaned or Formatted.
-h[n]	Report n-word names (after formatting). Names which qualify are written to n3r1.err. This file is created in addition to the frequency report file. This switch requires the -fmt switch to be specified.

Reporting initials

Option	Description
-init	Normally, initials are not reported. Use this switch if you want initials included in the report.
-g[n noword]	Analyze and count the frequency of groups of initials from the cleaned or formatted name. n is used to specify the number of initials in a group and defaults to 2.
-t I	Only words with a Word-type of I (initial) will be selected from the word stack. The -init switch should also be specified if this switch is set.

Reporting sub-strings

Option	Description
-e[n]	Will cause strings of consecutive consonants to be reported. If n is specified, only strings of length n will be reported, otherwise all strings that are at least two characters in length will be reported. This parameter can be used in conjunction with the -l, -p and -s parameters. These former parameters are used to select candidate strings of specified length from the words, which are then analyzed for consonant strings. For example, you could obtain a frequency report of words ending with two consonants by specifying -s2 -e2.
-iXXX	Will find all occurrences of the sub-string XXX (after cleaning) and report them to the file n3r1.err. This file is created in addition to the frequency report file.

Option	Description
-px[-y]	Will cause word prefixes of length x to y (inclusive) to be reported. This parameter is mutually exclusive with the -l and -s parameters. A prefix is a subset of a word; it never matches a whole word.
-sx[-y]	Will cause word suffixes of length x to y (inclusive) to be reported. This parameter is mutually exclusive with the -l and -p parameters. A suffix is a subset of a word; it never matches a whole word.

Reporting Patterns

The following table provides details on the options:

Option	Description
-a	Analyze and count the frequency of Account Name patterns. This requires that the Account Name Markers are first defined to the Edit-list. For a description of Account Names, refer to the *DEFINITION and CUSTOMIZATION* Guide.
-w	Analyze and count the frequency of word and word initial patterns. If used in conjunction with the -fmt switch, the patterns are based on the formatted words stack, that is, the major word will be first, followed by the minor words and initials in left to right order. If not used with the -fmt switch, the patterns are reported in the order of the words and initials in the cleaned name, that is, left to right. For more information on how to use the analysis of these patterns, see the Algorithm Definition/ Customset section in the *DEFINITION and CUSTOMIZATION Guide*.
-sel[NAMESET Function]	Produce a NAMESET search selectivity report of the form: Min Avg Max Searches STOP=WWWWI 0 0.00 0 0 STOP=WWWW 0 0.00 0 0 STOP=WWWI 2 2.00 2 1 STOP=WWW 1 1.86 4 11 STOP=WWI 1 1.40 4 35 STOP=WW 1 1.27 6 998 STOP=WI 1 2.14 9 998 STOP=W 3 17.12 88 1000 \textbf{STOP= 4 282.55 724 1000}

The default NAMESET Function used is FINE. Other NAMESET Functions can be used by tagging them to the option. For example:

```
-selCOARSE
```

To use more than one NAMESET Function, a comma separates the keywords and quotes are required around the whole option. Note that ** is not required to delimit the Functions. For example:

```
-selCOARSE,STOP=WW
```

The report shows the minimum, average and maximum number of records that will be read if you do a positive search and STOP at certain levels. The "Searches" column tells you how many names in the population (input file) contained enough words to produce a search range at a specific level. From the example it can be seen that this population had mostly two word names. The "Avg" number of records read column shows that it would be a mistake to process the last range, but stopping at the WI level would be tolerated although it is about twice as expensive as a search at the WW level.

Probes, Customset and Secondary ranges appear before the positive search ranges and, if present, are counted as if they are always processed first. So, a `STOP=WWWWI` may not normally have any positive searches that qualify for this level, but you will see the extra probes and ranges added to its, and all other, counts. To use this option you must have already defined and generated the Edit-list and Frequency Table for the Algorithm which will be used by the genrep utility; however, the Algorithm does not need to be authorized. Remember that the values in the `Min`, `Max` and `Searches` columns relate to the actual number of records in the input file. If this file is not your complete file of names and you want more accurate figures, either use the full file as input, or extrapolate the results.

File formats

The following are the options that can be used:

Option	Description

`-x`	This is used with `-b`, -i or `-h`. It limits the output written to `n3r1.err` to just the original name. For example, without `-x`, `n3r1.err` contains,
	
	`M011100> Found substr "ADAM" on line 101`
	`M011202> Original line was:`
	`"Adam Chwasz "`
	`M011100> Found substr "ADAM" on line 102`
	`M011202> Original line was:`
	`"Adam Mammano "`
	
	With `-x`, `n3r1.err` contains,
	
	`Adam Chwasz`
	`Adam Mammano`

`-dbcs`	Specifies that the input file is double-byte character set data. The output report will be the same

Temporary files

The following are the options that can be used:

Option	Description
`-d`	This switch will cause the deletion of temporary files as soon as they are not needed. Useful if you are short of disk space. For example, `genrep n3sgus PERSON names.dat` will generate a file called `person.rep`. With no switches, the report will contain only a list of words from `names.dat` sorted in descending frequency order (the most common words come first). This file can now be browsed to select those words that you believe require special treatment (Example, exclusion from the key, replacement, concatenation). Those words are shown in a format suitable for adding directly to an Edit-list definition file. You may extract the selected line with an editor, add a Category type at the beginning and update your Edit-list definition file with the added lines. The files `n3r1.dat`, `n3r1.srt`, `n3r2.dat` and `n3r2.srt` can be discarded (or use the `-d` option).

SSASRT.EXE

SSASRT is a utility program available on Windows which sorts fixed-length records in a file by any number of keys. SSASRT has the following syntax:

```
ssasrt <InFile> <OutFile> [t/v] <RecLen>
     [<Pos> <Len> <Order>...]
     [-w<path1> [<path2>]] [-mn] [-rn] [-v] [-d]
```

where,

<InFile>

The input filename.

<OutFile>

The sorted output filename.

[t/v]<RecLen>

The length of the input record, optionally preceded by a format flag, `t` or `v`. If neither `t` or `v` are used, the input record is assumed to be fixed length. If t is used, this indicates the input is text format (cr/lf terminated) and v is used to indicate variable length records (which are preceded by a 2-byte length).

The following three parameters can be repeated any number of times to sort on multiple fields.

<Pos>

The position of sort field (starts from 1).

<Len>

The length of sort field in bytes.

<Order>

The sort order (`a` = ascending, `d` = descending).

-w<path1> [<path2>]

Path to a working directory(s), up to two can be defined. This can be used to make the sort more efficient by,

- Defining two working directories on physically separate drives. This will considerably reduce head movements.
- Defining a RAM drive as a working drive.

Example

```
-wc:\temp d:\temp
```

-mn

Defines the maximum amount of memory `SSASRT` will use, where `n` is in units of Kb. The default is to use all available memory which is obviously the fastest.

-rn

Defines the amount of memory `SSASRT` will reserve for other programs – `n` is the amount in units of kB.

-v

Verbose, print progress messages.

-d

Delete the input file when finished.

This program can be used to sort key data, Edit-lists and Word Frequency Reports.

SSASRT is only licensed to be used within the SSA-NAME3 customization, generation and testing environment.

GENCS

The `gencs` utility, available only on Windows, can be used to generate two programs, `a2e` and `e2a`. These are conversion programs which will convert data from ASCII to EBCDIC and EBCDIC to ASCII respectively. These programs use the conversion tables specified in your character set module and can handle embedded Double Byte characters. The syntax is as follows:

```
gencs [SGname] [ALGname] [-d]
```

where,

[SGname]

The name of the Service Group you are working with.

[ALGname]

The name of the Algorithm that contains the Character Set module to be included in the conversion programs.

-d

An optional switch that will cause the deletion of temporary files as soon as they are not needed. Useful if you are short of disk space.

For example,

```
gencs n3sgus PERSON
```

GENXREF

This utility, available only on Windows, is used to check the validity of a particular Character-set's ASCII to EBCDIC (Table 12) and EBCDIC to ASCII (Table 13) conversion tables.

If you have modified the tables, you may run this utility to produce a report.

`genxref` has the following syntax:

```
genxref [SGname] [ALGname] [Report Name]
```

where,

[SGname]

The name of the Service Group you are working with.

[ALGname]

The name of the Algorithm which contains the Character-set module to be validated.

[Report Name]

The name of the output report file.

INDEX