Informatica® SSA-NAME3(EXTN)
10.1

Introduction to SSA-NAME3 Service Groups

Informatica SSA-NAME3(EXTN) Introduction to SSA-NAME3 Service Groups
10.1
June 2018

Publication Date: 2018-07-04

Table of Contents

Preface

Welcome to the Informatica SSA-NAME3 (Extn) Introduction Guide.

This guide is intented for users who has no knowledge of the product and wants to know about it. It explains the problems SSANAME3 overcomes and what approach it uses in doing so.

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 Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- 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 search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.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

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

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

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access 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. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access 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 augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

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

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

CHAPTER 1

Introduction

This manual is intended for new users of SSA-NAME3. It provides a background into the Name Search problem and an overview of Informatica Corporation approach. It can be read by any person involved with name searching - the IS manager, system designer, end-user, business analyst, analyst/programmer, DBA or systems programmer.

The Need for Name Search

As the use of computer systems has evolved, a growing mass of data about people and organizations has been, and is continually being, collected and processed. In most cases this data is associated with a formal identity such as an account number, national identity number etc., and it is true that the majority of accesses to this data will be made by that identification number.

In many countries, however, every important service that could be provided to a person, company, business or household, has been computerized, and this has led to a proliferation of identification numbers. Due to the size of our growing populations, this has also meant that such numbers and codes are increasing in diversity and complexity. The opportunities for an identity number to be incorrect are increasing every day, despite our attempts at reliability through the use of check-digits, bar-codes and codes built from actual identification data.

In addition, some systems must cope with finding and matching data when there is no stable or reliable identification number available, (for example, police persons of interest, directory inquiries, prospect and marketing lists, intra-organization data matching, check payments with no payment slip, fraud investigation systems, grouping of accounts in a bank or insurance company application, data warehouse creation, credit reference checking etc.).

Another factor affecting the availability of identity numbers is that many people do not have them readily available when making an inquiry or filling out a form, and often even if they do, do not provide them.

The need for retrieval or matching of databases on names and addresses has become quite common and well known and the number of applications where ’name searching and matching’ techniques are needed is growing rapidly.

The Name Search Problem

The nature of applications requiring name searching and matching vary considerably as does the relative importance of ’name search’ between different applications and users.

In the past, this 'name search' problem has been studied and researched for some very restricted application areas where successful retrieval was critical (e.g. law enforcement searches). The more general approach to solving the name search problem has been for systems designers and analysts to design their own solutions typically using methodologies such as exact match alpha key, soundex, match-code, wild-card and text retrieval, and to apply the same solution to all system areas requiring name search.

Each of these methodologies we will call a name search 'Algorithm'.

The growing duplication of records in databases, the increasing frustration of customer service operators at slow or unreliable name searches, and worsening fraud problems based on name or address variations, all point to the fact that most of these Algorithms are, alone, not adequate for today's volumes of data and nature of society.

The reasons are many. As soon as a system requires a search by name, the designers and eventually the users start encountering some or all of the following problems:

- Errors made in spelling the spoken name.
- Transcription errors for written names.
- Missing first names or initials.
- Mixed usage of first names and initials.
- Nick-names, abbreviations, synonyms, unintentional concatenation or splitting of names
- Extra words and word sequence variations.
- Growing multiculturalism bringing more and more names and name structures which are not easily recognizable by the 'locals'.
- Failures to find all parts of compound or account names.
- Anglicization (Localization) of names causing variation between formal name as on Driver's License and informal names on other documentation.
- The problems created by the frequent use of certain common last and first names.

If the application has any significant volume of data at all the following problems will arise:

- Length of response time of the system before an answer is available.
- The problem of the system eliminating relevant names, and on the other hand, of showing too many names to make a choice.

If the problem is of special concern or is researched fully, the following points are often encountered:

- The design of dialogues so that neither the operator nor the system comes too quickly to the conclusion that there is not a relevant match. (e.g. volume can cause data to be missed).
- That increasing the width of the search to allow for more error significantly aggravates the response time and performance problem.
- That progressive refinement of the system by addressing special cases introduces undiscovered problems elsewhere and progressively degrades the system.
- That the system's name rules cannot be changed unless all files are fully reprocessed according to the new rules.
- That integration of data from different systems into an integrated search leads to new frustration for users because of variations in the name handling.
- That a change in the Name search algorithm may improve overall performance and quality while achieving less success for certain previously satisfied special cases.

The On-Line Response Time Problem

An important characteristic of a name search is the response time it takes before the search Algorithm presents a good candidate to the user.

The ’on-line response’ performance of a name search Algorithm is an important concept. An Algorithm that analyses thousands of records and, after a long period, supplies a small group of candidates to a user is usually less acceptable than an Algorithm that can rapidly supply the user with a few highly probable candidates, but takes quite a long time to display the low probability candidates.

An example of the difficulty with this aspect can be seen in those algorithms that provide an exact match as a fast path to the file entries. While there is certainly a fast response to some matching records many other records that are from a business point of view just as significant (e.g. minor variations only), are not available unless the rest of the algorithm is used. This exact match with its quick response often leads to the other entries being ignored.

The important aspect to consider is that the algorithm used must not force users to either extreme. The algorithm should allow rapid access to records of a particular significance. Each search dialogue that is designed should be able to take any level or depth of search that is relevant to the problem and should not be constrained by the algorithm.

Most implementations of name search algorithms do not provide multiple levels or depths of search from a physical access point of view. Such algorithms provide one ’group code’, ’coded name’ or ’phonetic key’ that is used to select or access a ’bucket’ of file entries. This whole group or bucket is then analyzed to choose what to show to the user. Of course, this group can be ’scored’ to achieve a particular probability to decide what to show the user. However, if the first level or depth of search is inadequate the whole group or bucket is reprocessed to provide the next level. With large volumes these buckets or groups are themselves very large.

Good algorithms will actually allow buckets or groups to be subdivided based upon the concept of level, depth or probability such that only the records of appropriate level, depth or probability are physically accessed.

The Name Distribution Problem

The most confusing and aggravating characteristic of files of names is the unusual distribution of the actual names. It is common knowledge that there are a few family names that encompass large groups of each population. It is also common knowledge that this is so for given names.

It is not so obvious as to how extreme this distribution really is.

It is not unusual to find several common surnames (e.g. `SMITH` or `WILLIAMS`) in a population of file entries where each accounts for in excess of 1% of the population (thus on a 5,000,000 record file the group with that surname may exceed 50,000 entries).

What is not usually realized is that this fact is devastatingly important as, not only is the file distributed in such a skewed fashion, it is also usually true that the queries or searches will be identically distributed. That is, that in excess of 1% of the searches can be on one common surname.

If you extend this observation, to the fact that usually 10% of searches made will, with the above example, access a surname group where at least 25,000 entries exist, one can imagine how easy it is to bias the design of an algorithm to the ’common names’ area.

Conversely the distribution has an enormous ’tail’ of very uncommon names where very few members of the population have these names. If the algorithm design is biased towards performance for this ’tail’ it also usually aggravates the problem for common names.

In fact algorithms that are badly formulated often confuse a large percentage of the uncommon names with the common names they were derived from.

This distribution problem is not as stable as most designers would imagine. In a particular country its name distribution characteristics may be stable, but imagine a system specializing in Vietnamese migrants where 30% of the population hold one surname and another 15% has another (ie.45% of the population is covered by two surnames.)

The most successful name handling algorithms have to be aware of or designed for a specific population of names.

The Variation Problem

The reasons, that two reports covering the same individual person (organization or address), end up with differing variations of the person's names stored in the system, are many. Understanding these variations will lead to an appreciation of the 'search' problem.

Phonetic Variation

Where names are spoken, especially over a radio or telephone, a whole class of variations in the spelling can occur. This is usually referred to as a phonetic problem and the recognition of its existence leads to such original algorithms as `PHONIC` and `SOUNDEX`.

This phonetic variation is itself compounded by the fact that even when a name is spelled out by saying the letters, a degree of phonetic confusion can still occur.

The false presumption of many algorithms is that the phonetic problem is in its own right the major variation.

There is in fact some evidence to suggest that phonetics accounts for less than 25% of the variations.

Subconscious Correction

Probably one of the most common reasons for variation is to do with automatic or subconscious "correction" of names that have sounds or letter combinations that are very similar to common names. For example, `SMITHY` becomes `SMITH`; `WILLIAM` as a family name becomes `WILLIAMS`.

Such variations are often well handled by 'phonetic' algorithms.

Orthographic Variation

A significant amount of error can occur when transcribing names from paper to paper or computer terminal. This type of error is often mechanical and can be keyboard dependent (e.g. `R` instead of `E` on a QWERTY keyboard). This error is often a mental one as in transcription or truncation (e.g. `beth` becomes `beht`).

However, the major form of this error is to do with substitution of a graphically similar letter when using hand writing (e.g. `G` for `Q` or `S` for `Z` or `M` for `N`).

Real Variation

A possible but usually low volume problem is associated with name changes. The familiar one being associated with marriage and divorce.

The most normal and fortunately addressable problem is Anglicization (more generally localization into local language, style or dialect). In populations where foreign migrants are frequently introduced it is normal to adjust the pronunciation and then the spelling of a foreign name to fit into local conventions.

Sequence Variation

A large class of variation arrives from the fact that several words can be used to make up either a surname or a person's given names.

In some cases words are left out, especially middle names. In other cases they are re-sequenced. In certain cases, the set of words used is a choice from one of two or three subsets of a group of name words. This is typical of names given in cultures where it is normal to adopt new legal given names at puberty or on coming of age (e.g. `Papua New Guinea`) or in Western style countries where eastern faiths are common (e.g. `Fiji`).

One of the most complex cases encountered is that where identification of the family name is difficult.

This can arise for many reasons not the least of which is frequently used names that can be either family or given names (e.g. `William Andrews` or `Joseph James`).

There are also several populations where the practice is to create compound family names out of both parents' family names. In certain Spanish, Portuguese and Far East countries this problem is exaggerated by the fact that different sequences are used by different members of the same family when referring to the same individual.

The Algorithm Design Problem

When one sets out to develop an algorithm that solves the problems previously described, one encounters a whole class of new project management and testing problems.

- To establish test data to test volume performance for on-line search is difficult let alone expensive.
- To establish test data to allow one to examine algorithm performance requires a representative set from a real population. The data one is interested in could be as low as 0.1% of a real population. Identifying it is nearly impossible.
- The absence of objective criteria for deciding if a change to an algorithm is right or wrong leads to empirical testing only. This means that simulation testing is necessary across the whole population of names. It is no good testing test cases or problem cases because every change to the algorithm introduces both benefits and disadvantages. The only process for deciding to accept the change is to measure the net gain in benefit in real use on a real population. The extreme skew distribution of names coupled with the high degree of refinement being sought leads to one discarding sampling even when working on very large populations.
- The relative significance of problems, with an algorithm, change with volume. (An example would be the barrier one goes through when a set of candidates no longer normally fits on one screen in a dialogue).
- The preoccupation of the designer, programmer, and users with "special cases" leads to an enormous waste of time.
- The fact that the algorithm needs to work on different populations of users in one organization can confuse the decision making.
- The reluctance of users to accept new algorithms that give ’better answers’ in the majority of cases but ’not the same’ answers in the minority.

We have encountered users with an algorithm with 92% reliability and 2% selectivity refuse one with 98% reliability and 0.1% selectivity because it did not give the same answers in a parallel run.

CHAPTER 2

SSA-NAME3 Overview

This chapter includes the following topics:

Product History

The Informatica Corporation software was born in 1986 out of the experience gained by its original architects in building bespoke name matching systems, mostly for government agencies.

This experience had shown that, while the significance of name matching in different systems varied considerably, there had always been the same basic set of concerns:

1. Solving the performance implication that the most frequently occurring names are also those upon which searches are most often performed.
2. The problems of widely based phonetic algorithms creating a response time problem and also a user problem in locating the match from many candidates.
3. That true phonetics is only a subset of the errors in names.

Some of the projects undertaken emphasized the need to quickly achieve a match, if there was one. Others placed their emphasis on proving that there was no match at all.

One project presented the unusual opportunity for empirically developing and modifying an algorithm designed to solve phonetic, orthographic and Anglicization problems in more than 2,000,000 hand written credit records.

During the project development activity some 300,000 computerized matches were compared to manually made matches done by expert searchers. Whenever the searcher found data not found by the system, the algorithm was revised.

Another project involved the re-processing of 25 million records where it was known that at least 99% of the records were in fact pairs of records about the same person. This project also included the need to develop a name search system to handle over 30,000 inserts per day.

This project demanded a performance breakthrough as the design objective was to support a 50,000,000 record on-line database. The successful solution, based on the fact that the project's purpose was to identify the records that were not in pairs, and that this population was smaller than the error rate in the data, required considerable research.

One of the characteristics of SSA-NAME3 is that it is an empirical rather than theoretical solution to the aforementioned problems.

The original company set up to develop and market this technology was known as Search Software America (SSA), giving meaning to the "SSA" in SSA-NAME3. The Search Software America name existed until 2005 when the company name was changed to Identity Systems. In May 2008 Informatica Corporation acquired Identity Systems. Even after the company name changes, the SSA-NAME3 product name was retained because of its brand awareness and usage in hundreds of production systems worldwide.

The early releases of the SSA-NAME3 Algorithms were known as SSA-NAME1. These were superseded by SSA-NAME2 and the latest generation was called SSA-NAME3. At this point, the underlying algorithm design stabilized and the product name stayed as SSA-NAME3.

Since inception, our mission has been to help organizations design optimum dialogues for their specific needs, and overcome the complexities and variation in their data. The experience gained from these diverse projects and those of our customers around the world is incorporated into the ongoing development and maintenance of SSA-NAME3.

In the late 1990’s SSA-NAME3 also became the core-technology inside our two newer products, the Data Clustering Engine (DCE) and the Informatica Identity Resolution (IIR). While SSA-NAME3 continues to be sold and supported as a software development kit, it is now more commonly distributed as the core component of these newer solution based products.

Continuing research & development into the core SSA-NAME3 key-building algorithms, search strategies, match processes and multi-country support continues apace, (a) because of the large installed SSA-NAME3 user base and (b) because the IIR and DCE products are highly dependent on the quality, reliability and selectivity generated by SSA-NAME3.

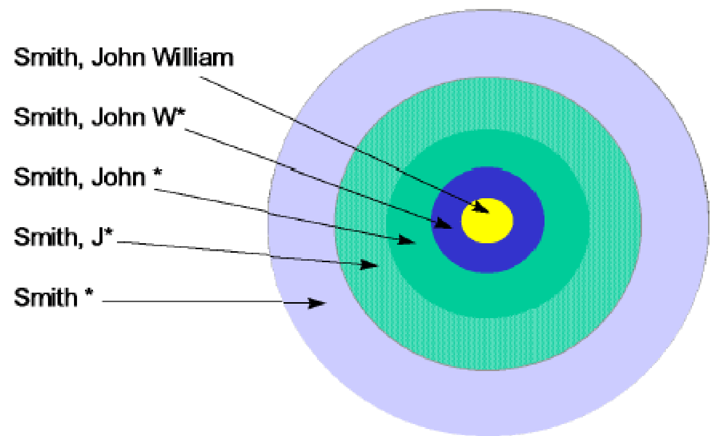
Major Features

SSA-NAME3 provides any computer system with a variety of access paths to data using names or addresses.

Applications dealing with relatively reliable and complete data can use a high performance access path, while applications dealing with less reliable data or with a more critical problem can use more complex access paths. For any name or address, SSA-NAME3 provides a logical access path to the set of records that are likely to include relevant matching records.

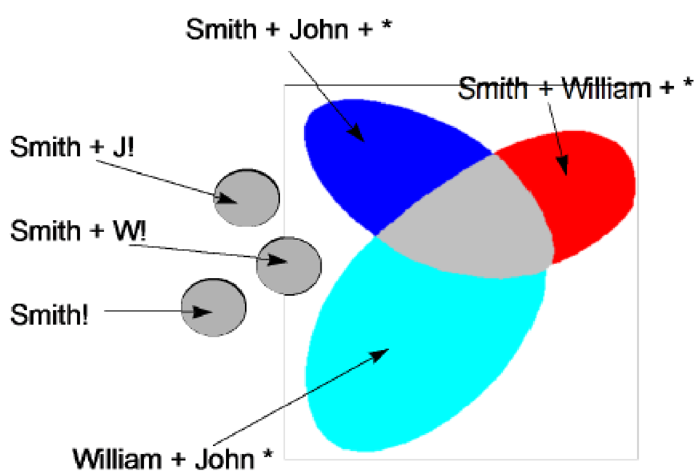
In an application where a match is expected to be found (e.g. customer search), it is possible to present to a user, or process, a set of highly relevant records and then, if necessary, cast a wider net for records that are less likely to be relevant. We call this a ’positive’ search. This approach allows the designer to use techniques that progressively increase the depth of the search. At each particular depth or level it is only necessary to physically access the set of records that match that depth of search. This is significantly different than the approach of accessing a large bucket including all possible candidates and then refining the records using a "scoring" technique to control the depth. For example:

Figure 1: Diagrammatic example of a ’positive’ search * means "and anything else"



In applications where the search or file data is unreliable or incomplete, or where a match is not expected to be found, or where the search is critical, it is possible to invoke different types of search depending on the thoroughness required or the extent of the unreliability. These are collectively called a 'negative' search. For example, this is one type of negative search:

Figure 2: Diagrammatic example of a 'negative' search



The major features of SSA-NAME3 are summarized below:

SSA-NAME3:

- addresses the problems of error, variation and performance for systems which need to use name or address data for searching or matching.
- can be applied to any types of names including Person, Company and Business names, Account and Compound names, Addresses, Product names, Song Titles, Book Titles and any other short descriptive text.
- is a set of Callable software tools which provide key generation, search strategy, data matching and other services to applications.
- can be used by applications in either on-line or batch mode.
- allows applications to achieve very high-performance by using the full-name in a search where a match is expected to be found (the typical customer search).
- provides different search strategies for different application and user needs (e.g. 'positive' and 'negative' search strategies).

- is extensively customizable and automatically tailors its Algorithms to the users actual population of names.
- performs with large files of names (100,000 to 100,000,000 or greater).
- once installed in a particular environment, can be used by all systems in that environment.
- is hardware independent.
- can be used with any programming language that supports a Call to an external routine.
- will work with any access method or DBMS that provides a key sequential access path on either its primary or secondary keys.

SSA-NAME3 will work equally well in the following environments:

Environment	As
Index	Sequential a primary key or secondary if supported.
VSAM	a key sequential set key or as secondary key if logical sequential supported.
Hierarchical	for VSAM
Network	an access key
Inverted	List as long as the key is inverted
Relational	an index, preferably
Object	an index, preferably

The SSA-NAME3 Service Group Environments

Once installed, SSA-NAME3 has a Customization Environment and a Development/Run-time Environment.

The SSA-NAME3 Customization Environment

The customization environment is comprised of five major components:

- The Country specific Definition files, which contain the default rules for how the name search & matching software operate on data for different populations and different countries.
- The Customization and Generation modules, which allow the default Definition files to be customized for an organization’s data and applications, and then re-built.
- The Testing module, which allows a callable SSA-NAME3 module to be tested in a stand-alone manner.
- The Visual Workshop, which allows the definition files to be customized, generated and tested in a visual environment.

The customization environment is on Windows for all platforms except MVS, where native customization is available. (However, MVS users are encouraged to use Windows as there are certain benefits).

The Country Specific Definition Files are supplied pre-loaded with the basic definitions suitable for a given country. These are called "Fast-start" definition files and are usually used as the basis for any customization work.

There is a minimum amount of customization that must be done to these files and tables before SSANAME3 can be used. See Tips on Customizing an Algorithm section in the *DEFINITION and CUSTOMIZATION GUIDE FOR SSA-NAME3 SERVICE GROUPS* guide for more details.

In summary, the customizable components within the Country Definition files are:

Algorithms

For a given population of names or addresses, an Algorithm manages the way a name or address within that population is transformed prior to key-building or matching. This is done by selecting the Character-set tables, Edit-lists, Frequency tables, Scaler Frequency tables (optional), Word Stabilization routines and other processes it will use. An Algorithm also controls the type, style and number of keys generated for a name or address.

Character-set tables

These tables control the way individual characters are treated. Examples of treatment are casing, dropping of accents and deletion of non alpha-numeric characters.

Edit-lists

An Edit-list controls the way common words or phrases within a name or address population are transformed or marked. Examples of transformation are replacements for nick-names, abbreviations, synonyms & phrases; deletions for noise, delete or stop words; concatenation or splitting for prefixes and suffixes. Examples of marking are, finding the major word and, classification of data.

Frequency tables

These tables are automatically built from the user's data and control the way common words within the name or address population are compressed into keys.

Scaler Frequency tables

These tables are optionally built from either the user's data or the user's SSA-NAME3 keys. They enhance the Scale value returned in the `NAMESET` search ranges.

Search Strategies

A Search Strategy controls the way name or address key index is searched and varies according to the needs of the application. A customer search is one search strategy, a pre-insert search is another and a fraud search may be quite different again.

Matching Schemes

A Matching Scheme controls the way two records, usually a search and a file record, are matched. Matching occurs after Searching and can be performed on names, addresses, dates, telephone numbers, codes and other data strings. The goal of a Matching Scheme is to mimic the choice process of a user. The results of Matching are used for filtering, matching or ranking records.

After these Definition files and tables have been customized, they are then generated. If the development environment is on a different platform to the customization environment, the Service Group Data File must be transferred to the target platform. The Data File is plain ASCII text which should be considered when transferring.

A final `Gendll` step is performed which creates the dll/shared object that can be called from the program.

The SSA-NAME3 Development/Run-time Environment

The development environment is on the platform where the applications which will use SSA-NAME3 are developed. This is often referred to as the target platform as it is the target for the customization work. The development environment is comprised of:

- The Core modules, which represent the non-modifiable kernel of the software and without which the other components would not work.

- The Testing module, which allow a callable SSA-NAME3 module to be tested in a stand-alon manner.
- The API's, which allow user applications to call SSA-NAME3 Services.

Most major platforms are catered for as targets.

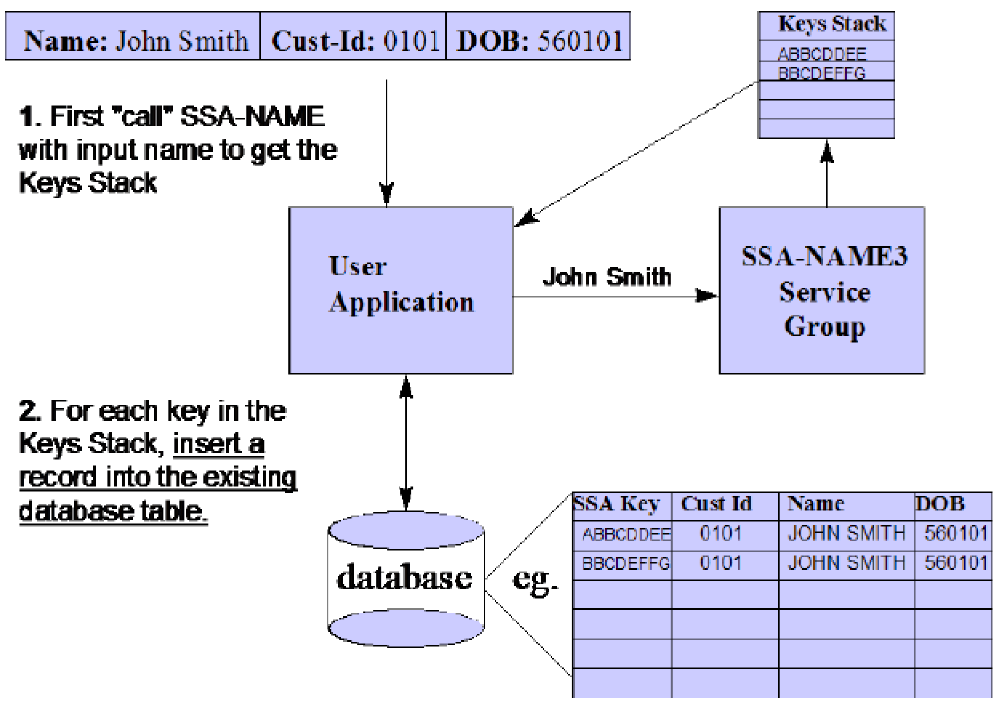
The run-time environment is on the platform where the applications, which will use SSA-NAME3, are executed. The run-time environment is comprised of:

- The Callable SSA-NAME3 module, which has been prepared in the development environment. No other SSA-NAME3 modules are required in the run-time environment.

User applications communicate with the Callable SSA-NAME3 Service Group via documented API conventions. For on-line applications, it is normal for one copy of the SSA-NAME3 Service Group to be shared by many user transactions. For batch applications, it is more common for the Service Group to be statically linked to the application.

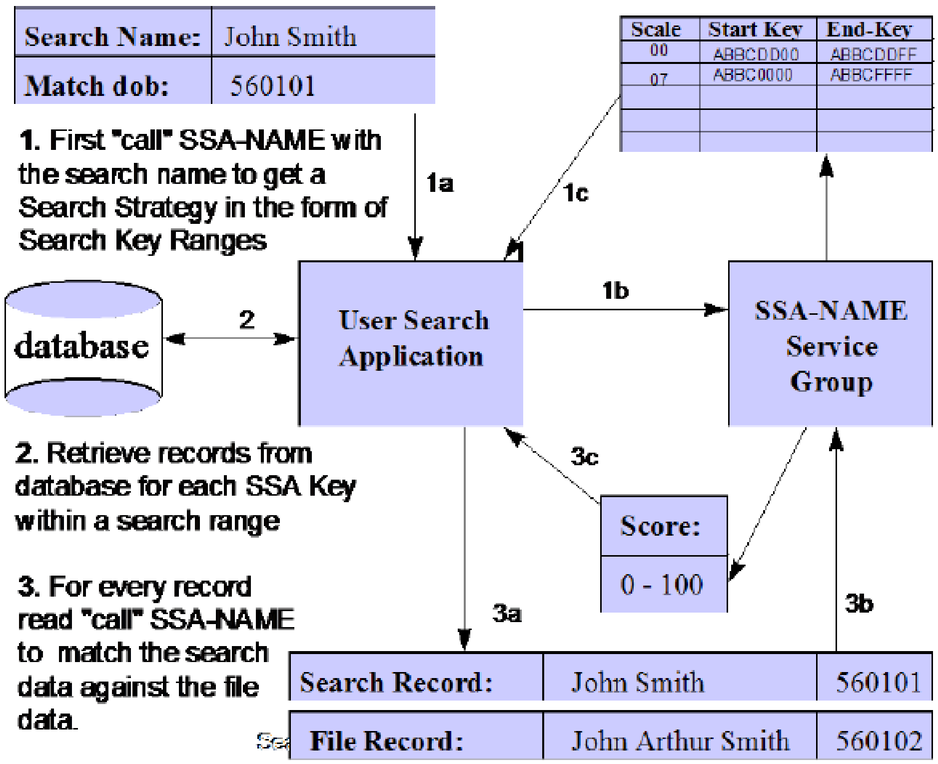
User applications use the SSA-NAME3 Service Group for three main reasons:

1. To build keys for the name or address population to be searched. The User application calls the SSA-NAME3 Service Group passing it a name or address and indicating which Algorithm it wishes to use. The Service Group passes back one or more keys ("`Keys Stack`") which are to be stored by the application in the user's database. These keys are either 5-bytes binary or 8-bytes character depending on the DBMS being used. Applications making use of this service are commonly the initial key-load program, and name or address maintenance programs. For example,



2. To build search strategies for a search name or address. The User search application calls the SSA-NAME3 Service Group passing it the name or address and indicating which Algorithm and Search Strategy it wishes to use. The Service Group passes back an array of key ranges which define the sets of records which are candidates for this name or address search. The User application then does a key sequential access of its SSA-NAME3 key index for the key range or key ranges specified. It returns the name or address and other data from the database which is then used for matching.

3. To match two records, a search record and a file record. After reading a candidate record in the search, the User application calls the SSA-NAME3 Service Group indicating which Matching Scheme to use and passing it the search name or address, and any other identity data entered, and a file name or address, and the identity data retrieved for the file record. The Service Group compares the two records and passes back a Score between 1 and 100. The User Application may then use the Score to match, filter or rank the records. For example,



SSA-NAME3 Services

The following paragraphs give an overview of the various SSA-NAME3 Services, what function they perform and in some cases, their basic operation.

Key and Search Strategy Building

The Service used to perform Key and Search Strategy building is called `NAMESET`.

When a name or address is passed to NAMESET, the Algorithm to use is also identified by the Calling program. This Algorithm will cause the passed name or address to be processed in four phases:

Cleaning

This routine uses internal routines and the Algorithm’s Character-set tables to edit the passed string. Common actions include removing special characters and replacing lower-case with upper-case. The Cleaning routine itself is not customizable.

Formatting

This routine uses internal routines and the Algorithm's Edit-list to edit the passed string into separate words, removing noise, delete or stop words, replacing selected words, concatenating prefix words and other such actions. The Formatting routine itself is not customizable, however, an exit is supplied which may be customized. A working example of this exit is supplied for English style nick-name processing.

Word Stabilization

This routine stabilizes the Cleaned, Formatted words using country specific rules to cater for phonetic and orthographic error. For example de-duping double characters is one common rule. The Stabilization routine is not generally available for customization.

Key and Search Strategy Generation

Builds the keys to be stored in the database, and the key ranges to be used at search time. The type, style and number of keys is controlled by user customizable Algorithm options. The type of Search Strategy is controlled either by Algorithm options or by the Calling program.

Matching

The Service used to perform Matching is called `MATCH`.

When two records are passed to MATCH for matching, the name of the Matching Scheme to use is also identified by the Calling program. This Matching Scheme has been pre-defined as part of the customization process and contains the view of the passed records, which Matching Methods are to be used for each field in the record view, and what weights to assign to each field. There are different methods for matching names, address, dates, codes and other strings.

The Methods used for matching names or addresses indicate which Algorithm to use as part of the Matching process. Before two names or two addresses are compared they are first processed through the Cleaning & Formatting processes of the Algorithm as described in the previous section. The Matching Method then uses its own internal processes, which respond to customizable options, to compare the two Cleaned and Formatted strings. The Method may also resort to using the Word Stabilization routine as part of the matching process.

The Matching Methods used for dates, codes and other strings do not use Algorithms. These Methods use their own internal processes, which respond to customizable options, to compare the two fields.

Other SSA-NAME3 Services

The overview of SSA-NAME3 so far has concentrated on its 'core' services, that is name and address searching, and matching.

SSA-NAME3 also provides other services. These are:

Word-key

Builds a compact key for a user supplied word (as opposed to a 'name'). For example, can be used to build fuzzy keys for text indexing.

Major-word-key

Builds a compact key from the one word in a name which is most suitable for indexing.

Trace

Traces the actions taken by the Formatting process and makes these available back to the Caller. For example, can be used to identify the components of a name or address for such business needs as personalization of marketing letters, or geo-coding of addresses.

Support Routines

Access is provided to the Cleaning, Formatting and Word Stabilization routines directly. For example, names or addresses could be cleaned before they are stored in the database or before printing. The Formatting routine could be called directly to 'tokenize' a name or address for some other purpose.

Test-bed

A stand-alone test facility which can invoke other SSA-NAME3 services and show the results of the Call.

Browse

Reports internal data and is generally used for debugging purposes.

Info

An API for retrieving information from internal SSA-NAME3 tables.

Debug

Assists in developing the rules for record matching by allowing user programs to dynamically alter Matching rules.

SSA-NAME3 CJK-SUPPORT

SSA-NAME3 CJK-SUPPORT is the name given to a separately licensable product which deals with the special nature of Chinese, Japanese and Korean data. Its features include the ability to recognize and encode double-byte names and addresses, handle special representations of Chinese numbers, and allow Edit-lists to be maintained using the CJK language characters.

CHAPTER 3

Introduction for Application Programmers

This chapter includes the following topics:

Overview

The purpose of this section is to describe SSA-NAME3 Service Groups and its usage at a high-level and from the analyst programmer's point of view. The example application being described is that of an on-line customer name search, but could be interpreted for any type of search.

What is the Purpose of SSA-NAME3?

The purpose of using SSA-NAME3 is to find potential matches on a database for exact, incomplete or inaccurate names upon which we are `SEARCHING`. For example, if `Mary Evans Jones` is stored on the customer database, we want to be able to retrieve her record even if we search on `Mary Evans`, `Maria Jones`, `M. Evens`, or `M E Jones`, to name a few.

How do we Achieve this?

This is accomplished by storing specially formatted 5-byte KEYS (8-byte keys in some implementations) for each customer. Then, for a search name, we locate all keys within a RANGE that is likely to include this search name or a variation of it.

In fact, we usually store MORE THAN ONE KEY – `alternate` keys – for each customer. For example, for `Mary Evans Jones`, we would include a key not only with `Jones` as its major component, but also one with `Evans` as a major word. Then, if an operator searches on the name `Mary Evans`, the search key range produced for `Mary Evans` will INCLUDE the Mary Evans key we had created for the customer `Mary Evans Jones`.

Where are the SSA-NAME3 Keys Stored and in What Format?

These keys are stored on `ANY TYPE OF INDEXED FILE` and in `ANY RECORD LAYOUT`. The only requirement is that they are stored somewhere where you can `FIND` or `LOCATE` a given key and then `READ NEXT` until the end of a range of keys is reached.

If you are accustomed to using a relational database, you could define a simple relational table containing the SSA-NAME3 key, the customer name from which the SSA-NAME3 key was created, and the unique key for this customer record - for example, customer number.

It will become clearer as to why the format of the indexed file is unimportant to SSA-NAME3. But how do you decide, then, exactly how to lay out this file? Here is an EXAMPLE of a typical record layout for what we will refer to as the `SSA-NAME3 KEY FILE`:

SSA-NAME3 Key	Customer Name	Other Identifying Data	Customer No
@#$%^	John Smith	...	A12345
(&%$#	John Smith	...	A12345
(*#$%	Geoff Brown	...	B23671
)(&&%	Geoff Brown	...	B23671

With this type of format, there will be MULTIPLE SSA-NAME3 key records for most customer names - one for each alternate key for this customer. Moreover, the customer name and other identifying data will be `REPEATED` for each SSA-NAME3 key record created. You could also simply store the SSA-NAME3 key and the Customer No, however, the above method will provide the best performance when doing name searches and is well worth the extra disk space required. For more information on the subject of database design and performance, see the *How Does a User Determine Which is the Correct Record?* section below, or the *Database Design Notes* section in the *APPLICATION REFERENCE FOR SSA-NAME3 SERVICE GROUPS* manual.

How are the SSA-NAME3 Keys Created?

You first create a KEY LOAD application program, using COBOL, Visual Basic or whatever development language you are accustomed to using. The program should be linked or have access to the Callable SSA-NAME3 routine using the conventional methods for your environment.

The key load program will `CALL SSA-NAME3` with each customer name as a parameter passed to it; all SSA-NAME3 keys (i.e., alternate keys) for that customer name will be returned in an array. You then WRITE these keys to the "SSA-NAME3 Key" database table (or other indexed file), along with whatever other matching information you have decided to store, and in whatever format you decided upon. As you read on to see more on how SSA-NAME3 is used, you will better be able to decide just what information you want stored on the SSA-NAME3 Key File.

The following pseudo-code for the key load program (with comments) will help clarify how this works:

```
READ NEXT CUSTOMER DATA BASE RECORD UNTIL EOF;
     MOVE CUSTOMER-NAME TO SSA-NAME3-NAME-IN;
     MOVE 'BLDKEY' TO SSA-NAME3-FUNCTION;
     CALL SSA-NAME3 USING SSA-NAME3-FUNCTION,
                          SSA-NAME3-NAME-IN;
```

Note: The above Call does not show the exact API requirements – these are described in the *APPLICATION REFERENCE FOR SSA-NAME3 SERVICE GROUPS* manual.

You now have returned to you a `KEYS-STACK` array with the different alternate keys for this customer. For each key in the array, create a record in the SSA-NAME3 Key File.

```
DO FOR (ALL KEYS IN KEYS-STACK FOR THIS CUSTOMER-NAME);
   MOVE KEY(N) TO SSA-NAME3-KEY-FILE.SSA-NAME3-KEY;
   MOVE CUSTOMER-NAME TO SSA-NAME3-KEY-FILE.CUST-NAME;
```

Optionally, move other identifying information. This will be addressed at a later stage.

```
        MOVE CUSTOMER-ID TO SSA-NAME3-KEY-FILE.CUST-ID;
        WRITE SSA-NAME3-KEY-FILE RECORD;
    END DO;
```

Note that the only interface the key load application program had with SSA-NAME3 was to send it a name and some other parameters. SSA-NAME3 didn't care what kind of database this name came from. Also, all that SSA-NAME3 returned essentially was an array of keys. It didn't care where those keys were stored they could be written to a DB2 or Oracle table, a VAX RMS file, a VSAM KSDS, or any other type of indexed data base or file system.

What we have at the conclusion of the load program is a file indexed on SSA-NAME3 key.

How are the SSA-NAME3 Keys Used for Searching?

The initial loading of the SSA-NAME3 keys is a one-time operation. Of course, in a production environment, whenever new customers are added to the database, or names changed, new SSA-NAME3 keys must be generated for these - this is normally done as part of the maintenance transactions.

Once the SSA-NAME3 Key file has been created you can use it from within an application program that performs the `SEARCH` function. Let's assume that the search program is an on-line program, although the concepts apply as well to batch programs.

The following pseudo-code will help describe how this search program works:

```
    GET CUST-SEARCH-NAME FROM SCREEN;
    MOVE CUST-SEARCH-NAME TO SSA-NAME3-NAME-IN;
    MOVE 'POSSRCH' TO SSA-NAME3-FUNCTION;
    CALL SSA-NAME3 USING SSA-NAME3-FUNCTION,
                         SSA-NAME3-NAME-IN;
```

Note: The above Call does not show the exact API requirements – these are described in the *APPLICATION REFERENCE FOR SSA-NAME3 SERVICE GROUPS* manual.

The example `POSSRCH` function will return an array of `SEARCH KEY RANGES`. It is these `SEARCH` ranges that we are now interested in, rather than the storage keys.

The positive search ranges in this example go from narrow (only the closest matches to search record returned) to wider than would ever actually be used in practice. The choice of which search range(s) to use is a discussion in itself, and should be discussed more thoroughly with an SSA technical support person. Let's assume we're using the narrowest range (the first one in the search table array).

To continue with the code:

```
    FIND SSA-NAME3-KEY-FILE.SSA-NAME3-KEY >= START-SEARCH-KEY
    DO WHILE (SSA-NAME3-KEY-FILE.SSA-NAME3-KEY <= END-SEARCH-KEY);
       MOVE SSA-NAME3-KEY-FILE.CUST-NAME TO DISPLAY SCREEN;
       MOVE SSA-NAME3-KEY-FILE.CUST-ID TO DISPLAY SCREEN;
       READ NEXT SSA-NAME3-KEY-FILE RECORD;
    END DO;
```

What are the Results of the Search?

After performing the above search you will have on the screen a list of customer names whose SSANAME3 keys fall within the search range just referenced.

Now your application program may provide a facility for the operator to select one of these names and retrieve the actual `CUSTOMER DATA BASE` record to which a certain `SSA-NAME3 KEY FILE` record points.

For example, if the customer search name was `Mary Evans Jones`, the list displayed may contain:

```
            Name        Customer No
```

```
Mary Jones  1236
Mary Evans  9812
M.Jones     7745
M.E.Jones   2176
M.Evans     3737
Evan Marie  9508
```

How Does a User Determine Which Is the Correct Record?

Displaying the name alone will most likely not be enough to tell the user which customer is the correct one other identifying information will be required. The operator could go and display the customer master record for each name, but this is not the most efficient way.

This brings us back to one of the first questions - what data do we store on the SSA-NAME3 Key File? In the key load example described above we stored the customer name and customer id along with each SSA-NAME3 key created. This customer name is what is displayed on the screen list of names.

For efficiency, we also recommend storing additional identifying information with each key on the SSA-NAME3 key file. For example, you may want to store date of birth and street address if this what the users would commonly use to confirm a match. Now, when you display the records retrieved from the SSA-NAME3 Key File for a given search range, you will see not only the customer name but this additional information as well making the choice easier.

The reason for storing the other identifying data in the same table as the SSA-NAME3 name keys is that the application or DBMS does then not need to join multiple tables to retrieve that data, and this leads to a saving in I/O. Equally as important for performance is that the table must be ordered physically by the SSA-NAME3 name key this means that records with similar names are stored physically close to each other in the table and the DBMS will need less I/O to retrieve those records.

How to Use SSA-NAME3 Matching to Make the Choice Easier?

If your database contains a large number of records, and the search name was for example, `John Smith` or some very common name in your population, the number of records and screens to be displayed could be significant. In this case, simply showing the other identifying data on the screen is not enough to make the choice easy for the user because the correct record could be on the last screen.

This is where SSA-NAME3 Matching comes in. If the user also enters some of the other identifying data as part of his/her search criteria, this data, along with the name, can be used to get a 'Score' for each candidate record found in the search. This Score is a value between 1 and 100 and can be used in a number of ways:

- When each candidate record is retrieved you can pass it to SSA-NAME3 for Matching and use the Score to `ELIMINATE` the candidate from the list to be displayed.
- After all of the candidate records have been read and matched, you can sort the candidate list in descending order by the Score before they are displayed. This way, the most likely candidates will appear at the top of the first screen.

For example, if your search criteria was:

```
Name:     Mary Evans Jones
Address:  1445 East Putnam Avenue
```

Using Matching, the search results could be returned in the following order:

```
Name          Address              Cust Id
M E Jones     1445 E. Putnam Ave.  2176
Mary Jones    14 Peter Street      1236
M. Evans      107 Putnam Drive     3737
```

The following updated pseudo code shows how Matching fits into the Search program:

```
GET CUST-SEARCH-NAME,SEARCH-ADDRESS FROM SCREEN;
MOVE CUST-SEARCH-NAME TO SSA-NAME3-NAME-IN;
MOVE 'POSSRCH' TO SSA-NAME3-FUNCTION;
CALL SSA-NAME3 USING SSA-NAME3-FUNCTION,
                     SSA-NAME3-NAME-IN;
MOVE CUST-SEARCH-NAME TO SEARCH-RECORD.NAME;
MOVE CUST-SEARCH-ADDRESS TO SEARCH-RECORD.ADDRESS;
MOVE 'SCH01' TO SSA-NAME3-MATCH-SCHEME;

FIND SSA-NAME3-KEY-FILE.SSA-NAME3-KEY >= START-SEARCH-KEY
DO WHILE (SSA-NAME3-KEY-FILE.SSA-NAME3-KEY <= END-SEARCH-KEY);
   MOVE SSA-NAME3-KEY-FILE.CUST-NAME TO FILE-RECORD.NAME;
   MOVE SSA-NAME3-KEY-FILE.ADDRESS TO FILE-RECORD.ADDRESS;
   CALL SSA-NAME3 USING SSA-NAME3-MATCH-SCHEME,
                        SSA-NAME3-SEARCH-RECORD,
                        SSA-NAME3-FILE-RECORD;
```

Note: The above Call does not show the exact API requirements these are described in the *APPLICATION REFERENCE FOR SSA-NAME3 SERVICE GROUPS* manual.

```
   IF SSA-NAME3-SCORE > '050'
      MOVE SSA-NAME3-KEY-FILE.CUST-NAME TO PROGRAM ARRAY;
      MOVE SSA-NAME3-KEY-FILE.CUST-ID TO PROGRAM ARRAY;
      MOVE SSA-NAME3-SCORE TO PROGRAM-ARRAY;
   END-IF
   READ NEXT SSA-NAME3-KEY-FILE RECORD;
END DO;

SORT PROGRAM ARRAY DESCENDING BY SSA-NAME3-SCORE;
DISPLAY PROGRAM ARRAY TO SCREEN;
```

APPENDIX A

Glossary

This section provides glossary of terms.

Account Name

A name field, often referring to account details, which implicitly refers to more than one simple name, for example,

```
JOHN AND MARY SMITH
```

See also section *Compound Name*.

Algorithm

A combination of SSA-NAME3 routines that have been generated for a specific Population (example, person names, company names, street names). The Algorithm is accessed through Calls to Services which are linked to it.

Alternate Keys

The Name-key(s) built from different word orders in a name. These can be either Positive or Negative Keys.

Authorization

The process of collecting the signatures and details of the routines linked to an Algorithm. These signatures are checked at run-time to ensure that no 'rogue' modules have been linked.

Authorized Algorithm

The Algorithm for a Population that is currently available for use by application programs.

Bad key

SSA-NAME3 never returns an unusable Name-key. If for some reason a key could not be generated a special "bad key" is returned. This bad key has a value of 800000HEX, when using 5-byte binary keys, and `K$$$$$$$` when using 8-byte character keys. It can be used to group bad names so that they can be found.

An example would be the name, `THE LIMITED`. If both words in this name are removed by the Edit-list the result is that there is no name to build the key from. In this case the bad key is returned.

Candidates

The set of records returned from a **Name** search. For optimum quality these candidates should be passed to the **Matching** Service for further qualification before being displayed or otherwise used in a search process.

Cascade

The name given to the Search-table structure built for the most common type of Positive Search strategy. The Search-table starts with the narrowest Name-key range, which could contain the 'searched-for' record, and continues with progressively widening ranges.

Cleaning

The process of applying character set conversion rules to a **Name** with the intention of cleaning and/or converting unwanted characters.

Code-character

Any character marked as a code in character-set table 2. This is normally only the digits 0 - 9.

Code-word

Any token with one of the following attributes:

1. 2 or more code characters,
2. an initial that is a code character,
3. 1 or 2 characters in length and either preceded or followed by a Code-word.

Common Name

A **Name** that occurred often enough in the sample user data provided to the section Frequency Table generation to be considered common.

Compound Name

A **Name** field which explicitly refers to more than one simple name, for example,

```
JOHN SMITH AND GEORGE BROWN
```

See also section Account Name.

Delimiter

Any character defined as a delimiter in character-set table 4.

Edit-list

A table of user controlled words & phrases that undergo special processing in **Name-key** building and **Matching**, example, noise words, personal titles, prefixes & suffixes, nicknames, common abbreviations, phrase replacements and **Compound Name** markers.

Edit-rule

A line in an Edit-list, for example the line

```
RR ROB     >ROBERT <
```

is an Edit-rule that says "Replace `ROB` with `ROBERT`".

Fast-start

A collection of sample definition modules for a specific country used to create a first-cut Service Group. The Fast-start definition files are used when first installing SSA-NAME3 as a quick way to test the Installation process and environment. They are later used as the basis for further customization work to make SSA-NAME3 achieve the objectives of the application and end-users.

Filtering

The process of **Matching Candidates** to reduce the number of records shown to the user.

Formatting

The process of applying Edit-rules to words, phrases and sub-strings.

Frequency Table

A table generated from an organization's **Name** data holding the most frequently used words that have not been deleted or skipped as a result of **Edit-list** processing.

Generation

The process of creating compilable source code modules from Definition files, either on a Windows computer or an MVS system.

Initial

A single character word or the first character of a word.

I+n

Nomenclature for "Initial plus n consonants".

Key Generation

The process of building one or more **Name-keys** from the Cleaned, Formatted and Stabilized **Tokens** of a **Name**.

Major-word

The word in a Name identified as being the most significant. It is used as the primary part of the **Preferred Key**, as the primary part of the Search key ranges of a positive search, and for weighting in **Matching Schemes**. See also section *Minor-word*.

Major word-key

A 3-byte or 6-byte key generated from the **Major-word** of a **Name**.

Matching

The process of determining the probability that a search identity and a File entry are the same identity.

Matching Method

The way in which section Matching matches two data items of the same data type. There are methods for names, addresses, dates, strings and codes. (See section *Method*).

Matching Scheme

A definition of the structure of the data items to be matched and the Matching Methods and options to be used.

Method

A routine used for Matching two data items. There are Matching Methods for names, dates, codes and strings. For example, in the following Matching Definition line,

```
DEFINE METHOD=METHOD1,EP=N3SCL,ALGORITHM=PERSON
```

The method is `N3SCL` (the name matching method).

Minor-word

Any token in a Name which is not the Major-word and is not a word deleted by an Editrule.

Name

The name of a person, company, business or organization; an address; a product title, song title or book title; any short description. A name consists of a number of words, each with a limit of 24 characters.

Name-key

A compressed five-byte binary or eight-byte character key built from a Name using the NAMESET Service.

Negative Keys

The Name-key(s) built using each non-delete Token (word) in a Name in combination with every other word in the name. See the *Positive Keys* section.

Negative Search Strategy

A method of building a Search-table for a search application whose normal requirement is to prove that a name does not already exists on a database.

Population

Population refers to a class or group of names that requires its own SSA-NAME3 Algorithm. Typical examples of "populations" are: customers, street lines from addresses, song titles, file titles etc.

Positive Keys

The **Name-key**(s) built using each non-delete **Token** (word) in a Name followed by the other words in a set order. See the *Negative Keys* section.

Positive Search Strategy

A method of building a **Search-table** for a search application whose normal requirement is to find a name that already exists on a name database.

Preferred Key

The **Name-key** built from the **Major-word** followed by the **Minor-words** in a **Name**.

Probe

A very narrow search range.

Ranking

The process of sorting the Matched section Candidates to show the records to the user in descending order of their likeness to the search identity.

Reliability

The probability that a section Search Strategy will find a name if one exists that should be considered as a match to the search **Name**.

Response Code

A unique number indicating the success or otherwise of the Service just called.

Scaler Frequency Table

Atable generated from an organization's **Name** data or an organization's SSANAME3 keys holding the most frequently occurring **Name-keys**. Generation of this table is optional, and it is used to enhance the scale value returned in the NAMESET search ranges. See the *Search Scale* section.

Score

A value between 1 and 100 returned by the **Matching Service** to an application. This defines the level of confidence that two candidate records match.

Search Contents

A term used to describe the number of **Tokens** (words and **Initials**) used in a particular search range.

Search Depth

A term used to describe the width of a **Name-key** search range or its **Selectivity**.

Search Dialogue

The method by which a search application processes a Search-table and displays the Candidates to the user.

Search Scale

An estimate of the number of Candidates that would be returned using a particular search range.

Search Strategy

The method by which a Search-table is built to achieve the optimum search results for the particular application requirement (e.g. **Positive Search Strategy** or **Negative Search Strategy**).

Search-table

A table of **Name-key** ranges used by a search application to access a **Name** database on a **Name-key** index. This is the physical implementation of a section Search Strategy.

Selectivity

The percentage of records that are accessed to satisfy the average search.

Service

An SSA-NAME3 function that has been defined and generated for some specific user required purpose and for a specific **Population**; e.g. building **Name-keys** and **Search-tables**; **Matching** two records according to a specific set of rules.

Service Group

A collection of SSA-NAME3 Services that are grouped as one program under one name. In Call statements you call a Service Group name requesting a Service, passing parameters according to the service rules.

Service Group Data File

An ASCII text file containing the Service Group "ruleset". It is invoked at runtime by the shared object or dll code.

Service Name

The name used when referring to a Service, e.g. NAMESETP is a name typically used to define a service of type NAMESET for the PERSON Algorithm.

Service Type

The type of a **Service**, this defines its functionality whereas the **Service Group Data File** An ASCII text file containing the Service Group "ruleset". It is invoked at runtime by the shared object or dll code. Service Name is simply a handle to refer to the Service with, e.g. the Service NAMESETP is of type NAMESET.

Skip-word

Any **Token** in a **Name** which is defined by an Edit-rule not to take part in **Name-key** building.

SSA-NAME3

The latest version of the SSA-NAME3 **Algorithms**.

Stabilization (Word Stabilization)

The process of applying phonetic and orthographic transformation rules to Name Tokens to stabilize the error and variation.

Suspect Code-word

A word of 3 or more characters that includes 1 **Code-character**. See the *Code-word* section.

Token

The individual word components of a section Name after **Cleaning** are called Tokens.

Target Platform

The combination of hardware and software that will execute an SSA-NAME3 Algorithm.

Test-bed

An SSA-NAME3 utility which enables quick and reliable testing of Algorithms and Matching Schemes, either interactively or in batch mode. For Microsoft Windows, a Windows based Test-bed can also be used.

Vowel

Any character defined as a vowel in character-set table 4.

Word-key

A 3-byte or 6-byte key generated from any single **Token** (word).

Word-type

A one-character code assigned to a **Token** by the **Formatting** routine. Possible values are:

B - a Suspect Code-word.

C - a Code-word.

D - a deleted word (used only by the TRACE Service)

I - an Initial.

M - the Major-word.

N - the Major-word if it is a Code-word.

S - a Skip-word.

T - a Skip-word if it is a Code-word.

Y - any other word

INDEX