Web Service API for ENS (IBM InfoSphere Global Name Management Enterprise Name Search) Version 6.0

Prepared by: Richard Strangfeld

Edition

This edition applies to Version 6.0 IBM InfoSphere Global Name Management (product number 5724-Q20) and to all subsequent releases and modifications until otherwise indicated in new editions.

1 Table of Contents

1 Introduction

IBM InfoSphere Global Name Management Enterprise Name Search, or ENS, is software that allows sophisticated name searching based on Global Name Management NameWorks to be done at the enterprise level. It uses proven IBM middleware and database components to:

- Perform efficient GNM NameWorks-based searches in large lists of names.
- Distribute those names across multiple servers to allow larger lists, higher performance, and reliability through redundancy.
- Make it easy to manage those name lists and servers.

This document discusses the web service API exposed by ENS for managing and searching in name lists. It assumes that someone has already installed and set up an ENS system, as described in a separate document, "Setting Up and Managing Enterprise Name Search".

Another document, "Searching for Names with Enterprise Name Search", describes searching for names using the web-based search GUI client provided with ENS. That client program uses the search web services described in this document.

1.1 Overview – ENS System Architecture

An ENS system consists of a number of servers with "searcher" components. Each searcher holds an instance of NameWorks with some collection of names. Working together, these searchers provide searching in a potentially very large combined list of names, and can support numerous users searching at one time. Another server-side component called a "dispatcher" coordinates the work of the searchers.

The ENS user who wants to search for names does so in a search client program. This can be the browser-based search client included in ENS, or some custom search program specific to the user's organization.

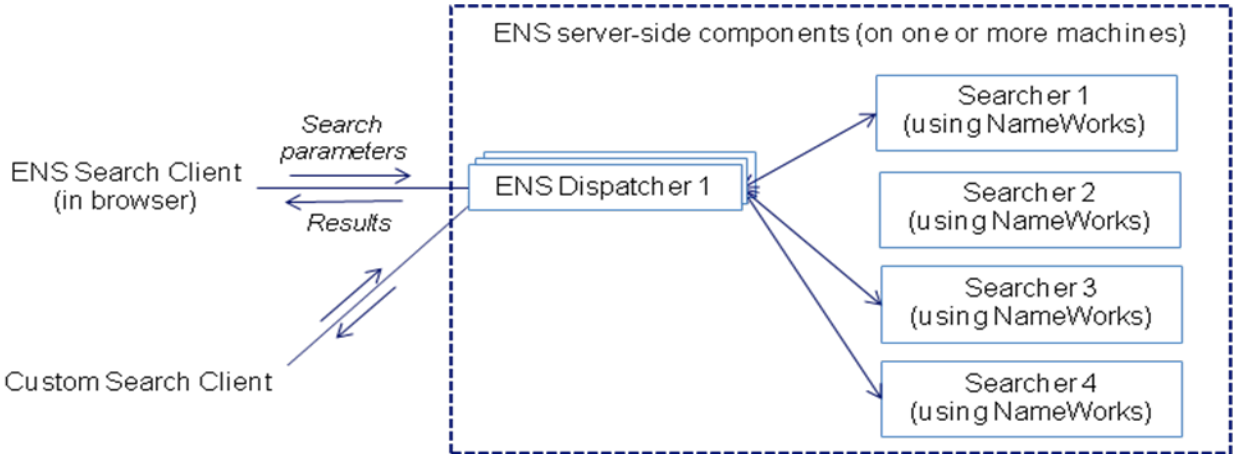

Figure 1 - Simplified client view of ENS Architecture

In either case, the client program makes web service calls to perform searches. At the back end, a dispatcher component handles that call and delegates the request to some set of searchers, then collects their responses and returns results to the client.

On the server side, the ENS system may consist of a single dispatcher and searcher on one machine, or of numerous dispatchers and searchers on many machines. This can vary widely based on the requirements for scale (number of names and number of clients) and performance. Servers and components can be added and removed as needed, even in a running system.

From the client's point of view, it looks the same regardless. The fact that the server side has one machine or many is not important, or even noticeable, to the client. The client just makes search requests to a dispatcher and gets name results.

1.2 Loading Names into ENS

Names may be added to ENS in two different ways: with the NameLoader program, or with the `addName` web service.

The `addName` web service is described in section 3.1 of this document. NameLoader is described in the separate "Setting Up and Managing Enterprise Name Search" document.

1.3 ENS Web Services and Authentication

The ENS web services require authentication in the HTTP request, provided via either a basic authentication header or a session cookie from a logged-in session. See sections 4.2 and 4.3 for more information.

2 Methods Used for Searching

This section lists the search-related methods available in the ENS web service. For all of these methods, the HTTP request must include either a preemptive basic authentication header as in section 4.2, or an ensSecurityCookie for a session established by a login request as described in section 4.3.1.

The user thus identified must have privileges as follows:

- For the `search` method, the user must have search privileges on the specified name lists (or on at least one name list, when no name list is specified in the request.
- For the other methods in this section, the user must have search privileges on at least one name list.

For a user, having search permissions on a name list may mean being a member of a specific numbered searcher or manager group associated with that list, or being a member of the more general searcher_all_lists, manager_all_lists, or admin group. Assigning users to groups is described in the separate "Setting Up and Managing Enterprise Name Search" document.

2.1 Get name lists

This method gets the name lists available for searching by the current user. It takes no inputs. It returns an array of objects, each having a name and a description.

This method takes no inputs.

2.1.1 JSON/REST

Method: GET
URL: <host>:<port>/ws/dispatcher/api/rest/search/namelists
Inputs: None.
Output: The result is a JSON object with a single "nameLists" field. Its value is an array of entries, each having a "nameListCode" string and a "description" string.

Sample output JSON:

```
{
   "nameLists":[
      {
         "nameListCode":"PASSENGERS",
         "description":"Every passenger who ever flew"
      },
      {
         "nameListCode":"BAD_GUYS",
         "description":"All the bad guys"
      },
      {
         "nameListCode":"NO_FLY",
         "description":"The no-fly list"
```

```
      }
   ]
}
```

2.1.2 SOAP

Port Search
Name getSearchNameLists
Inputs None
Notes If no name lists are available, an empty array/list is returned. The return value itself is never null or blank.
Return type An array of NameListDetail objects (see Table 1)

Field Name	*Type*	*Required*	*Example*	*Notes*
nameListCode	String	yes	"PASSENGERS"	name of the name list
description	String	no	"Some passengers"	description of the name list

Table 1 - NameListDetail data type (SOAP)

Sample input SOAP
Sent using POST with a preemptive basic auth header to
http://<host>:<port>/ws/dispatcher/api/soap/SearchPort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services/">
  <soapenv:Header />
  <soapenv:Body>
    <ser:getSearchNameLists />
  </soapenv:Body>
</soapenv:Envelope>
```

Sample output SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:common="http://common/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getSearchNameListsResponse xmlns="http://services/">
      <nameLists xmlns="http://common/">
        <nameListCode>EMPLOYEES</nameListCode>
        <description>Our employees</description>
      </nameLists>
      <nameLists xmlns="http://common/">
        <nameListCode>PASSENGERS</nameListCode>
        <description>Some passengers</description>
      </nameLists>
    </getSearchNameListsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

2.2 Get strategies

A NameWorks *strategy* is a named combination of NameWorks configuration details, defined in a NameWorks configuration file. When searching for a name, you can optionally specify a strategy to influence the search behavior used by NameWorks. See the separate "Searching for Names with Enterprise Name Search" document for more information on strategies.

This method lists the strategies defined in the NameWorks configuration file used by your instance of ENS. These are the ones that you can optionally specify in a search request. The ENS search GUI uses this method when populating the dropdown where users can select a strategy.

This method takes no inputs. The output is a list of strategy names.

2.2.1 JSON/REST

Method:	GET
URL:	<host>:<port>/ws/dispatcher/api/rest/search/strategies
Inputs:	None.
Output:	The response body contains a JSON object with a single "strategies" field whose value is an array of strings. Each string is a strategy name.

Sample output JSON:

```
{
   "strategies": [
      "Broad",
      "Default",
      "Narrow"
   ]
}
```

2.2.2 SOAP

Port	Search
Name	getSearchStrategyCodes
Inputs	None
Return type	Array[String]

Sample input SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services/">
  <soapenv:Header />
  <soapenv:Body>
    <ser:getSearchStrategyCodes />
  </soapenv:Body>
</soapenv:Envelope>
```

Sample output SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:common="http://common/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getSearchStrategyCodesResponse xmlns="http://services/">
      <codes xmlns="http://common/">Broad</codes>
      <codes xmlns="http://common/">Default</codes>
      <codes xmlns="http://common/">Narrow</codes>
    </getSearchStrategyCodesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

2.3 Get cultures

This method lists the cultures defined in the NameWorks configuration file used by this instance of ENS. The cultures listed include the ones built into ENS plus any custom cultures defined in the NameWorks configuration file. If the configuration file specifies nonstandard display names for any cultures, this information is also included.

The ENS search GUI uses this method when populating the culture dropdowns in the search forms.

This method takes no inputs. The output is a list of objects describing cultures.

2.3.1 JSON/REST

Method: GET
URL: <host>:<port>/ws/dispatcher/api/rest/search/cultures
Input: None.
Output: The result is a JSON object with a single "cultures" field. That field's value is an array of objects. Each object in the array has fields as in Table 2.

Name	***Type***	***Example Value***	***Notes***
category	number	2	2 means that the culture should be listed in dropdowns other values mean that it should not.
code	number	1	numeric value for the culture enum
defaultDisplayName	Boolean	true	true means that the default display name (for which we have localized translations) should be used false means that this culture has a name specified in the configuration file (and listed here in the displayName field) which should be used instead
defined	Boolean	true	true means this culture has a definition. Most custom culture enum values are placeholders with no definition, and should not be listed in dropdowns. For them, "defined" is false.
displayName	string	"Anglo"	The name to display if the defaultDisplayName is false. This is mostly used for custom cultures.
enumValue	string	"ANGLO"	The enum value as a string.

Table 2 - Fields in a result element from the getCultures method (JSON)

Sample output JSON:


```
{
   "cultures":[
      {
         "category":0,
         "code":0,
         "defaultDisplayName":true,
         "defined":true,
         "displayName":"Ambiguous",
         "enumValue":"AMBIGUOUS"
      },
      {
         "category":2,
         "code":1,
         "defaultDisplayName":true,
         "defined":true,
         "displayName":"Anglo",
         "enumValue":"ANGLO"
      },

       (numerous entries omitted here)


      {
         "category":2,
         "code":60,
         "defaultDisplayName":true,
         "defined":false,
         "displayName":"Custom20",
         "enumValue":"CUSTOM20"
      }
   ]
}
```


2.3.2 SOAP

Port Search
Name getCultures
Inputs None
Return type Array[CultureDetail] (see Table 3)
Types used CultureDetail, for return values (Table 3)

Field Name	*Type*	*Example*	*Notes*
code	int	2	See Table 2 for all notes
enumValue	Culture	"ANGLO"	
category	int	0	
displayName	String	"Anglo"	
defaultDisplayName	Boolean	true	
defined	Boolean	true	

Table 3 – CultureDetail data type (SOAP)

Sample input SOAP


```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services/">
  <soapenv:Header />
  <soapenv:Body>
    <ser:getCultures />
```

```
  </soapenv:Body>
</soapenv:Envelope>
```

Sample output SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:common="http://common/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getCulturesResponse xmlns="http://services/">
      <cultures xmlns="http://common/">
        <code>0</code>
        <enumValue>AMBIGUOUS</enumValue>
        <category>0</category>
        <displayName>Ambiguous</displayName>
        <defaultDisplayName>true</defaultDisplayName>
        <defined>true</defined>
      </cultures>
      <cultures xmlns="http://common/">
        <code>1</code>
        <enumValue>ANGLO</enumValue>
        <category>2</category>
        <displayName>Anglo</displayName>
        <defaultDisplayName>true</defaultDisplayName>
        <defined>true</defined>
      </cultures>

      (numerous entries omitted here)

      <cultures xmlns="http://common/">
        <code>60</code>
        <enumValue>CUSTOM20</enumValue>
        <category>2</category>
        <displayName>Custom20</displayName>
        <defaultDisplayName>true</defaultDisplayName>
        <defined>false</defined>
      </cultures>
    </getCulturesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

2.4 Search

This method performs the main part of a NameWorks-based search. Specifically, it:

- Uses NameWorks to parse and analyze the name
- Searches for the name in NameWorks instances in one or more searchers
- Uses database information to map the search names found back to original source-name forms

- Returns the parses determined by NameWorks analysis, and the matching source names found by the search.
- In the search results, provides a list of distinct original source-name forms of the name. Each has a distinct combination of raw surname, raw given name, alt parse flag, surname culture, source name culture, script type, and (for Chinese-script names or Japanese-script organization names) the original-script given name and surname. Each result also includes match scores for the surname, given name, and overall name, and a list of counts by name list.

The user must have search privileges on the specified name list(s). If no name lists are specified in the request, the user must have search privileges on at least one name list. The search results only include names in lists for which the user has search permissions.

2.4.1 JSON/REST

Method: POST (because of complex inputs)
URL: <host>:<port>/ws/dispatcher/api/rest/search/matches
Input: JSON in the POST body, as follows.

2.4.1.1 Inputs

Inputs are provided in a JSON object in the POST body, including fields as listed here:

Field Name	***Type***	***Req'd***	***Notes***
nameType	String	yes	This can be one of the following: • "PARSED" for a personal name supplied as givenName and surname, with optional givenNameCulture and surnameCulture. There must be at least a nonblank givenName or a nonblank surname (or both). • "FULL" for a personal name supplied as a single nameText with optional fullNameCulture. The nameText must be nonblank. • "ORGANIZATION" for an organization name supplied as a single nonblank nameText. • "GENERIC" for a name of unspecified category supplied as a single nonblank nameText. Note that this field determines what input parameters are used in providing the query name. It does not specify the categories of names to be searched for; see searchCategories for that.
givenName	String	no	For nameType PARSED only, the given name portion of a personal name.
givenNameCulture	String	no	For nameType PARSED only, the optional culture for the given name. Knowing the culture of a name can improve search results by letting GNM use culture-specific mapping rules and settings. If this is omitted or null/blank, the system will attempt to determine a culture from the name's script and/or content. Otherwise, the legal values are "AFGHANI", "AMBIGUOUS", "ANGLO", "ARABIC", "CHINESE", "EUROPEAN", "FARSI", "FRENCH", "GERMAN", "HAN", "HISPANIC", "INDIAN", "INDONESIAN", "JAPANESE", "KOREAN", "PAKISTANI", "POLISH", "PORTUGUESE", "RUSSIAN", "SOUTHWESTASIAN", "THAI",

			"TURKISH", "VIETNAMESE", and "YORUBAN", as well as "CUSTOM1" through "CUSTOM20". These should correspond to the names in the enumeration in the NameWorks API.
surname	String	no	For nameType PARSED only, the surname portion of a personal name. The givenName and surname cannot both be blank
surnameCulture	String	no	For nameType PARSED only, the optional culture for the surname. If omitted or null/blank, the system will choose a culture. Legal values are as shown above.
nameText	String	no	For nameType FULL, ORGANIZATION, or GENERIC, the full text of the name, which may not be blank.
culture	String	no	For nameType FULL or ORGANIZATION, the culture for the full name. If omitted or null/blank, the system will attempt to choose a culture based on the script or (for personal names) the name text. Legal values are as shown above.
strategyCode	String	no	The name of a strategy as provided by getSearchStrategyCodes. May be omitted.
maxResults	Int	no	Maximum number of results. If not specified, this is determined by the strategy or a system default.
includeAlternate Parses	Boolean	no	For a personal name, true means that the system should analyze the name for a possible better parse, and search based on that as well as on the specified parse.
searchCategories	Array of Strings	no	A list of the categories wanted. The values in the list can be "PERSONAL" or "ORGANIZATION", so the only meaningful inputs are: ["PERSONAL"] ["ORGANIZATION"] ["PERSONAL","ORGANIZATION"]
minScore	Int	no	The minimum full-name score to qualify a search candidate for inclusion in the result set. Specify -1 or omit this field to use a default minScore based on system defaults, or on the search strategy if one is being used.
nameListCodes	Array of strings	no	An array of name list codes, identifying the name lists to be searched. If omitted or empty, the search includes all name lists which this user is authorized to search. The names here should always be values originally provided by the getNameListCodes service.
scriptType	String	no	Optionally identifies the script type for a name. In most cases, ENS can determine the script type by examining the input string for the name. This parameter is only required for organization names using Chinese or Japanese script, where the distinction cannot be determined automatically. In such cases, this parameter lets a caller supply an explicit script hint. Values for this case would be "JAPANESESCRIPT" or "CHINESESCRIPT".

2.4.1.2 Sample Input JSON for a parsed personal name

```
{
   "nameType":                "PARSED",
   "givenName":               "Jose",
   "surname":                 "Martin",
   "givenNameCulture":        "HISPANIC",
   "surnameCulture":          "ANGLO",
   "strategyCode":            "Broad",
   "searchCategories":        ["PERSONAL"],
   "nameListCodes":           ["EMPLOYEES", "PASSENGERS"],
   "minScore":                75,
   "maxResults":              400,
```

```
   "includeAlternateParses":     true
}
```

2.4.1.3 Sample Input JSON for a full personal name

```
{
   "nameType":                   "FULL",
   "nameText":                   "Jose Martin",
   "strategyCode":               "Broad",
   "searchCategories":           ["PERSONAL" ],
   "nameListCodes":              ["EMPLOYEES", "PASSENGERS"],
   "minScore":                   75,
   "maxResults":                 400,
   "includeAlternateParses":     true
}
```

2.4.1.4 Sample Input JSON for an organization name, not specifying name lists

In this example, we use the default strategy, minScore, and maxResults, and don't specify any name lists. Note that the searchCategories parameter is independent of the type of name provided.

```
{
   "nameType":                   "ORGANIZATION",
   "nameText":                   "Martin Guitars",
   "searchCategories":           ["PERSONAL", "ORGANIZATION"],
   "includeAlternateParses":     false
}
```

2.4.1.5 Sample Input JSON for an unspecified-category name

In this example, we use the default strategy, minScore, and maxResults, and don't specify any name lists.

```
{
   "nameType":                   "GENERIC",
   "nameText":                   "Martin Guitars",
   "searchCategories":           ["PERSONAL", "ORGANIZATION"],
   "includeAlternateParses":     false
}
```

2.4.1.6 Outputs

The output object, in JSON, has two main fields:

1. `analyzedNames`: shows the parses of the query name used in the search. It is a list of 1-3 objects representing QueryName objects from analyzeForSearch.
2. `nameResults`: This represents the search results. It is an array of result objects. Each object represents one distinct combination of search name id, raw given name, raw surname, alt parse flag, and (where Chinese scripts are used in input) script type, original given name, and original surname.

Each entry in the `analyzedNames` list contains the following fields to describe the result of NameWorks analysis.

Subfield Name	***Type***	***Example***	***Notes***
categories	array of strings	["PERSONAL"]	The set of categories that this name appears to fall into. Can be

			["PERSONAL"], ["ORGANIZATION"], or ["PERSONAL", "ORGANIZATION"].
confidence	int	60	Parse confidence value. 100 is best
givenName	string	"JOSE"	Given name (for a person), transliterated to upper-case roman letters.
givenName Culture	culture name	"AMBIGUOUS"	Single culture for the given name. If the external caller originally supplied an explicit givenNameCulture, that is shown here.
givenName CultureSet	array of culture names	["ANGLO", "KOREAN"]	Set of possible cultures for a personal given name. If an external caller originally supplied an explicit givenNameCulture, this list contains only that specified culture.
name	string	"SMITH BARNEY"	Text of the full name; used for organization names.
nameCulture	culture name	"AMBIGUOUS"	Single culture for the name.
nameCultureSet	array of culture names	["ANGLO", "KOREAN"]	Set of possible cultures for the full name. If an external caller originally supplied an explicit culture, this list contains only that specified culture.
qualifiers	strings	"ESQ"	Text of qualifiers.
scriptType	String	"LATINSCRIPT"	Script type for the name
surname	string	"MARTIN"	Text of the surname, transliterated to upper-case roman letters.
surnameCulture	culture name	"AMBIGUOUS"	Single culture for the surname. If the external caller originally supplied an explicit surnameCulture, that is shown here.
surname CultureSet	array of culture names	["ANGLO", "KOREAN"]	Set of possible cultures for the surname. If an external caller originally supplied an explicit surnameCulture, this list contains only that culture.
titles	string	"HERR PROFESSOR DOCTOR"	Text of the titles.

Table 4 - Fields in an analyzedName entry in search method output

In an entry in `nameResults`, the fields shown for a personal name are:

Field Name	***Type***	***Example***	***Notes***
searchNameID	String[1]	1200001	Internal numeric id for a search name, assigned by ENS when the name was loaded. Expressed as a string because JSON cannot exactly represent the full range of Java long values.
nameCategory	String	PERSONAL	Name category.
rawGivenName	String	José	From source name. Can have accents or non-Roman characters
rawSurname	String	Martín	Ditto.
givenName	String	JOSE	From search name. Transliterated Roman all-caps.
givenNameCulture	String	HISPANIC	From search name.
surname	String	MARTIN	From search name. Transliterated Roman all-caps.
surnameCulture	String	EUROPEAN	From search name.

[1] The searchNameID is passed as a string even though on the server side it is represented as a Java long. This is because JavaScript cannot exactly represent Java longs as numbers (it falls back to floating point for large values). Treating searchNameID as a string ensures that different values are kept distinct, and one ID is not rounded into another.

alternate	Boolean	true	Indicates whether the search name found was originally added to the database as the result of an alternate parse.
surnameScore	number	80	Score for the surname match. 100 is best.
givenNameScore	number	90	Score for the given name match.
score	number	95	Score for the match, overall.
scriptType	String	LATINSCRIPT	Script type for the name
regularized	Boolean	true	Indicates whether the score is based on a match of regularized names or "original" (though Romanized, transliterated, and all-caps) names.
externalReferences	Array	see below	Array of objects representing how many external references to this result were found in each name list. Each object in this array has a nameListCode (a string like "PASSENGERS") and a count (a number).

Table 5 - Fields in a personal nameResults entry in search method output

For an organization name, the fields in a listed `nameResult` entry are:

Field Name	***Type***	***Example***	***Notes***
searchNameID	String	1200001	A generated numeric id for a search name (as found in ens_search_name.search_name_id).
nameCategory	String	ORGANIZATION	Category of the name
rawOrganizationName	String	Smith Barney	Original raw form of the organization name.
organizationName	String	SMITH BARNEY	From search name. Transliterated Roman all-caps.
organizationNameCulture	String	AMBIGUOUS	From search name. This is either as specified in the original input for the name, or inferred from the name's script, or AMBIGUOUS.
score	number	95	Score for the match. 100 is best.
regularized	Boolean	true	Indicates whether the score is based on a match of regularized names or "original" (though Romanized, transliterated, and all-caps) names. Probably this is always true for an organization name.
scriptType	String	LATINSCRIPT	Script type for the name
externalReferences	Array	see below	Array of objects representing how many external references to this result were found in each name list. Each object in this array has a nameListCode (a string like "PASSENGERS") and a count (a number).

Table 6 - Fields in a organization nameResults entry in search method output

2.4.2 Sample response JSON

This is for a personal name with one parse (analyzedNames entry) and two search results (nameResults entries).


```
{
   "analyzedNames":[
      {
         "alternate":false,
         "categories":[
            "PERSONAL"
         ],
         "confidence":83,
         "givenName":"MARTIN",
         "givenNameCulture":"EUROPEAN",
```

```
            "givenNameCultureSet":[
               "ANGLO",
               "HISPANIC",
               "FRENCH",
               "GERMAN"
            ],
            "name":"MARTIN ALBURY",
            "nameCulture":"ANGLO",
            "nameCultureSet":[
               "ANGLO"
            ],
            "originalGivenName":"Martin",
            "originalSurname":"Albury",
            "qualifiers":"",
            "scriptType":"NOSCRIPT",
            "surname":"ALBURY",
            "surnameCulture":"ANGLO",
            "surnameCultureSet":[
               "ANGLO"
            ],
            "titles":""
         }
      ],
      "nameResults":[
         {
            "alternate":false,
            "externalReferences":[
               {
                  "count":1,
                  "nameListCode":"PASSENGERS"
               }
            ],
            "givenName":"MARTIN",
            "givenNameCulture":"EUROPEAN",
            "givenNameScore":100,
            "nameCategory":"PERSONAL",
            "originalGivenName":"",
            "originalSurname":"",
            "rawGivenName":"MARTIN",
            "rawSurname":"ALBURY",
            "regularized":false,
            "score":100,
            "scriptType":"NOSCRIPT",
            "searchNameID":"100482551",
            "surname":"ALBURY",
            "surnameCulture":"ANGLO",
            "surnameScore":100
         },
         {
            "alternate":false,
            "externalReferences":[
               {
                  "count":1,
                  "nameListCode":"PASSENGERS"
               }
            ],
            "givenName":"MARTINEZ",
            "givenNameCulture":"EUROPEAN",
            "givenNameScore":75,
            "nameCategory":"PERSONAL",
            "originalGivenName":"",
            "originalSurname":"",
            "rawGivenName":"MARTINEZ",
            "rawSurname":"ALBORY",
            "regularized":true,
            "score":75,
            "scriptType":"NOSCRIPT",
            "searchNameID":"100482405",
            "surname":"ALBORY",
            "surnameCulture":"ANGLO",
            "surnameScore":75
         }
```

```
    ]
}
```

2.4.3 SOAP

Port Search
Name searchName
Input Parameters

Parameter Name	*Parameter Type*	*Req'd*	*Description*
name	Name (see description for concrete subtypes)	yes	The abstract Name object representing the name being searched for. Concrete subtypes are: ParsedName (Table 16), FullName (Table 17), Organization Name (Table 18), and GenericName (Table 19). An exception is generated if this is null/blank or if any required fields of the concrete subtype are null or blank.
nameListCodes	Array[String]	no	Array of name list codes identifying the name lists to be searched, or null/blank/empty-array if all name lists which the user is authorized to search should be searched. An exception is generated if an unrecognized name list is specified or if a name list is specified for which the user is not authorized.
strategyCode	String	no	The string identifier for the search strategy to use to override the default search settings, or null/blank if the system-configured default settings should be used.
searchCategories	Array[NameCategory]	no	The name categories for names to be considered during the search, or null/blank/empty-array if names of all categories should be considered as result candidates.
maxResults	Integer	no	The maximum number of results to be returned by the search. Specify negative-one (-1) to use the default value configured with the search strategy being used, or 0 to indicate no limit.
minScore	Integer	no	Minimum full-name score to qualify a search candidate for inclusion in the result set. Specify negative-one (-1) to use the default configured with the search strategy being used.
includeAlternate Parses	Boolean	no	Set to true if you want to include results based on alternate parses. (Default is false).
scriptType	ScriptType	no	Optional explicit script type.

Return type SearchResult object (Table 7), containing
(1) an array of AnalyzedName objects (Table 8) for parses, and
(2) an array of SearchNameResult objects for names found

The concrete subtypes of SearchNameResult are SearchNamePersonal (Table 10) and SearchNameOrganization (Table 11).

Field Name	*Type*	*Notes*
analyzedNames	Array[AnalyzedName]	The parses resulting from analysis of the query name

nameResults	Array[SearchNameResult]	The names found in the search

Table 7 - SearchResult data type (SOAP)

Field Name	*Type*	*Notes*
categories	NameCategory	The categories for the name (PERSONAL and/or ORGANIZATION)
name	String	Full name.
nameCulture	Culture	The single recommended culture for the name as a whole.
nameCultureSet	Culture(s)	The set of recommended cultures for the name as a whole.
givenName	String	Given name
givenNameCulture	Culture	The recommended culture for the given name.
givenNameCultureSet	Culture(s)	The recommended cultures for the given name .
surname	String	Surname
surnameCulture	Culture	The recommended culture for the surname.
surnameCultureSet	Culture(s)	The recommended cultures for the surname .
alternate	Boolean	Indicates whether this is an alternate parse of the original name.
confidence	Integer	confidence of this parse
qualifiers	String	qualifiers found in the name
titles	String	titles found in the name

Table 8 - AnalyzedName data type (SOAP)

Field Name	*Type*	*Notes*
searchNameID	Long	An internal identification number used to identify the analyzed name loaded internally into NameWorks. When represented in JSON, searchNameIDs are passed as strings because JavaScript does not support exact representation of 64-bit integers.
nameCategory	NameCategory	ORGANIZATION or PERSONAL
score	int	The match score for this name against the query name.
regularized	Boolean	Indicates whether name regularization contributed to the score.
externalReferences	Array[ExternalReference]	Name/count pairs showing how many times this name appears in particular name lists.

Table 9 - SearchNameResult base data type (SOAP)

Field Name	*Type*	*Notes*
(See also the base type SearchNameResult, in Table 9)		
rawGivenName	String	Given name as originally received
rawSurname	String	Surname as originally received
originalGivenName	String	For Chinese Script, the original given name
originalSurname	String	For Chinese Script, the original surname
scriptType	ScriptType	Currently this can be CHINESESCRIPT or NOSCRIPT
surname	String	Surname as transliterated
surnameCulture	Culture	Culture for the surname, as originally provided or as analyzed
surnameScore	Int	Score for the surname match
givenName	String	Given name
givenNameCulture	Culture	Culture for the given name, as originally provided or as analyzed
givenNameScore	Int	Score for the given name match
alternate	Boolean	true if this was an alternate parse of the original name

Table 10 - SearchNamePersonal data type (SOAP)

Field Name	*Type*	*Notes*
(See also the base type SearchNameResult, in Table 9)		
rawOrganizationName	String	organization name as originally received
organizationName	String	organization name as transliterated
organizationNameCulture	String	Culture for the organization, as originally provided or as analyzed

Table 11 – SearchNameOrganization data type (SOAP)

Sample input SOAP – search for a parsed personal name

Sent using POST with a preemptive basic auth header to
http://<host>:<port>/ws/dispatcher/api/soap/SearchPort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services/">
  <soapenv:Header />
  <soapenv:Body>
    <ser:searchName>
      <name type="ParsedName">
        <givenName>Jose</givenName>
        <surname>Alfaro</surname>
      </name>
      <maxResults>100</maxResults>
      <minScore>80</minScore>
      <includeAlternateParses>false</includeAlternateParses>
    </ser:searchName>
  </soapenv:Body>
</soapenv:Envelope>
```

Sample output SOAP – search for a parsed personal name

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:common="http://common/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <searchNameResponse xmlns="http://services/">
      <searchResult xmlns="http://common/">
        <analyzedNames xsi:type="common:AnalyzedName">
          <categories>PERSONAL</categories>
          <scriptType>LATINSCRIPT</scriptType>
          <name>JOSE ALFARO</name>
          <nameCulture>HISPANIC</nameCulture>
          <nameCultureSet>HISPANIC</nameCultureSet>
          <givenName>JOSE</givenName>
          <givenNameCulture>EUROPEAN</givenNameCulture>
          <givenNameCultureSet>HISPANIC</givenNameCultureSet>
          <givenNameCultureSet>PORTUGUESE</givenNameCultureSet>
          <surname>ALFARO</surname>
          <surnameCulture>HISPANIC</surnameCulture>
          <surnameCultureSet>HISPANIC</surnameCultureSet>
          <alternate>false</alternate>
          <confidence>98</confidence>
```

```
</analyzedNames>
<nameResults xsi:type="common:SearchNamePersonal">
  <searchNameID>100438478</searchNameID>
  <nameCategory>PERSONAL</nameCategory>
  <score>100</score>
  <regularized>false</regularized>
  <externalReferences>
    <nameListCode>PASSENGERS</nameListCode>
    <count>1</count>
  </externalReferences>
  <rawGivenName>JOSE</rawGivenName>
  <rawSurname>ALFARO</rawSurname>
  <scriptType>LATINSCRIPT</scriptType>
  <surname>ALFARO</surname>
  <surnameCulture>HISPANIC</surnameCulture>
  <surnameScore>100</surnameScore>
  <givenName>JOSE</givenName>
  <givenNameCulture>EUROPEAN</givenNameCulture>
  <givenNameScore>100</givenNameScore>
  <alternate>false</alternate>
</nameResults>
<nameResults xsi:type="common:SearchNamePersonal">
  <searchNameID>100438498</searchNameID>
  <nameCategory>PERSONAL</nameCategory>
  <score>98</score>
  <regularized>false</regularized>
  <externalReferences>
    <nameListCode>PASSENGERS</nameListCode>
    <count>1</count>
  </externalReferences>
  <rawGivenName>JOSE BERNARDO</rawGivenName>
  <rawSurname>ALFARO PEON</rawSurname>
  <scriptType>LATINSCRIPT</scriptType>
  <surname>ALFARO PEON</surname>
  <surnameCulture>HISPANIC</surnameCulture>
  <surnameScore>98</surnameScore>
  <givenName>JOSE BERNARDO</givenName>
  <givenNameCulture>EUROPEAN</givenNameCulture>
  <givenNameScore>99</givenNameScore>
  <alternate>false</alternate>
</nameResults>
<nameResults xsi:type="common:SearchNamePersonal">
  <searchNameID>100438750</searchNameID>
  <nameCategory>PERSONAL</nameCategory>
  <score>98</score>
  <regularized>false</regularized>
  <externalReferences>
    <nameListCode>PASSENGERS</nameListCode>
    <count>1</count>
  </externalReferences>
  <rawGivenName>JOSE ANTONIO</rawGivenName>
  <rawSurname>ALFARO UGALDE</rawSurname>
  <scriptType>LATINSCRIPT</scriptType>
  <surname>ALFARO UGALDE</surname>
  <surnameCulture>HISPANIC</surnameCulture>
```

```
          <surnameScore>98</surnameScore>
          <givenName>JOSE ANTONIO</givenName>
          <givenNameCulture>EUROPEAN</givenNameCulture>
          <givenNameScore>99</givenNameScore>
          <alternate>false</alternate>
        </nameResults>
      </searchResult>
    </searchNameResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

2.5 Get external references

This method retrieves all the external references associated with some list of search names and associated raw name forms.

The search method in section 2.4 finds distinct original forms of names matching some query name. It returns (along with parse information on the query name) a list of search result objects, each describing a distinct combination of search name, category, raw given name, raw surname (or raw organization name). For each such result, it shows how many references to that name were found in each name list, but it does not list the actual external ids of those references.

This method provides that additional detail. Given information from an entry in the main search results, it lists the external references found. For each one it lists the name list id and externalId, with some additional information.

The user must have search privileges on the specified name list(s). If no name lists are specified, the user must have search privileges on at least one name list.

2.5.1 JSON/REST

Method: POST (because of complex inputs)
URL: <host>:<port>/ws/dispatcher/api/rest/search/references
Input: JSON in the POST body, as follows:

2.5.1.1 Inputs

For a personal name, the input is a JSON object with fields as shown in Table 13. The rawGivenName and rawSurname may not both be blank.

Field Name	*Type*	*Required*	*Notes*
searchNameIDs	Array of strings	yes	As received in the searchNameIDs field of a result from searchNames.
rawGivenName	String	see above	As in the rawGivenName field of a result from searchNames.
rawSurname	String	see above	As in the rawSurname field of a result from searchNames.
nameListCodes	Array of strings	no	Name list names as received from the GetSearchNameList service. If omitted, all name lists are considered.
maxResults	int	no	Maximum number of results to be displayed

Table 12 - Inputs for getExternalReferences for a personal name

For an organization, the input fields are as in Table 13.

Field Name	*Type*	*Required*	*Notes*
searchNameIDs	Array of strings	yes	As received in the searchNameIDs field of a result from searchNames.
rawOrganizationName	String	no	As in the rawOrganizationName field of a result from searchNames. If provided, this must be nonblank.
nameListCodes	Array of strings	no	Name list names as received from the GetSearchNameList service. If omitted, all name lists are considered.
maxResults	int	no	Maximum number of results to be displayed

Table 13 - Inputs for getExternalReferences for an organization name

Sample JSON input – personal name

```
{
   "searchNameIDs":[
      "100482551"
   ],
   "rawSurname":"ALBURY",
   "rawGivenName":"MARTIN",
   "alternate":false,
   "nameListCodes":[

   ],
   "strategyCode":""
}
```

Sample JSON input – organization name

```
{
   "searchNameIDs":[
      "100480817"
   ],
   "rawOrganizationName":"Martin Guitars",
   "nameListCodes":[

   ],
   "strategyCode":""
}
```

2.5.1.2 Outputs

The response body contains a JSON object with a single field named "externalReferences". That field's value is an array of objects, each representing an external reference to a person or to an organization.

For a personal-name external reference, the entry has the following subfields:

Subfield Name	*Type*	*Example*	*Notes*
searchNameID	String	"1000000012"	The client can use this information, combined with results from an earlier NameSearch request, to display separate scores for different external references.
nameListCode	String	"AIRLINE_EMPLOYEES"	Identifies the name list
externalID	String	"abcdef12345"	Original external ID for the source name
rawGivenName	String	"John"	Original raw given name.
rawSurname	String	"Smith"	Original raw surname.
givenNameCulture	String	"ANGLO"	Original given-name culture

surnameCulture	String	"HISPANIC"	Original surname culture

For an organization-name external reference, the entry has the following subfields:

Subfield name	*Type*	*Example*	*Notes*
searchNameID	String	"1000000012"	See above.
nameListCode	String	"CUSTOMERS"	Identifies the name list
externalID	String	"abcdef12345"	Original external ID for the source name
rawOrganizationName	String	"Smith Barney"	Original raw org name
culture	String	"ANGLO"	Original org-name culture

Sample JSON output – personal name, with one external reference found


```
{
   "externalReferences":[
      {
         "externalID":"909546",
         "givenNameCulture":"EUROPEAN",
         "nameListCode":"PASSENGERS",
         "rawGivenName":"MARTIN",
         "rawSurname":"ALBURY",
         "searchNameID":"100482551",
         "surnameCulture":"ANGLO"
      }
   ]
}
```


Sample JSON output – organization name, with two external references found


```
{
   "externalReferences":[
      {
         "externalID":"mg55555",
         "nameListCode":"PASSENGERS",
         "organizationNameCulture":"AMBIGUOUS",
         "rawOrganizationName":"Martin Guitars",
         "searchNameID":"100480817"
      },
      {
         "externalID":"mg11111",
         "nameListCode":"PASSENGERS",
         "organizationNameCulture":"AMBIGUOUS",
         "rawOrganizationName":"Martin Guitars",
         "searchNameID":"100480817"
      }
   ]
}
```


2.5.2 SOAP

Port Search
Name getExternalReferences
Input Parameters

Parameter Name	*Type*	*Req'd*	*Description*
searchNameIDs	Array of 64-bit integers	Yes	As received in the searchNameIDs field of a result from searchName. An internal identification number used to identify unique search names found in the search method. A particular search name may be linked to multiple source names.
rawGivenName	String	see	As in the rawGivenName field of a result from

		note	searchName.
rawSurname	String	see note	As in the rawSurname field of a result from searchName.
rawOrganizationName	String	see note	As in the rawOrganizationName field of a result from searchName.
nameListCodes	Array[String]	no	An array of name list names identifying the name lists to be searched, or null/blank/empty-array if all name lists which the user is authorized to search should be searched. An exception is generated if an unrecognized name list is specified or if a name list is specified for which the user is not authorized.
maxResults	Integer	no	The maximum number of results to be returned by this method. Specify negative-one (-1) to use the default value configured in the database., or 0 to indicate no limit.

Note At least one of rawGivenName, rawSurname, and rawOrganizationName must be specified. If rawOrganizationName is specified, neither rawGivenName nor rawSurname may be specified.

Return type Array of ExternalReferenceDetail objects (see Table 14 and Table 15).

Field Name	***Type***	***Example***	***Notes***
nameListCode	String	"PASSENGERS	Name of the external data source (name list name).
externalID	String	"601332"	The text representation of the identifier for the record with which the name is associated in the external system. In this example the externalID is short and numeric, but it can be longer and alphanumeric.
searchNameID	String	"100438478"	As received in the searchNameIDs field of a result from searchName. The internal identification numbers used to identify unique search names. A particular search name may be linked to multiple source names..
rawSurname	String	"ALFARO"	Raw source form of the surname as received in the original data by NameLoader or the addName service. This may be in mixed case and/or in non-Roman script.
rawGivenName	String	"JOSE"	Raw source form of the given name as received in the original data by NameLoader or the addName service. This may be in mixed case and/or in non-Roman script.
givenNameCulture	String	"EUROPEAN"	Culture for the source name.
surnameCulture	String	"HISPANIC"	Culture for the source name.

Table 14 - External Reference Detail data type for personal name (SOAP)

Field Name	***Type***	***Example***	***Notes***
nameListCode	String	"PASSENGERS	Name of the external data source (name list name).
externalID	String	"601332"	The text representation of the identifier for the record with which the name is associated in the external system. In this example the externalID is short and numeric, but it can be longer and alphanumeric.
searchNameID	String	"100438478"	As received in the searchNameIDs field of a result from searchName. The internal identification numbers used to identify unique search names. A particular search name may be linked to multiple source names.
rawSurname	String	"ALFARO"	Raw source form of the surname as received in the original data by NameLoader or the addName service. This may be in mixed case and/or in non-Roman script.

rawGivenName	String	"JOSE"	Raw source form of the given name as received in the original data by NameLoader or the addName service. This may be in mixed case and/or in non-Roman script.
givenNameCulture	String	"EUROPEAN"	Culture for the source name.
surnameCulture	String	"HISPANIC"	Culture for the source name.

Table 15 - External Reference Detail data type for organization name (SOAP)

Sample input SOAP – get external references for a personal name

Sent using POST with a preemptive basic auth header to
http://<host>:<port>/ws/dispatcher/api/soap/SearchPort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services/">
  <soapenv:Header />
  <soapenv:Body>
    <ser:getExternalReferences>
      <searchNameIDs>100438478</searchNameIDs>
      <nameCategory>PERSONAL</nameCategory>
      <rawSurname>ALFARO</rawSurname>
      <rawGivenName>JOSE</rawGivenName>
      <alternate>false</alternate>
      <strategyCode />
    </ser:getExternalReferences>
  </soapenv:Body>
</soapenv:Envelope>
```

Sample output SOAP – get external references for a personal name

In this example there is just a single reference found in a single namelist. If there were more, there would be more <externalRef> elements. The main output information in each externalRef is the combination of nameListCode and externalID.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:common="http://common/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getExternalReferencesResponse xmlns="http://services/">
      <externalRef xmlns="http://common/"
xsi:type="common:ExternalReferenceDetailPersonal">
        <nameListCode>PASSENGERS</nameListCode>
        <externalID>601332</externalID>
        <searchNameID>100438478</searchNameID>
        <rawSurname>ALFARO</rawSurname>
        <rawGivenName>JOSE</rawGivenName>
        <givenNameCulture>EUROPEAN</givenNameCulture>
        <surnameCulture>HISPANIC</surnameCulture>
      </externalRef>
    </getExternalReferencesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Sample input SOAP – get external references for an organization name
Sent using POST with a preemptive basic auth header to
http://<host>:<port>/ws/dispatcher/api/soap/SearchPort

In this example, the search call had reported multiple search names associated with the same source name form, and the raw source name happens to be in mixed case.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services/">
  <soapenv:Header />
  <soapenv:Body>
    <ser:getExternalReferences>
      <searchNameIDs>100436007</searchNameIDs>
      <searchNameIDs>100441500</searchNameIDs>
      <nameCategory>ORGANIZATION</nameCategory>
      <rawOrganizationName>Martin Guitars</rawOrganizationName>
      <strategyCode />
    </ser:getExternalReferences>
  </soapenv:Body>
</soapenv:Envelope>
```

Sample output SOAP – get external references for an organization name

The output in this example includes multiple results, though they still happen to all be found in the same namelist. As above, the most important information is the combination of name list code and external id in each externalRef element.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:common="http://common/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getExternalReferencesResponse xmlns="http://services/">
      <externalRef xmlns="http://common/"
xsi:type="common:ExternalReferenceDetailOrganization">
        <nameListCode>CUSTOMERS</nameListCode>
        <externalID>mg11111</externalID>
        <searchNameID>100436007</searchNameID>
        <rawOrganizationName>Martin Guitars</rawOrganizationName>
      </externalRef>
      <externalRef xmlns="http://common/"
xsi:type="common:ExternalReferenceDetailOrganization">
        <nameListCode>CUSTOMERS</nameListCode>
        <externalID>mg55555</externalID>
        <searchNameID>100436007</searchNameID>
        <rawOrganizationName>Martin Guitars</rawOrganizationName>
      </externalRef>
      <externalRef xmlns="http://common/"
xsi:type="common:ExternalReferenceDetailOrganization">
        <nameListCode>CUSTOMERS</nameListCode>
        <externalID>x12345</externalID>
        <searchNameID>100441500</searchNameID>
```

```
            <rawOrganizationName>Martin Guitars</rawOrganizationName>
          </externalRef>
        </getExternalReferencesResponse>
      </soapenv:Body>
    </soapenv:Envelope>
```

3 Methods for Managing Names

This section lists the methods for managing names available in the ENS web service. For all of these methods, the HTTP request must include either a preemptive basic authentication header as in section 4.2, or an ensSecurityCookie for a session established by a login request as described in section 4.3.1.

The user thus identified must have privileges as follows:

- For `addName` and `removeName`, the user must have manager privileges on the relevant name list. This means either being a member of a specific numbered manager group associated with that list, or being a member of the more general manager_all_lists or admin group.
- For `getNameDetails`, the user must have search privileges on the relevant name list. This means either being a member of a specific numbered searcher or manager group associated with that list, or being a member of the more general searcher_all_lists, manager_all_lists, or admin group.

The assigning of users to groups is described in the separate "Setting Up and Managing Enterprise Name Search" document.

3.1 Add name

This method adds a single source name and related information to ENS. More precisely, it:

- Adds a single source name in the database,
- Uses NameWorks to parse and analyze the name, to determine one or more search names that will represent this name in searchers
- Adds those search names to the database (or finds existing search names already present that should be used)
- Adds linking records to map the search names to the source name
- Adds revision records used to inform searchers about the added name(s) that they need to load. Any running searchers check for these records at intervals, and pick up the new search names that belong to their assigned partitions.

This behavior on individual names is essentially the same as that of the NameLoader program if it is run while the cell is active.

This web service will not add a new name list to the system. That must be done using NameLoader. Once a name list exists, you can use either NameLoader or the web service to add additional names. For adding large numbers of names, NameLoader is faster. See the separate "Setting Up and Managing Enterprise Name Search" document for more information on NameLoader.

This method is not used by the ENS GUI client.

3.1.1 JSON/REST

Method: PUT
URL: <host>:<port>/ws/dispatcher/api/rest/namelist/<nameListCode>/name/<externalID>
Input:
When adding a parsed personal name, the input JSON includes the following fields:

Field Name	*Type*	*Req'd*	*Example Value*	*Notes*
nameCategory	String	yes	"PERSONAL"	For a personal name, "PERSONAL".
givenName	String	See note	"永裕"	Given name.
surname	String	See note	"莊"	Surname.
givenNameCulture	String	no	"CHINESE"	One of the enum strings returned by getCultures (section 2.3)
surnameCulture	String	no	"CHINESE"	as above
nameListCode	String	yes	"PASSENGERS"	Name of an already-existing name list, as listed by getNameLists (section 2.1)
externalID	String	yes	"xyz123"	External id for the name, often a key into some data store external to ENS. For ENS, this is an opaque string.
includeAlternateParses	Boolean	no	false	true means ENS should also add a different parse if one is found that is more plausible than the parse provided
scriptType	String	no		Very rarely used. See note below.

Note: If the givenName is blank or omitted, the surname is required.

The scriptType parameter is only required for one obscure case: a Japanese name consisting entirely of Chinese characters (with no Kana), such that the system cannot automatically determine the script type. For those cases, it should be set to "JAPANESESCRIPT". In all other cases, this parameter should be omitted.

For a full personal name, the input fields are:

Field Name	*Type*	*Req'd*	*Example Value*	*Notes*
nameCategory	String	yes	"PERSONAL"	For a personal name, "PERSONAL".
nameText	String	yes	"莊永裕"	Full name.
culture	String	no	"CHINESE"	One of the enum strings returned by getCultures (section 2.3)
nameListCode	String	yes	"PASSENGERS"	Name of a name list, as listed by getNameLists (section 2.1)
externalID	String	yes	"xyz123"	External id for the name, often a key into some data store external to ENS. For ENS, this is an opaque string.
includeAlternateParses	Boolean	no	false	true means ENS should also add a different parse if one is found that is more plausible than the parse provided

scriptType	String	no		Very rarely used. See note above.

For an organization name, the input fields are:

Field Name	*Type*	*Req'd*	*Example Value*	*Notes*
nameCategory	String	yes	"ORGANIZATION"	For an organization name, "ORGANIZATION".
nameText	String	yes	"Smith Brothers"	Full name.
culture	String	no	"CHINESE"	One of the enum strings returned by getCultures (section 2.3)
nameListCode	String	yes	"CUSTOMERS"	Name of a name list, as listed by getNameLists (section 2.1)
externalID	String	yes	"xyz123"	External id for the name, often a key into some data store external to ENS. For ENS, this is an opaque string.
scriptType	String	no		Very rarely used. See note above.

Sample input JSON:

```
{
   "nameCategory":"PERSONAL",
   "givenName":"永裕",
   "surname":"莊",
   "surnameCulture":"CHINESE",
   "givenNameCulture":"CHINESE",
   "nameListCode":"PASSENGERS",
   "externalID":"xyz123",
   "includeAlternateParses":false
}
```

Output: The response body contains a JSON object with a "nameID" field. That field's string value is the new source name's id. This id is internal to ENS, typically not used by client programs.

Sample output JSON:

```
{
   "nameID": "100453500"
}
```

3.1.2 SOAP

Port NameList
Name addName
Input Parameters

Name	*Type*	*Notes*
name	Name	Name object representing the name being added. An exception is generated if this is null/blank or the required fields of the concrete Name-derived type are null/blank.
externalRef	ExternalReference (see Table 20)	Identifies the name within some external system by specifying a nameListCode (name list name) and external ID.
nameCategory	NameCategory	PERSONAL or ORGANIZATION
includeAlternateParses	Boolean	Optional. Not used for organization name.

Notes If the specified external reference identifies an existing name and the name has not changed, then nothing is done. The return value is the nameID of the existing name.

Output Source name id (String) for the new or existing name. Client applications typically do not use this information, but it helps serve as a confirmation that the addition succeeded.

The Name type in the first parameter is abstract. Its concrete subtypes are ParsedName (Table 16), FullName (Table 17), Organization Name (Table 18), and GenericName (Table 19).

Field Name	*Type*	*Example*	*Notes*
givenName	String	"John"	The non-null/non-blank text of the given name portion of the personal name.
givenNameCulture	Culture	ANGLO	The culture for the given name or null/blank if not known.
surname	String	"Smith"	The non-null/non-blank text of the surname portion of the personal name.
surnameCulture	Culture	ANGLO	The culture for the surname or null/blank if not known.

Table 16 - ParsedName data type (SOAP)

Field Name	*Type*	*Example*	*Notes*
nameText	String	"John Smith"	The non-null/non-blank text of the full personal name.
culture	Culture	ANGLO	The culture for the entire name, or null/blank if not known.

Table 17 - FullName data type (SOAP)

Field Name	*Type*	*Example*	*Notes*
nameText	String	"Smith Brothers"	The non-null/non-blank text of the full organization name.
culture	Culture	ANGLO	The culture for the organization name, or null/blank if not known.

Table 18 - OrganizationName data type (SOAP)

Field Name	*Type*	*Example*	*Notes*
nameText	String	"Madison Dodge"	The non-null/non-blank text of the full name.

Table 19 - GenericName data type (SOAP)

Field Name	*Type*	*Example*	*Notes*
nameListCode	String	"PASSENGERS	Name of the external data source (name list name).
externalID	String	"xyz123"	The text representation of the identifier for the record with which the name is associated in the external system.

Table 20 - External Reference data type (SOAP)

Sample input SOAP – adding a parsed personal name

Sent using PUT with preemptive basic auth header to
http://<host>:<port>/ws/dispatcher/api/soap/NameListPort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services/">
  <soapenv:Header />
  <soapenv:Body>
    <ser:addName>
      <name type="Parsed">
        <givenName>June</givenName>
        <givenNameCulture>KOREAN</givenNameCulture>
        <surname>Park</surname>
        <surnameCulture>KOREAN</surnameCulture>
      </name>
```

```
      <externalRef>
        <nameListCode>PASSENGERS</nameListCode>
        <externalID>JP1111</externalID>
      </externalRef>
      <nameCategory>PERSONAL</nameCategory>
    </ser:addName>
  </soapenv:Body>
</soapenv:Envelope>
```

Sample input SOAP – adding an organization name

Sent using PUT with preemptive basic auth header to
http://<host>:<port>/ws/dispatcher/api/soap/NameListPort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services/">
  <soapenv:Header />
  <soapenv:Body>
    <ser:addName>
      <name type="Organization">
        <nameText>Martin Guitars</nameText>
      </name>
      <externalRef>
        <nameListCode>PASSENGERS</nameListCode>
        <externalID>x12345</externalID>
      </externalRef>
      <nameCategory>ORGANIZATION</nameCategory>
    </ser:addName>
  </soapenv:Body>
</soapenv:Envelope>
```

Sample output SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:common="http://common/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <addNameResponse xmlns="http://services/">
      <nameID xmlns="http://common/">100409752</nameID>
    </addNameResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

3.2 Remove name

This method removes a single source name from ENS, along with any linking records that map it to search names. The result is that this source name will no longer appear in search results. This method is not used by the ENS GUI client.

The user must have manager privileges on the specified name list.

This method does not directly remove *search* names from the database, since a particular search name that was linked to the removed source name may also be linked to other source names that are still wanted. Instead, a cleanup operation runs at intervals in ENS servers to find and remove orphaned search names (those that are no longer linked to a source name).

The fastest way to remove a complete name list or all name lists is to use the -clear and -clearall options in the NameLoader, rather than web services.

3.2.1 JSON/REST

Method: DELETE
URL: <host>:<port>/ws/dispatcher/api/rest/namelist/<nameListCode>/name/<externalID>
Input: None
Output: JSON, just indicating success.

Sample output JSON:

```
{
   "successful":true
}
```

3.2.2 SOAP

Port NameList
Name removeName
Input Parameters

Name	*Type*	*Req'd*	*Notes*
externalRef	ExternalReference (see Table 20)	yes	Identifies the name within some external system by specifying a nameListCode (name list name) and external ID. If this is null/blank or identifies a record from an unrecognized data source, then an exception is generated.

Return type Boolean
Notes The return value is true if the name is found to exist for the data source and is removed, and false if the external reference does not identify a known name for the data source.
Other types used ExternalReference (see Table 20)

Sample input SOAP
Sent using DELETE with preemptive basic auth header to
http://<host>:<port>/ws/dispatcher/api/soap/NameListPort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services/">
  <soapenv:Header />
  <soapenv:Body>
    <ser:removeName>
      <ser:nameListCode>PASSENGERS</ser:nameListCode>
      <ser:externalID>763219</ser:externalID>
    </ser:removeName>
```

```
  </soapenv:Body>
</soapenv:Envelope>
```

Sample output SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:common="http://common/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <removeNameResponse xmlns="http://services/">
      <successful xmlns="http://common/">true</successful>
    </removeNameResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

3.3 Get name details

This method gets details of the source name in a particular name list having a particular external ID. The user must have searcher privileges on the specified name list. This method is not used by the ENS GUI client.

3.3.1 JSON/REST

Method: GET
URL: <host>:<port>/ws/dispatcher/api/rest/namelist/<nameListCode>/name/<externalID>
Input: None
Output format: JSON
Outputs: A single JSON object with the following fields:

Field Name	*Type*	*Req'd*	*Example*	*Notes*
externalID	String	yes	"xyz123	External ID of the source name.
nameListCode	String	yes	"PASSENGERS"	Name of the name lists where this source name is found

Sample output JSON:

```
{
   "externalID":"xyz123",
   "nameListCode":"PASSENGERS"
}
```

3.3.2 SOAP

Port NameList
Name getNameDetails
Input Parameters

Name	*Type*	*Req'd*	*Notes*
externalRef	ExternalReference (seeTable 20)	yes	Identifies the name for which the details are being requested. If this is null/blank or the nameListCode (name list name) is unknown, an exception is generated. If the nameListCode is valid but the external id does not match any name record, a null/blank value is returned.

Return type SourceName (see Table 21)

Field Name	*Type*	*Example*	*Notes*
externalID	String	"xyz123"	The text representation of the identifier for the record with which the name is associated in the external system.
sourceNameID	String	"10004025"	Unique internal id of the source name within the database
nameListCode	String	"PASSENGERS	Name of the external data source (name list name).
rawGivenName	String	John	The given name portion of the original name.
rawSurname	String	Smith	The surname portion of an original personal name, or the complete original organization name.

Table 21 - Source Name data type (SOAP)

Sample input SOAP

Sent using POST with preemptive basic auth header to
http://<host>:<port>/ws/dispatcher/api/soap/NameListPort

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://services/">
  <soapenv:Header />
  <soapenv:Body>
    <ser:getNameDetails>
      <ser:nameListCode>PASSENGERS</ser:nameListCode>
      <ser:externalID>469626</ser:externalID>
    </ser:getNameDetails>
  </soapenv:Body>
</soapenv:Envelope>
```

Sample output SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:common="http://common/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getNameDetailsResponse xmlns="http://services/">
      <nameDetail xmlns="http://common/">
        <externalID>469626</externalID>
        <sourceNameID>100405905</sourceNameID>
        <nameListCode>PASSENGERS</nameListCode>
        <rawGivenName>ALI</rawGivenName>
        <rawSurname>AHMAD</rawSurname>
      </nameDetail>
    </getNameDetailsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

4 Additional Notes

4.1 Enterprise Name Search Requires UTF-8 Encoding

If you are working with data that includes non-ASCII characters with the SOAP Web services, make sure that the locale environment variables are set to use UTF-8 encoding. REST Web services are not affected. If your systems do not default to UTF-8 encoding, do the following:

4.1.1 Linux/UNIX environment:

Find or install the UTF-8 version of the locale for your language and country. This locale must be installed on all machines used in the ENS cell. Use the `locale -a` command to see a list of installed locales on the machine. For example:

```
jksmith@din:~$ locale -a
C
C.UTF-8
en_AG
en_AG.utf8
en_AU.utf8
en_GB.utf8
en_US.utf8
ja_JP.utf8
```

4.1.2 POSIX

Find or install the UTF-8 version of the locale for your language and country. For example, if you are in the US, you can select "en_US.UTF-8" from the list.

Make sure that the shell environment that starts ENS WebSphere processes (runs the `<ENS Install Home>/bin/start-<profile name>.sh` scripts) has the following environment set to the selected UTF-8 locale BEFORE starting ENS.

Example for Bourne shell (sh):

```
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
```

Example for an sh shell:

```
LC_ALL=en_US.UTF-8
LANG=en_US.UTF-8
export LC_ALL
export LANG
```

4.1.3 Windows environment:

Windows does not support making UTF-8 the system-wide or user-wide default character encoding and applications must be configured to use UTF-8 on a case-by-case basis. For ENS, you must configure the Java JVM that runs WebSphere and ENS to use UTF-8. IBM J9 Java JVM as bundled with WebSphere has an environment variable for specifying JVM options, called "`IBM_JAVA_OPTIONS`". This variable can be set on Windows in multiple ways.

From the Windows command prompt:

1. Set the environment variable: `IBM_JAVA_OPTIONS=-Dfile.encoding=UTF-8`
2. Run the start-(ENS Profile Name).bat script.

Scoped to a specific Windows user:

This setting will propagate to all IBM J9 based Java applications run by the user.

1. Click the Environment Variables... button in the Advanced tab of the system Control Panel. On Windows 2008 Server go to Start > Control Panel > Advanced system settings to be taken directly to the Advanced tab of the System control panel). The Environment Variables dialog appears.

2. In the User variable for <current user name> section, click the New... button.

3. The New User Variable box appears.
   In the "Variable name" field, enter `IBM_JAVA_OPTIONS`.
   In the "Variable value" field, enter `-Dfile.encoding=UTF-8`.

4. OK in the box. Then click OK in both the Environment Variables dialog and in the System control panel.

5. Log out of Windows and then log back in. Any Java application started by this user using the IBM J9 JVM will now use the UTF-8 encoding by default.

Scoped to an entire Windows system

This setting will propagate to all IBM J9 based Java applications started on the Windows server.

1. Click the Environment Variables... button in the Advanced tab of the system Control Panel. On Windows 2008 Server go to Start > Control Panel > Advanced system settings to be taken directly to the Advanced tab of the System control panel). The Environment Variables dialog appears.

2. In the System variables section, click the New... button.

3. The New System Variable box appears.
   In the "Variable name" field, enter `IBM_JAVA_OPTIONS`.
   In the "Variable value" field, enter `-Dfile.encoding=UTF-8`.

4. OK in the box. Then click OK in both the Environment Variables dialog and in the System control panel.

5. Reboot the system. Any Java application started on this system that uses the IBM J9 JVM will now use the UTF-8 encoding by default.

Note: The above `IBM_JAVA_OPTIONS` environment variable solution works on any other platform that uses the IBM J9 JVM to run WebSphere, including Linux and AIX. It does not work for Solaris because IBM does not use IBM J9 JVM with that operating system.

4.2 Providing Credentials with Basic Authentication

The simplest way to provide credentials for ENS web services is by including a standard basic authentication request header in every HTTP request. Such a header contains a userid and password, base 64 encoded[2], looking like this:

```
Authorization: Basic ZW5zYWRtaW46ZW5zcHdk
```

The disadvantage of this approach is that every web service request involves the overhead of credential authentication and session creation.

4.3 Using ENS Web Services in a Session

Alternatively, you can establish a session with a special login request, use session cookies in your web service requests, and then end the session with a logout request. If you do this, the overhead of credential authentication and session creation only needs to happen once, not in every web service request. This helps performance.

4.3.1 Logging in

To start a session, send a POST request to https://host:port/j_security_check with a post body containing your credentials like this:

```
j_username=myusername&j_password=mypassword&action=Login
```

Assuming valid credentials, this request logs you in with a session. The response will include an ensSecurityCookie cookie that you can then include in subsequent web service requests.

Each such request happens in the context of your session. The request does not need an authentication header or per-request authentication.

---

[2] Since basic authorization credentials are only encoded and not encrypted in any way, SSL and HTTPS are recommended if the network is not secure. You can configure your ENS installation to require HTTPS by setting https.enabled to true in the ENS_CONFIG table in your ENS database.

4.3.2 Logging out

To end your session, make a POST request to `/ens/dispatcherLogoutAction`, `including` the ensSecurityCookie.

5 Appendix – The EnterpriseNameSearcher.WSDL File

In a running ENS server, this file is available at `http://<host>:<port>/ws/resources/wsdl/EnterpriseNameSearcher.wsdl`

```
<!--
  Description: Enterprise Name Searcher Webservices Definition Document

  IBM Confidential
  OCO Source Materials
  5724-Q20
  Copyright IBM Corp. 2013, 2016
  The source code for this program is not published or otherwise divested of
  its trade secrets, irrespective of what has been deposited with the
  U.S. Copyright Office.
  -->

<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:services="http://services/" xmlns:common="http://common/"
xmlns:ens="http://ens.gnr.ibm.com/EnterpriseNameSearcher/" targetNamespace="http://ens.gnr.ibm.com/EnterpriseNameSearcher/">

  <wsdl:types>
    <xsd:schema targetNamespace="http://common/">
      <xsd:simpleType name="ScriptType">
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="NOTSPECIFIEDSCRIPT"/>
          <xsd:enumeration value="NOSCRIPT"/>
          <xsd:enumeration value="CHINESESCRIPT"/>
          <xsd:enumeration value="JAPANESESCRIPT"/>
          <xsd:enumeration value="INDIANSCRIPT"/>
          <xsd:enumeration value="CYRILLICSCRIPT"/>
          <xsd:enumeration value="LATINSCRIPT"/>
          <xsd:enumeration value="KOREANSCRIPT"/>
          <xsd:enumeration value="ARABICSCRIPT"/>
          <xsd:enumeration value="GREEKSCRIPT"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="NameCategory">
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="PERSONAL"/>
          <xsd:enumeration value="ORGANIZATION"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="Culture">
        <xsd:restriction base="xsd:string">
```

```
<xsd:enumeration value="AFGHANI"/>
<xsd:enumeration value="AMBIGUOUS"/>
<xsd:enumeration value="ANGLO"/>
<xsd:enumeration value="ARABIC"/>
<xsd:enumeration value="CHINESE"/>
<xsd:enumeration value="HISPANIC"/>
<xsd:enumeration value="KOREAN"/>
<xsd:enumeration value="RUSSIAN"/>
<xsd:enumeration value="FRENCH"/>
<xsd:enumeration value="GERMAN"/>
<xsd:enumeration value="THAI"/>
<xsd:enumeration value="INDONESIAN"/>
<xsd:enumeration value="YORUBAN"/>
<xsd:enumeration value="FARSI"/>
<xsd:enumeration value="PAKISTANI"/>
<xsd:enumeration value="INDIAN"/>
<xsd:enumeration value="JAPANESE"/>
<xsd:enumeration value="AFGHANI"/>
<xsd:enumeration value="VIETNAMESE"/>
<xsd:enumeration value="POLISH"/>
<xsd:enumeration value="PORTUGUESE"/>
<xsd:enumeration value="TURKISH"/>
<xsd:enumeration value="UNUSED21"/>
<xsd:enumeration value="UNUSED22"/>
<xsd:enumeration value="UNUSED23"/>
<xsd:enumeration value="UNUSED24"/>
<xsd:enumeration value="UNUSED25"/>
<xsd:enumeration value="UNUSED26"/>
<xsd:enumeration value="UNUSED27"/>
<xsd:enumeration value="UNUSED28"/>
<xsd:enumeration value="UNUSED29"/>
<xsd:enumeration value="UNUSED30"/>
<xsd:enumeration value="UNUSED31"/>
<xsd:enumeration value="UNUSED32"/>
<xsd:enumeration value="UNUSED33"/>
<xsd:enumeration value="UNUSED34"/>
<xsd:enumeration value="UNUSED35"/>
<xsd:enumeration value="UNUSED36"/>
<xsd:enumeration value="UNUSED37"/>
<xsd:enumeration value="SOUTHWESTASIAN"/>
<xsd:enumeration value="EUROPEAN"/>
<xsd:enumeration value="HAN"/>
```

```
      <xsd:enumeration value="POLISH"/>
      <xsd:enumeration value="PORTUGUESE"/>
      <xsd:enumeration value="TURKISH"/>
      <xsd:enumeration value="CUSTOM1"/>
      <xsd:enumeration value="CUSTOM2"/>
      <xsd:enumeration value="CUSTOM3"/>
      <xsd:enumeration value="CUSTOM4"/>
      <xsd:enumeration value="CUSTOM5"/>
      <xsd:enumeration value="CUSTOM6"/>
      <xsd:enumeration value="CUSTOM7"/>
      <xsd:enumeration value="CUSTOM8"/>
      <xsd:enumeration value="CUSTOM9"/>
      <xsd:enumeration value="CUSTOM10"/>
      <xsd:enumeration value="CUSTOM11"/>
      <xsd:enumeration value="CUSTOM12"/>
      <xsd:enumeration value="CUSTOM13"/>
      <xsd:enumeration value="CUSTOM14"/>
      <xsd:enumeration value="CUSTOM15"/>
      <xsd:enumeration value="CUSTOM16"/>
      <xsd:enumeration value="CUSTOM17"/>
      <xsd:enumeration value="CUSTOM18"/>
      <xsd:enumeration value="CUSTOM19"/>
      <xsd:enumeration value="CUSTOM20"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="ExternalReference">
    <xsd:sequence>
      <xsd:element name="nameListCode" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
      <xsd:element name="externalID" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="ExternalReferenceDetail" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="common:ExternalReference">
        <xsd:sequence>
          <xsd:element name="searchNameID" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ExternalReferenceDetailPersonal">
    <xsd:complexContent>
```

```
      <xsd:extension base="common:ExternalReferenceDetail">
        <xsd:sequence>
          <xsd:element name="rawSurname" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
          <xsd:element name="rawGivenName" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
          <xsd:element name="givenNameCulture" type="common:Culture" minOccurs="1" maxOccurs="1" nillable="false"/>
          <xsd:element name="surnameCulture" type="common:Culture" minOccurs="1" maxOccurs="1" nillable="false"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ExternalReferenceDetailOrganization">
    <xsd:complexContent>
      <xsd:extension base="common:ExternalReferenceDetail">
        <xsd:sequence>
          <xsd:element name="rawOrganizationName" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
          <xsd:element name="organizationNameCulture" type="common:Culture" minOccurs="1" maxOccurs="1" nillable="false"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Name" abstract="true">
    <xsd:sequence>
      <xsd:element name="scriptType" type="common:ScriptType" minOccurs="0" maxOccurs="1" nillable="true"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="PersonalName" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="common:Name"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ParsedName">
    <xsd:complexContent>
      <xsd:extension base="common:PersonalName">
        <xsd:sequence>
          <xsd:element name="givenName" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
          <xsd:element name="givenNameCulture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
          <xsd:element name="surname" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
          <xsd:element name="surnameCulture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

```
<xsd:complexType name="FullName">
  <xsd:complexContent>
    <xsd:extension base="common:PersonalName">
      <xsd:sequence>
        <xsd:element name="nameText" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
        <xsd:element name="culture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="OrganizationName">
  <xsd:complexContent>
    <xsd:extension base="common:Name">
      <xsd:sequence>
        <xsd:element name="nameText" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
        <xsd:element name="culture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="GenericName">
  <xsd:complexContent>
    <xsd:extension base="common:Name">
      <xsd:sequence>
        <xsd:element name="nameText" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
        <xsd:element name="culture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="NameListDetail">
  <xsd:sequence>
    <xsd:element name="nameListCode" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
    <xsd:element name="description" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CultureDetail">
  <xsd:sequence>
    <xsd:element name="code" type="xsd:int" minOccurs="1" maxOccurs="1" nillable="false"/>
    <xsd:element name="enumValue" type="common:Culture" minOccurs="1" maxOccurs="1" nillable="false"/>
    <xsd:element name="category" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="false"/>
    <xsd:element name="displayName" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
```

```
      <xsd:element name="defaultDisplayName" type="xsd:boolean" minOccurs="0" maxOccurs="1" nillable="false"/>
      <xsd:element name="defined" type="xsd:boolean" minOccurs="0" maxOccurs="1" nillable="false"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="AnalyzedName">
    <xsd:sequence>
      <xsd:element name="categories" type="common:NameCategory" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
      <xsd:element name="scriptType" type="common:ScriptType" minOccurs="0" maxOccurs="1" nillable="true"/>
      <xsd:element name="name" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
      <xsd:element name="nameCulture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
      <xsd:element name="nameCultureSet" type="common:Culture" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
      <xsd:element name="givenName" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
      <xsd:element name="originalGivenName" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
      <xsd:element name="givenNameCulture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
      <xsd:element name="givenNameCultureSet" type="common:Culture" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
      <xsd:element name="surname" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
      <xsd:element name="originalSurname" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
      <xsd:element name="surnameCulture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
      <xsd:element name="surnameCultureSet" type="common:Culture" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
      <xsd:element name="alternate" type="xsd:boolean" minOccurs="0" maxOccurs="1" nillable="false"/>
      <xsd:element name="confidence" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="false"/>
      <xsd:element name="qualifiers" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
      <xsd:element name="titles" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="SourceName">
    <xsd:sequence>
      <xsd:element name="externalID" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
      <xsd:element name="sourceNameID" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
      <xsd:element name="nameListCode" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
      <xsd:element name="rawGivenName" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
      <xsd:element name="rawSurname"   type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="ExternalReferences">
    <xsd:sequence>
      <xsd:element name="nameListCode" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
      <xsd:element name="count" type="xsd:long" minOccurs="1" maxOccurs="1" nillable="true"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="SearchNameResult" abstract="true">
    <xsd:sequence>
```

```
            <xsd:element name="searchNameID" type="xsd:long" minOccurs="1" maxOccurs="1" nillable="false"/>
            <xsd:element name="nameCategory" type="common:NameCategory" minOccurs="0" maxOccurs="1" nillable="false"/>
            <xsd:element name="score" type="xsd:int" minOccurs="1" maxOccurs="1" nillable="false"/>
            <xsd:element name="regularized" type="xsd:boolean" minOccurs="1" maxOccurs="1" nillable="false"/>
            <xsd:element name="externalReferences" type="common:ExternalReferences" minOccurs="1" maxOccurs="unbounded"
    nillable="false"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="SearchNameOrganization">
          <xsd:complexContent>
            <xsd:extension base="common:SearchNameResult">
              <xsd:sequence>
                <xsd:element name="rawOrganizationName" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
                <xsd:element name="organizationName" type="xsd:string" minOccurs="1" maxOccurs="1" nillable="false"/>
                <xsd:element name="organizationNameCulture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
              </xsd:sequence>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="SearchNamePersonal">
          <xsd:complexContent>
            <xsd:extension base="common:SearchNameResult">
              <xsd:sequence>
                <xsd:element name="rawGivenName" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
                <xsd:element name="rawSurname" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
                <xsd:element name="originalGivenName" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
                <xsd:element name="originalSurname" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
                <xsd:element name="scriptType" type="common:ScriptType" minOccurs="0" maxOccurs="1" nillable="true"/>
                <xsd:element name="surname" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
                <xsd:element name="surnameCulture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
                <xsd:element name="surnameScore" type="xsd:int" minOccurs="1" maxOccurs="1" nillable="false"/>
                <xsd:element name="givenName" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
                <xsd:element name="givenNameCulture" type="common:Culture" minOccurs="0" maxOccurs="1" nillable="true"/>
                <xsd:element name="givenNameScore" type="xsd:int" minOccurs="1" maxOccurs="1" nillable="false"/>
                <xsd:element name="alternate" type="xsd:boolean" minOccurs="1" maxOccurs="1" nillable="false"/>
              </xsd:sequence>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
        <xsd:complexType name="SearchResult">
          <xsd:sequence>
            <xsd:element name="analyzedNames" type="common:AnalyzedName" minOccurs="1" maxOccurs="3" nillable="false"/>
```

```
            <xsd:element name="nameResults" type="common:SearchNameResult" minOccurs="0" maxOccurs="unbounded" nillable="false"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:schema>
      <xsd:schema targetNamespace="http://services/">
        <xsd:element name="removeName">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="externalRef" type="common:ExternalReference" minOccurs="1" maxOccurs="1" nillable="false"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="removeNameResponse">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="successful" type="xsd:boolean" minOccurs="1" maxOccurs="1" nillable="false"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="getNameDetails">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="externalRef" type="common:ExternalReference" minOccurs="1" maxOccurs="1" nillable="false"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="getNameDetailsResponse">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="nameDetail" type="common:SourceName" minOccurs="0" maxOccurs="1" nillable="false"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="addName">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="nameCategory" type="common:NameCategory" minOccurs="1" maxOccurs="1" nillable="false"/>
              <xsd:element name="name" type="common:Name" minOccurs="1" maxOccurs="1" nillable="false"/>
              <xsd:element name="externalRef" type="common:ExternalReference" minOccurs="1" maxOccurs="1" nillable="false"/>
              <xsd:element name="includeAlternateParses" type="xsd:boolean" minOccurs="0" maxOccurs="1" nillable="true"
    default="false"/>
            </xsd:sequence>
```

```
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="addNameResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="nameID" type="xsd:long" minOccurs="1" maxOccurs="1" nillable="false"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="getSearchNameLists"/>
  <xsd:element name="getSearchNameListsResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="nameLists" type="common:NameListDetail" minOccurs="0" maxOccurs="unbounded" nillable="false"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="getSearchStrategyCodes"/>
  <xsd:element name="getSearchStrategyCodesResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="codes" type="xsd:string" minOccurs="0" maxOccurs="unbounded" nillable="false"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="getCultures"/>
  <xsd:element name="getCulturesResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="cultures" type="common:CultureDetail" minOccurs="0" maxOccurs="unbounded" nillable="false"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="searchName">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="common:Name" minOccurs="1" maxOccurs="1" nillable="false"/>
        <xsd:element name="nameListCodes" type="xsd:string" minOccurs="0" maxOccurs="unbounded" nillable="false"/>
        <xsd:element name="strategyCode" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <xsd:element name="searchCategories" type="common:NameCategory" minOccurs="0" maxOccurs="unbounded" nillable="false"/>
        <xsd:element name="maxResults" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="false" default="-1"/>
        <xsd:element name="minScore" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="false" default="-1"/>
```

```
            <xsd:element name="includeAlternateParses" type="xsd:boolean" minOccurs="0" maxOccurs="1" nillable="true"
default="false"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="searchNameResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="searchResult" type="common:SearchResult" minOccurs="1" maxOccurs="1" nillable="false"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="getExternalReferences">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="searchNameIDs" type="xsd:long" minOccurs="1" maxOccurs="unbounded" nillable="false"/>
            <xsd:element name="rawGivenName" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="false"/>
            <xsd:element name="rawSurname" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="false"/>
            <xsd:element name="rawOrganizationName" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="false"/>
            <xsd:element name="nameListCodes" type="xsd:string" minOccurs="0" maxOccurs="unbounded" nillable="false"/>
            <xsd:element name="maxResults" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="false"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="getExternalReferencesResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="externalRef" type="common:ExternalReferenceDetail" minOccurs="0" maxOccurs="unbounded"
nillable="false"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>

  <wsdl:message name="addNameRequest">
    <wsdl:part element="services:addName" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="addNameResponse">
    <wsdl:part element="services:addNameResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="removeNameRequest">
```

```
    <wsdl:part element="services:removeName" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="removeNameResponse">
    <wsdl:part element="services:removeNameResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getNameDetailsRequest">
    <wsdl:part element="services:getNameDetails" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getNameDetailsResponse">
    <wsdl:part element="services:getNameDetailsResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getSearchNameListsRequest">
    <wsdl:part element="services:getSearchNameLists" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getSearchNameListsResponse">
    <wsdl:part element="services:getSearchNameListsResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getSearchStrategyCodesRequest">
    <wsdl:part element="services:getSearchStrategyCodes" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getSearchStrategyCodesResponse">
    <wsdl:part element="services:getSearchStrategyCodesResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getCulturesRequest">
    <wsdl:part element="services:getCultures" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getCulturesResponse">
    <wsdl:part element="services:getCulturesResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="searchNameRequest">
    <wsdl:part element="services:searchName" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="searchNameResponse">
    <wsdl:part element="services:searchNameResponse" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getExternalReferencesRequest">
    <wsdl:part element="services:getExternalReferences" name="parameters"/>
  </wsdl:message>
  <wsdl:message name="getExternalReferencesResponse">
    <wsdl:part element="services:getExternalReferencesResponse" name="parameters"/>
  </wsdl:message>
```

```
<wsdl:portType name="NameList">
  <wsdl:operation name="addName">
    <wsdl:input message="ens:addNameRequest"/>
    <wsdl:output message="ens:addNameResponse"/>
  </wsdl:operation>
  <wsdl:operation name="removeName">
    <wsdl:input message="ens:removeNameRequest"/>
    <wsdl:output message="ens:removeNameResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getNameDetails">
    <wsdl:input message="ens:getNameDetailsRequest"/>
    <wsdl:output message="ens:getNameDetailsResponse"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="Search">
  <wsdl:operation name="getSearchNameLists">
    <wsdl:input message="ens:getSearchNameListsRequest"/>
    <wsdl:output message="ens:getSearchNameListsResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getSearchStrategyCodes">
    <wsdl:input message="ens:getSearchStrategyCodesRequest"/>
    <wsdl:output message="ens:getSearchStrategyCodesResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getCultures">
    <wsdl:input message="ens:getCulturesRequest"/>
    <wsdl:output message="ens:getCulturesResponse"/>
  </wsdl:operation>
  <wsdl:operation name="searchName">
    <wsdl:input message="ens:searchNameRequest"/>
    <wsdl:output message="ens:searchNameResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getExternalReferences">
    <wsdl:input message="ens:getExternalReferencesRequest"/>
    <wsdl:output message="ens:getExternalReferencesResponse"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="NameListBinding" type="ens:NameList">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="addName">
    <soap:operation soapAction="http://ens.gnr.ibm.com/EnterpriseNameSearcher/addName/"/>
```

```
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="removeName">
      <soap:operation soapAction="http://ens.gnr.ibm.com/EnterpriseNameSearcher/removeName/"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getNameDetails">
      <soap:operation soapAction="http://ens.gnr.ibm.com/EnterpriseNameSearcher/getNameDetails/"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:binding name="SearchBinding" type="ens:Search">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getSearchNameLists">
      <soap:operation soapAction="http://ens.gnr.ibm.com/EnterpriseNameSearcher/getSearchNameLists/"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getSearchStrategyCodes">
      <soap:operation soapAction="http://ens.gnr.ibm.com/EnterpriseNameSearcher/getSearchStrategyCodes/"/>
      <wsdl:input>
        <soap:body use="literal"/>
```

```
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getCultures">
      <soap:operation soapAction="http://ens.gnr.ibm.com/EnterpriseNameSearcher/getCultures/"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="searchName">
      <soap:operation soapAction="http://ens.gnr.ibm.com/EnterpriseNameSearcher/searchName/"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getExternalReferences">
      <soap:operation soapAction="http://ens.gnr.ibm.com/EnterpriseNameSearcher/getExternalReferences/"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="EnterpriseNameSearcher">
    <wsdl:port binding="ens:NameListBinding" name="NameListPort">
      <soap:address
location="${request.scheme}://${request.serverName}:${request.serverPort}${request.contextPath}/dispatcher/api/soap/NameListPort"/
>
    </wsdl:port>
    <wsdl:port binding="ens:SearchBinding" name="SearchPort">
```

```
      <soap:address
location="${request.scheme}://${request.serverName}:${request.serverPort}${request.contextPath}/dispatcher/api/soap/SearchPort"/>
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

When deployed, the above markers ${request.scheme}://${request.serverName}:${request.serverPort}${request.contextPath} are replaced with values appropriate to the server where deployed, leading to locations like: http://myhost:14510/ws/dispatcher/....