Google Search Appliance

Google OneBox for Enterprise Developer's Guide

Google Search Appliance software version 7.2 and later

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
www.google.com

GSA-OBDEV_200.01
March 2015

Contents

Google OneBox for Enterprise Developer's Guide 4
Introduction 4
Summary of Steps to Define and Deploy a OneBox Module 6
Planning 7
Defining a OneBox Module in the Admin Console 8
Defining a OneBox Module Using an XML Configuration File 9
    Creating a Trigger 9
    Specifying and Calling a Provider 12
    Receiving a Provider's Response 13
    Transforming XML to HTML 14
    Formatting the Results 14
    Checking the Visual Layout 16
    Adding a Secure Connection to the Provider 16
    Defining User-Specific Results and Access Control 17
    Passing Optional Contextual Data 18
    Handling Errors and Lookup Failures 18
    Tips 18
Testing with the OneBox Simulator 19
    Downloading the OneBox Simulator 19
    Creating an XSLT File 19
    Editing customer-onebox.xsl to Call Your XSL File 20
    Merging a OneBox Module Output with the Search Results 20
    Applying the Overall Stylesheet to the Merged XML Output 21
OneBox Module Definition XML Reference 21
    Module Definition Schema 22
    Call Parameters 25
OneBox Results Schema Reference 26
    Results Definition Schema 26

Index 29

Google OneBox for Enterprise Developer's Guide

A OneBox module provides real-time access to data from an external source, or a collection on a search appliance.

This document describes how to define a OneBox module and how to use the OneBox Simulator (http://google-developers.appspot.com/search-appliance/download/downloadsdk), which enables you to test your OneBox code before putting your code on a search appliance.

Before starting development, refer to *Google OneBox for Enterprise Design Principles*, which provides information about how to design a OneBox module. Google also provides the Custom KeyMatch OneBox (http://code.google.com/p/custom-keymatch-onebox/).

For information about Onebox module limitations, see Specifications and Usage Limits.

Introduction

Google OneBox gives users access to real-time data through a simple, fast, and easy to configure search interface.

A OneBox defines a search type, the keyword that invokes the search, and the way that a search appliance obtains and returns information after a user invokes a search. You can define any number of OneBox modules, and a user search page can display results from multiple OneBox modules.

A OneBox module consists of the following components:

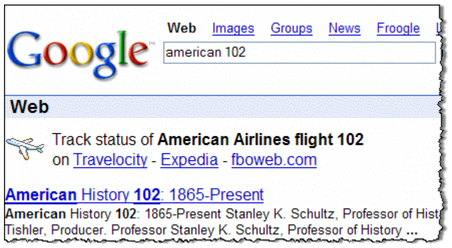
- **General Information**—Defines the module. Includes a name for reference and an internal description.
- **Trigger**—Specifies the keywords or query type that cause a module to request data from a search appliance.
- **Provider**—Gives the location of a search appliance collection or an external provider that is responsible for resolving the query. Defines access control parameters that indicate whether the module returns public or secure (user-specific) information.
- **Security**—Optionally specifies whether a search appliance securely authenticates itself, the end user, or both to the external provider, and passes authentication information if necessary.
- **Results Template**—Defines an XSLT template that translates the returned data into HTML.

You define this information on a search appliance by entering the information in the Admin Console or by specifying the information in an XML configuration file and importing it. For information on XML schema, see the “OneBox Module Definition XML Reference” on page 21.

Google.com uses the OneBox extensively to provide users with access to information in different content repositories, such as Google News, Google Images, and Google Book Search. On Google.com, the use of OneBox also provides real-time data such as weather, flight tracking, package tracking, and movie times. Google.com provides a single text entry box instead of a complex interface for specifying information types.

The name “OneBox” refers to the search box that provides access to information from many sources. OneBox can also refer to the formatted output that appears in response to specific query keywords.

The following figure shows the OneBox that displays on Google.com. Here, a user searches for `american 102`:



Track status of **American Airlines flight 102** on Travelocity - Expedia - fboweb.com

American History **102**: 1865-Present
American History **102**: 1865-Present Stanley K. Schultz, Professor of Hist
Tishler, Producer. Professor Stanley K. Schultz, Professor of History ...

In this example, the search interprets a user’s entry for an airline name and a number as a request for flight information. The OneBox results appear above other search results and are visually distinguished from the other results.

Google OneBox for Enterprise brings the power and simplicity of search to provide fast access to information in an enterprise network. Using Google OneBox for Enterprise, you can create Onebox modules that provide users with real-time business data from enterprise resource planning (ERP) systems, customer relationship management (CRM) applications, or business intelligence analysis.

For example, you can create OneBox modules for the following types of company information:

- Employee telephone numbers
- Organizational chart
- Customer contacts
- Product part numbers
- Inventory information
- Vendor information
- Sales figures per region

You can create OneBox modules for the following types of educational information:

- Course descriptions
- Faculty contact information
- Department addresses
- Major requirements

A search appliance can resolve OneBox queries using an internal or external provider. You specify provider information in an XML file that the search appliance interprets to determine how to resolve the OneBox query. The search appliance returns internal OneBox queries directly to the user interface. A search appliance can also resolve queries externally, through calls to external systems.

Google implements the OneBox request/response technology in XML, which enables you to make changes easily to the OneBox functionality. You can add OneBox functions by defining and implementing OneBox modules.

Note: A OneBox module cannot use a rewritten query term expanded by the query expansion feature. A OneBox module always uses the original query term before a query expansion occurs. For more information on query expansion, see "Widening Searches" in *Creating the Search Experience*.

Summary of Steps to Define and Deploy a OneBox Module

Google enables you to define a OneBox module that you can trigger by a keyword or a regular expression, or instead of triggering, the OneBox can appear on every search query. A OneBox module can either search a collection or access a URL for a site that returns XML results. You can define a OneBox from the search appliance's Admin Console. This section introduces the procedures described in this document.

To define and deploy a OneBox module:

1. Define what you want the OneBox module to do, what the search appliance needs to do when it invokes the OneBox module, and how you want the OneBox module results to appear. Having a clear definition of what you are trying to achieve is essential to a successful implementation. *Google OneBox for Enterprise Design Principles* provides information you can use to design a OneBox module.

2. Develop a provider, which is the source of information. A search appliance contacts a provider to deliver relevant information to a user. The OneBox module can call an internal or external provider as follows:

   - **Internal provider**—The OneBox module performs a full-text search across the contents of a collection and returns the results in a OneBox user interface.
   - **External provider**—The OneBox module calls a URL to get data from an external application that returns information as XML.

3. Create the OneBox module in the Admin Console from **Content Sources > OneBox Modules**. You can either use the Admin Console to specify all the parameters of the OneBox module or indicate the name of the XML configuration file that contains provider information.

4. Enable the OneBox module from the Admin Console from **Search > Search Features > Front Ends** by adding the module to one or more front ends.

5. Use the OneBox Simulator to test your OneBox module. For more information, see "Testing with the OneBox Simulator" on page 19.

6. Conduct test searches to display your OneBox module. If you specified a front end other than the default, ensure that the `&client=` parameter in the search URL contains the name of your front end.

7. You can view search log results for your OneBox module from **Content Sources > OneBox Modules**. Click **View Logs** for your OneBox module entry in the **Current OneBox Modules** list.

8. When done testing, if your OneBox module uses a trigger keyword or a regular expression, inform users of the keyword.

Planning

When defining a OneBox module, you need to work with the Admin Console for configuration and may need additional software to facilitate your use of OneBox modules:

- First time use, define the OneBox from the Admin Console. For more information, see “Defining a OneBox Module in the Admin Console” on page 8.
- Using an internal provider, you can define your OneBox from the Admin Console or with an XML configuration file. For more information, see “Using an Internal Provider” on page 12.
- Using an external provider:
  - You need to create an XML configuration file.
  - You need to ensure that the external provider has a programmatic way of formatting the data and creating XML data for the data that appears in the search results. The output XML file must conform to the OneBox Results Schema (see “OneBox Results Schema Reference” on page 26).
  - You need to determine the access control that a search user may need to access content. The choices for access control are no authentication, HTTP Basic authentication, Windows NT LAN Manager (NTLM) HTTP, Lightweight Directory Access Protocol (LDAP), or Single Sign On (SSO). For more information, see “Using an External Provider” on page 12.

Working with an XML configuration file:

- Ensure that you have a text editor that can output text files in UTF-8 format.
- Ensure that the XML configuration file conforms to the “Module Definition Schema” on page 22.
- Download the Google OneBox Simulator (http://google-developers.appspot.com/search-appliance/download/downloadsdk) to test your XML configuration file. For more information, see “Testing with the OneBox Simulator” on page 19. You can use any available XSLT tool with the simulator such as XML Spy, OxygenXML, tools from the major software vendors, or open source tools. The Saxon XSLT processor is available in open source at http://saxon.sourceforge.net/. Saxon and many other XSLT processors require that you have Java installed on your computer. Oracle provides J2SE at http://www.oracle.com/technetwork/java/archive-139210.html. Google has tested the simulator with J2SE version 5.5.

Defining a OneBox Module in the Admin Console

To define a OneBox module in the Admin Console:

1. In **Content Sources > OneBox Modules**, enter a name in the **OneBox Name** field and click **Create Module Definition**.

2. Enter a **Description**.

3. In the **Trigger** section, click one of the following choices:

   - **Always Trigger**—Cause the OneBox to appear on every search query. For more information, see “Triggering on Every Query” on page 10.
   - **Keywords**—Specify one or more words that a can user can enter that cause a OneBox to appear. Separate multiple keywords with a pipe symbol. For example, `phone|contact|info`. For more information, see “Triggering in Response to Specific Keywords” on page 10.
   - **Regular Expression**—Specify a Perl regular expression (see http://perldoc.perl.org/perlre.html), such as `phone (.*)` to match the phone keyword and any value a user enters to search for a phone number. For more information, see “Triggering When the Query Matches a Regular Expression” on page 11.

4. In the **Provider** section, specify the name of a collection or an external provider. For more information, see “Using an Internal Provider” on page 12. In version 6.0 and later, you can also specify **User Results**. For information, see “Enabling Authentication for User Results” in *Creating the Search Experience*.

   If you choose an external provider, you need a programmatic way of formatting an XML display object that can appear in the search results. For more information on working with an external provider, see “Using an External Provider” on page 12.

5. Specify the type of authentication users require to access content:

   - No authentication
   - Basic HTTP authentication (a user name and password for access to individual documents)
   - LDAP for access to multiple documents
   - SSO for single sign-on access across systems

6. If the search appliance and/or the external provider require username and password credentials, specify the credentials.

7. Click **Save**. The Admin Console displays the OneBox Modules screen

8. To edit the OneBox Style Template to format a OneBox for how the output appears in search results, click **Edit** for the OneBox module, and at the end of the edit screen, you can click **Edit XSL**. For more information, see “Receiving a Provider’s Response” on page 13.

After the search appliance crawls the content in the collection, you can test your OneBox from search.

Defining a OneBox Module Using an XML Configuration File

To define a OneBox from an XML configuration file:

1. Create a new XML file.

2. Define the top level `<onebox>` element (see “onebox Element” on page 22) to indicate whether the module uses an internal provider or external provider.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
     <GoogleEnterpriseSources xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <onebox type="external">
     ...
   ```

3. Give the module a `<name>` (see the element “name” on page 23) and `<description>` (see the element “description” on page 24). The name references the OneBox module on the outbound call from the search appliance. The description explains the module’s functionality to search appliance administrators and appears in the Admin Console.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <GoogleEnterpriseSources xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <onebox type="external">
       <name>directory_example</name>
       <description>This sample OneBox queries a phone directory</description>
       ...
     </onebox>
   </GoogleEnterpriseSources>
   ```

Next, create a trigger as described in “Creating a Trigger” on page 9.

Creating a Trigger

A trigger determines which search queries invoke a OneBox module. A trigger can be a keyword, type of phrase, or a regular expression. You specify the module’s keyword in the `trigger` element. The format of the `trigger` element is as follows:

```
<trigger triggerType="when_to_trigger">trigger_word</trigger>
```

You can define a trigger in the XML configuration file or by creating and editing a OneBox in the Admin Console.

The table below lists the `triggerType` attribute values.

Attribute	Description	Example
`null`	Invoke trigger on every query.	`<trigger triggerType="null"></trigger>`
`keyword`	Invoke in response to one or more keywords. Specify keyword as the value of the element.	`<trigger triggerType="keyword">directory</trigger>`
`regex`	Invoke when a query matches a regular expression.	`<trigger triggerType="regex">directory (.*)</trigger>`

The `trigger` element is optional. If you omit a trigger, the OneBox module invokes on every query.

Triggering on Every Query

If you have a high-bandwidth provider and network and query traffic is not a concern, the search appliance can invoke a module on every query. In that case, a provider must determine whether the module should return results to the user. You can achieve the best user experience by returning OneBox results only when they enhance a user's overall search results.

Triggering in Response to Specific Keywords

Our directory example uses a keyword trigger, `directory`. A keyword trigger must be the first word of the user's search query, so that the search appliance invokes a module when the search query has the form `directory Bill Smith` but not when the search query has the form `employee directory`. You can specify multiple keywords by separating each word with a pipe symbol, for example, `directory|phone|contact`. If you want the keyword trigger to invoke on a word other than the first word, use a regular expression.

To define a trigger in the XML configuration file, specify the `<trigger>` element as follows:

```
<onebox type="external">
  <name>directory_example</name>
  <description>This sample OneBox queries a phone directory</description>
  <trigger triggerType="keyword">directory</trigger>
  ...
</onebox>
```

To specify multiple keywords, the `<trigger>` tag could appear as follows:

```
<trigger triggerType="keyword">directory|dir|d:|phone</trigger>
```

This example shows how you can use multiple keywords to provide abbreviated versions of a keyword or an alternative keyword.

Triggering When the Query Matches a Regular Expression

If you use a regular expression trigger, a search appliance compares the search query to the regular expression pattern and invokes the OneBox module if the query and the pattern match. OneBox supports regular expressions as defined by the Perl Compatible Regular Expressions library (PCRE) at http://perldoc.perl.org/perlre.html. Enclose each expression in parentheses. You can separate multiple expressions with a space, for example,

```
<trigger triggerType="regex">(i) (.*) report</trigger>
```

ignores case and matches any word before `report`.

The table below lists common regular expression rules.

Syntax	Description
`(.*)`	Match any character zero or more times; that is, match any word
`(.?)`	Match a single character
`(\d+)`	Match a digit one or more times
`(i)`	Ignore case
`([a-z]+)`	Match lowercase letters one or more times
`(?:` *`values`*`)`	Match specific keyword values

The following are examples of the common regular expression rules:

- The regular expression `status (.*)` matches a query if the user types `status` and a project name.
- The regular expression `distance from (.*) to (.*)` matches a query such as `distance from Paris to Rome`.
- The regular expression `\d+` matches a query such as `123`.
- The regular expression `(?i)([a-z]+) airport(?: (?:status|delays?|conditions))?` matches queries such as the following:
  - `lax airport conditions`
  - `SFO airport delays`
  - `newark airport status`

A regular expression provides great flexibility, but you must ensure that a search appliance rapidly evaluates the expression so as not to degrade performance during searches. Note that the regular expression language used by OneBox triggers is not the same as the Google regular expressions that a search appliance uses for URL patterns.

Specifying and Calling a Provider

A OneBox provider can be internal or external. An internal provider is a collection on a search appliance. An external provider is an application outside a search appliance.

Using an Internal Provider

To use an internal provider, specify the `<collection>` element (see the element “collection” on page 24) and use the name of the collection as defined on a search appliance. The following example specifies an internal provider using the `InternalNews` collection for the `news` trigger:

```
<onebox type="internal">
  <name>news</name>
  <description>This sample OneBox queries an intranet news source</description>
  <trigger triggerType="keyword">news</trigger>
  <collection>InternalNews</collection>
  <resultsTemplate></resultsTemplate>
</onebox>
```

Using an External Provider

To specify an external provider, use the `<providerURL>` element (see the element “providerURL” on page 24) to specify the location of an external provider with the following guidelines:

- For standard providers, use a fully qualified URL such as:

  `http://server.mydomain.com:`*`port`*`/directory/...`

- For secure providers, use a fully qualified secure (HTTPS) URL such as:

  `https://secure.server.mydomain.com:`*`port`*`/directory/...`

The following example specifies `Acme.com` as the external provider for the `directory` trigger:

```
<onebox type="external">
  <name>directory_example</name>
  <description>This sample OneBox queries a phone directory</description>
  <security userAuth="none"/>
  <trigger triggerType="keyword">directory</trigger>
  <providerURL>http://directory.corp.acme.com/cgi-bin/phonebook</providerURL>
  <resultsTemplate>{xslt template} <resultsTemplate/>
</onebox>
```

Use of OneBox with an external provider starts with a search request that matches the trigger. In response, a search appliance issues a standard HTTP GET command to the provider. The request from a search appliance to the provider is a URL that combines the provider host and path with a set of name-value pairs. The name-value pairs start with a question mark and use an ampersand (&) characters to separate the input parameters that are sent to the provider.

The following example shows HTTP GET commands that the search appliance constructs and sends to the example Acme.com external provider:

```
GET http://directory.corp.acme.com/cgi-bin/phonebook?
apiMaj=1&apiMin=0&oneboxName=directory_example&
ipAddr=10.72.1.3&authType=none&lang=en&query=smith
GET https://directory.corp.acme.com/cgi-bin/phonebook?
apiMaj=1&apiMin=0&oneboxName=secure_example&
ipAddr=10.72.1.3&lang=en&authType=basic&userName=jdoe&password=Co0lOneBoX&query=
smith
```

The OneBox trigger term is not sent in the GET command. In the examples, the user enters `directory smith`, but the GET command includes only `smith` as the value for the `query` parameter. The complete set of input parameters from a search appliance are defined in "Call Parameters" on page 25.

Receiving a Provider's Response

Upon receiving the a request from a search appliance, a provider processes the input parameters and compiles a result set to return to the search appliance. The provider response must conform to the OneBox Results Schema (see "OneBox Results Schema Reference" on page 26).

The following example shows the results from a `directory smith` query and the directory items for William Smith and Bill Smith. The `<MODULE_RESULT>` tag contains the information sent from the provider for the search results:

```
<?xml version="1.0" encoding="UTF-8"?>
<OneBoxResults>
  <provider>ACME Employee Directory</provider>
  <title>
    <urlText>13 results in the ACME directory</urlText>
    <urlLink>http://directory.corp.acme.com/cgi-bin/search?smith</urlLink>
  </title>
  <IMAGE_SOURCE>http://directory.corp.acme.com/images/directory.jpg
    </IMAGE_SOURCE>
  <MODULE_RESULT>
    <U>http://directory.corp.acme.com/cqi-bin/lookup?empid=448473</U>
    <Field name="display">Smith, William</Field>
    <Field name="firstname">William</Field>
    <Field name="lastname">Smith</Field>
    <Field name="phone">617-555-1234</Field>
    <Field name="email">wsmith@acme.com</Field>
    <Field name="picture">http://directory.corp.acme.com/cqi-bin/lookup?
      photo=448473</Field>
  </MODULE_RESULT>
  <MODULE_RESULT>
    <U>http://directory.corp.acme.com/cqi-bin/lookup?empid=22638</U>
    <Field name="display">Smith, Bill R.</Field>
    <Field name="firstname">Bill</Field>
    <Field name="lastname">Smith</Field>
    <Field name="phone">617-555-9345</Field>
    <Field name="email">bsmith@acme.com</Field>
    <Field name="picture">http://directory.corp.acme.com/cqi-bin/lookup?
      photo=22638</Field>
  </MODULE_RESULT>
</OneBoxResults>
```

Transforming XML to HTML

You can transform the XML results into HTML by means of an XSLT stylesheet template.

The `<Field>` elements (see the element “MODULE_RESULT” on page 28) are available for both external and internal OneBox provider.

The `<Field>` elements provide data about the results from a provider, and you can specify parameters to improve the display of the results data. In the directory example, the provider sends the search appliance two results and title display information:

- The title is a clickable URL with the text “13 results in the ACME directory.”
- An icon image makes the results stand out as directory entries.
- The two results are directory listing results, including a display name, first name, last name, phone number, email address, and link to an image of the employee.

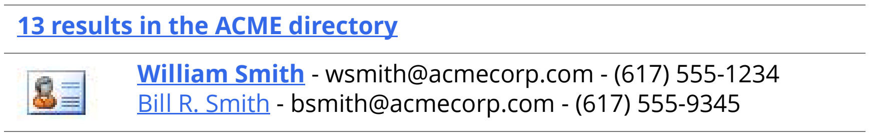
The `<Field>` element’s `name` attribute provides the information for the OneBox.

Formatting the Results

Both internal and external providers respond to a OneBox call with XML results.

Google enforces a vertical limit of 150 pixels on the HTML for the output.

To display the results as HTML, the OneBox module definition can contain an XSLT template that a search appliance applies in a front end. You specify the XSLT template inside the `<resultsTemplate>` element. For example, you can specify the directory example as follows:



13 results in the ACME directory

William Smith - wsmith@acmecorp.com - (617) 555-1234
Bill R. Smith - bsmith@acmecorp.com - (617) 555-9345

The XSLT template transforms the associated results into HTML suitable for display. The following XSLT example creates an HTML table and transforms the `<Field>` name attributes:

```
<xsl:template name="directory">
  <table border="0" cellpadding="1" cellspacing="0">
    <tbody>

      <tr>
        <td colspan="2">
          <nobr>
            <a>
              <xsl:attribute name="href">
                <xsl:value-of select="title/urlLink"/>
              </xsl:attribute>
              <b>
                <xsl:value-of select="title/urlText"/>
              </b>
            </a>
          </nobr>
        </td>
      </tr>

      <tr>
        <td valign="top" width="40">
          <img>
            <xsl:attribute name="src">
     <xsl:value-of select="IMAGE_SOURCE"/>
            </xsl:attribute>
          </img>
        </td>
        <td valign="top" width="99%">
          <xsl:for-each select="MODULE_RESULT">
            <font size="-1">
              <b>
                <a>
                  <xsl:attribute name="href">
     <xsl:value-of select="U"/>
                  </xsl:attribute>
                  <xsl:value-of select="Field[@name='firstname']"/>
                  <xsl:value-of select="Field[@name='lastname']"/>
                </a>
              </b> -
     <xsl:value-of select="Field[@name='email']"/> -
              <xsl:value-of select="Field[@name='phone']"/>
            </font>
            <br/>
          </xsl:for-each>
        </td>
      </tr>

    </tbody>
  </table>
</xsl:template>
```

The stylesheet template must begin with the `<xsl:template>` element, which is matched to generate the HTML results. The `<xsl:template>` element must include a `name` attribute, and must not include the `match` attribute, which interferes with other stylesheet operations in the search appliance's front end. You can include other `<xsl:template>` elements within the stylesheet.

If you do not specify an XSLT template, the search appliance uses the default template. The default XSLT template only returns a maximum of three results.

Checking the Visual Layout

You can verify an XSLT template by:

- Using the OneBox Simulator (see “Testing with the OneBox Simulator” on page 19).
- Inserting a temporary wrapper element in the XSLT file. The wrapper enables you to see how the OneBox results appears, but doesn’t provide an interface for testing different parameters as does the OneBox Simulator.

Note: If you use the wrapper code to verify your layout, you must remove the wrapper code before deploying your template on a search appliance. The `match` statement in the wrapper is not permitted in a search appliance template.

Creating a Wrapper

To create a wrapper:

1. Install an XSLT processing application such as Saxon and Java as described in “Planning” on page 7. The compilation step in this procedure uses Saxon.

2. Add the following statements at the start of your XSLT template after the `<?xml` version, `<xsl:stylesheet`, and `<xsl:output` statements. This example calls the XSLT template module shown in “Formatting the Results” on page 14:

   ```
   <xsl:output method="xml" />
     <xsl:template match="OneBoxResults">
     <html>
     <body>
       <xsl:call-template name="directory"/>
     </body>
     </html>
   </xsl:template>
   ```

   The code is as follows:

   - The first three statements identify the XSLT code and the output method.
   - The `xsl:template match` statement picks up the `<OneBoxResults>` tag in the example XML file shown in “Receiving a Provider’s Response” on page 13.
   - The `<html>` and `<body>` statements provide starting and closing and tags to wrap the code that the XSLT template generates in the `call-template` statement.
   - The `xsl:call-template` statement calls the XSLT template module.
   - The `</xsl:template>` statement closes the wrapper code block.

3. Use the lines similar to the following to compile and view the OneBox module (these are from a Windows command prompt):

   ```
   java -jar c:\saxon\saxon8.jar -t directory.xml dirtest.xsl > test.html
   call start firefox test.html
   ```

Adding a Secure Connection to the Provider

After defining a OneBox module, you can add advanced features to ensure security of data and provide additional user functionality.

When sensitive information passes between a search appliance and an external provider, it's best to use an SSL connection for secure data transfer. You do this by specifying the external provider URL as `https` in the OneBox module definition. The secure URL causes the search appliance to establish a protected session for transferring data, and request a valid certificate from the provider. The certificate is validated using the Certificate Authority and Certificate Revocation List information that is configured on the search appliance. If the provider requests a mutually authenticated certificate, the search appliance transmits its certificate as configured in the Admin Console.

Another form of authentication between a search appliance and the provider is HTTP Basic authentication. With this method, the search appliance sends a username and password in the HTTP header to the provider. To enable HTTP Basic authentication, set the `<GSA_username>` and `<GSA_password>` elements with a username and password that represent a provider "account" that is associated with the search appliance. The search appliance makes HTTP Basic authentication requests with each request to the provider. When using HTTP Basic authentication in production, always use SSL to avoid passing credentials over the network in clear text.

It's a good idea to disable security before testing a provider so that debugging is easier. After the provider is functioning properly, enable the secure connection.

Defining User-Specific Results and Access Control

The search appliance provides document-level security, so that users can view search results only for content to which they have access. Google Search Appliance supports HTTP Basic authentication, NTLM HTTP authentication, and LDAP authentication plus forms-based single sign-on (SSO) systems. OneBox also supports user-based information retrieval, and can interoperate with these access control schemes.

If you use user level access control, you must specify the `userAuth` attribute in the `<security>` element (see the element "security" on page 23) of the module definition. When a secure search is executed against a search appliance (`access=a`), OneBox modules with user access level control configured are called.

The `userAuth` attribute can have one of the following values:

- `none`—No user authentication performed.
- `basic`—HTTP Basic authentication. The search appliance passes a username and password to the provider.
- `ldap`—LDAP authentication. Authenticates a user against the configured LDAP directory server, and the user's distinguished name (DN) is passed to the provider.
- `sso`—Forms-based single sign-on authentication. The user's SSO cookie is passed to the provider. Used by the Google Search Appliance only. Forms-based authentication is limited to Google Search Appliance.

You can use a mixture of these access control mechanisms on OneBox modules within the same user query, but the search appliance may need to prompt the user for credentials or forward their session to a single sign on login page. The Google Search Appliance supports prompting the user for only one set of credentials (username and password for HTTP Basic authentication, NTLM HTTP, and LDAP) and one forms-based login per query.

For information on each authentication method and how to configure authentication on the Admin Console, see *Managing Search for Controlled-Access Content*.

Passing Optional Contextual Data

The OneBox system can pass a set of contextual data that is useful in developing an external provider and determining more personalized, relevant results. The following additional contextual information is passed to the provider:

- Date and time of the query
- IP address of the querying user's machine
- Language of the query as defined by a user's browser setting

You can use or suppress these options. You can use the options if the provider can use them to enhance search results. You might prefer to suppress these options to make the call to the provider smaller and decrease network overhead.

To suppress the optional data, you specify attributes on the top level `<onebox>` element. The following example suppresses the optional data in our directory example:

```
<?xml version="1.0" encoding="UTF-8"?>
  <GoogleEnterpriseSources xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <onebox type="external" suppressDateTime="true" suppressIPAddr="true">
  ...
```

Handling Errors and Lookup Failures

Sometimes a provider cannot return a result set. For example, a provider could fail to return results because the provider's request to its data source times out, because authentication fails, or because the provider's data lookup completes, but returns zero results. In such a case, a provider should send a results message with the following characteristics:

- The value of the `<resultCode>` element is `lookupFailure`, `securityFailure`, or `timeout`.
- There are no instances of the `<MODULE_RESULT>` element.

Optionally, a provider can use the `<Diagnostics>` element to return more detailed information.

When a search appliance receives a response whose `<resultCode>` element is set to an explicit value other than `success`, the search appliance logs the response. The user's search results do not include OneBox results from a provider or any explicit indication of a failure.

An error condition appears as XML code in the `<resultCode>` element. In the following example, the ACME employee directory requires a username and password, and in this example, the password passed is incorrect:

```
<?xml version="1.0" encoding="UTF-8"?>
<OneBoxResults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <resultCode>"securityFailure"</resultCode>
  <Diagnostics>invalid password</Diagnostics>
  <provider>ACME Employee Directory</provider>
</OneBoxResults>
```

If a OneBox provider becomes unreachable and a search appliance receives an HTTP error code, the search appliance logs the error. No error is shown to the user.

Tips

This section provides information about how to use OneBox more effectively.

Checking XML Results

When you are troubleshooting a OneBox module that has a running provider, you can issue a query, change the URL, and then view the XML results.

To view XML results:

1. Display the search page and issue a query that invokes the OneBox provider.

2. In the address bar of the browser window, remove the `&proxystylesheet=`*`front_end`* parameter from the URL. For example, the following URL searches for `query` (line breaks added for readability):

   ```
   http://gsa42/search?q=query&site=default_collection&btnG=Google+Search
   &access=p&entqr=0&sort=date%3AD%3AL%3Ad1&output=xml_no_dtd
   &client=default_frontend&ud=1&oe=UTF-8&ie=UTF-8
   &proxystylesheet=default_frontend
   ```

   After you remove `&proxystylesheet=default_frontend`, the URL becomes:

   ```
   http://gsa42/search?q=query&site=default_collection&btnG=Google+Search
   &access=p&entqr=0&sort=date%3AD%3AL%3Ad1&output=xml_no_dtd
   &client=default_frontend&ud=1&oe=UTF-8&ie=UTF-8
   ```

3. Click Enter in the address bar.

The browser now displays the results XML used to display the results. You can use this technique to compare XML results with the results page formatted by your XSLT template.

Testing with the OneBox Simulator

Google provides an open source simulator that you can use to test a OneBox module and XSLT stylesheet.

Downloading the OneBox Simulator

You can download the OneBox simulator from http://google-developers.appspot.com/search-appliance/download/downloadsdk.

Creating an XSLT File

Create an XSLT style sheet and rename the file as `projects.xsl` for use with the simulator:

```
<xsl:template name="projects">
  (your XSL code)
</xsl:template>
```

Your XSLT code can contain other `xsl:template` elements. Ensure that you do not include the `match` attribute in the XSLT code for a OneBox module.

Editing customer-onebox.xsl to Call Your XSL File

Edit the `customer-onebox.xsl` file to point to the `projects.xsl` file as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl=" http://www.w3.org/1999/XSL/Transform" version="1.0">

  <!-- edit the xsl:include to point to your stylesheet file -->

  <xsl:include href="projects.xsl"/>

  <!-- change the xsl:call-template to call your template’s name -->

  <xsl:template name="holder" match="OBRES">
    <div class="oneboxResults">
      <table>
        <xsl:call-template name="projects"/>
      </table>
    </div>
  </xsl:template>
</xsl:stylesheet>
```

This file calls your stylesheet for all OneBox results, whether or not the results come from the `projects` OneBox. In a search appliance, the `customer-onebox.xsl` file only passes results to your style sheet. The `customer-onebox.xsl` cannot be used as a OneBox module because it contains a `match` statement.

Merging a OneBox Module Output with the Search Results

To merge a OneBox module output with the search results:

1. Append the argument `--dumpOutput=1` to the command line when you invoke the simulator. Invoke the simulator from the directory in which you unzipped the simulator zip file.

2. Each time your provider returns a valid result, the simulator creates a new XML file in the current directory, named *`name`*`-Results-`*`number`*`.xml`, where *`name`* is the name of your OneBox module, and *`number`* is a unique integer. The simulator merges the XML for you.

3. Insert your OneBox module’s results into the search output XML. Copy the contents of your `<OneBoxResults>` element into the `<OBRES>` element, replacing the `module_name` attribute with your module name. Omit the `<OneBoxResults>` element itself.

   For example, the `projects` output appears as follows for a query on `project 1`:

   ```
   <OneBoxResults>
     <provider>Corporate Project Database</provider>
     <title>
       <urlText>Lookup results for project 1</urlText>
       <urlLink>http://www.mycompany.com/cgi-bin/projects?projectId=1</urlLink>
     </title>
     <IMAGE_SOURCE>http://www.mycompany.com/icons/favicon.ico</IMAGE_SOURCE>
     <MODULE_RESULT>
       <U>http://www.mycompany.com/cgi-bin/projects?projectId=1</U>
   <Field name="ProjectName">NextGen</Field>
       <Field name="ProjectStatus">Green</Field>
     </MODULE_RESULT>
   </OneBoxResults>
   ```

4. Edit `search.xml` to appear as follows:

```
<ENTOBRESULTS>
  <OBRES module_name="projects">
    <provider>Corporate Project Database</provider>
    <title>
      <urlText>Lookup results for project 1</urlText>
      <urlLink>
        http://www.mycompany.com/cgi-bin/projects?projectId=1
      </urlLink>
    </title>
    <IMAGE_SOURCE>http://www.mycompany.com/icons/favicon.ico</IMAGE_SOURCE>
    <MODULE_RESULT>
      <u> http://www.mycompany.com/cgi-bin/projects?projectId=1</u>
      <Field name="ProjectName">NextGen</Field>
      <Field name="ProjectStatus">Green</Field>
    </MODULE_RESULT>
  </OBRES>
</ENTOBRESULTS>
```

Applying the Overall Stylesheet to the Merged XML Output

The `gsa_default_stylesheet.en.xsl`, the overall stylesheet, includes `customer- onebox.xsl` that includes `projects.xsl`. After you apply `gsa_default_stylesheet.en.xsl`, the simulator transforms the merged results page into HTML for you to view. You can use any available XSLT tool such as XML Spy, OxygenXML, tools from the major software vendors, or open source tools.

The simulator requires that you use the `projects.xsl` name. The stylesheet name that you specify in the Admin Console must appear in the following places:

1. The stylesheet file name and `xsl:template` *name* attribute.
2. The `customer-onebox.xsl` file.
3. The *`module_name`* attribute of the `<OBRES>` element in the search XML.

Notes:

- The XSD files are not applied by the simulator, on input or on output. Ensure that your XML conforms to the relevant schema.
- Although a URL beginning with HTTPS is accepted in the `<providerURL>`, all simulator calls use HTTP, not HTTPS.

OneBox Module Definition XML Reference

You can define a OneBox module by editing an XML file and importing the file into the Admin Console from **Content Sources > OneBox Modules**. Alternatively, you can enter these settings from the same Admin Console page by editing an existing OneBox module or by defining a new OneBox module.

Module Definition Schema

The OneBox module definition schema file is available in the `onebox.xsd` file, which is available in the OneBox simulator download (http://google-developers.appspot.com/search-appliance/download/downloadsdk). Your OneBox module definition XML file must conform to this schema.

For information about GET call parameters, see "Call Parameters" on page 25.

The `GoogleEnterpriseSources` element contains the following elements:

- `<globals>`*`child-elements`*`</globals>`
- `<onebox>`*`child-elements`*`</onebox>`
- `<ModulesPerFrontend>`*`child-elements`*`</ModulesPerFrontend>`

Note: The `globals` element provides the Admin Console with the maximum number of OneBox results per search and the OneBox response timeout in milliseconds. Do not put the `globals` element in your XML file. Similarly, the Admin Console uses the `ModulesPerFrontend` element internally. Do not put the `ModulesPerFrontend` element in your XML file.

onebox Element

The `onebox` element provides the root element of a `onebox` module definition. This element contains all of the settings for a module.

Note: Do not set the `id=` attributes that appear in the `onebox.xsd` file. The search appliance does not use any of the `id` attributes.

Syntax

```
<onebox type="type" suppressDateTime="true-false" suppressIPAddr="true-false">
  <name>string</name>
  <security userAuth="authType" id="security">
    <GSA_username>username</GSA_username>
    <GSA_password>password</GSA_password>
  </security>
  <description>string</description>
  <trigger triggerType="null_keyword_regex">keyword1|keyword2|keywordn</trigger>
  <providerURL>url</providerURL>
  <collection>name</collection>
  <resultsTemplate>
    ...
  </resultsTemplate>
</onebox>
```

The `onebox.xsd` elements are as follows:

`<onebox>` *`attributes_and_child_elements`* `</onebox>`	Required. Indicates OneBox configuration parameters.

Element Attributes

Attribute	Description
`type="internal"|"external"`	Required. The type of onebox module, `internal` (calling a collection) or `external` (calling an external provider).
`suppressDateTime="true"|"false"`	Optional. Determines whether the search appliance passes date and time information to the external provider.
`suppressIPAddr="true"|"false"`	Optional. Determines whether the search appliance passes the IP address of the querying user's machine.

Child Elements

Element	Description
name	`<name>`*`string`*`</name>` Required. The name of the OneBox module. The name can be up to 32 characters in length and must contain at least one character. The name can consist only of the characters A–Z, a–z, 0–9, dot, and dash.
security	`<security>`*`child-elements`*`</security>` Optional. Defines user-level access control and security between the search appliance and the provider. User-level access is defined by the `userAuth` attribute and security between the search appliance and a provider is defined by means of the `GSA_username` and `GSA_password` attributes. The `<security>` element has the following attribute: • `userAuth="none"|"basic"|"LDAP"|"SSO"` Required if the `<security>` element is present. The type of user-level access control to apply. If you specify a value other than `none`, you must configure the search appliance with the appropriate settings. The `<security>` element contains the following child elements: • `<GSA_username>`*`username`*`</GSA_username>` Optional. The user name for the search appliance, in an HTTP Basic authentication request to the provider. The maximum user name length is 32 characters. If you define this element, the appliance makes an HTTP Basic authentication request to the provider passing the credentials defined in the HTTP header. If authentication is not required, specify without a value as `<GSA_username></GSA_username>`. • `<GSA_password>`*`password`*`</GSA_password>` Optional. The password for the search appliance, in an HTTP Basic authentication request to the provider. The maximum password length is 32 characters. If you define this element, the appliance makes an HTTP Basic authentication request to the provider passing the credentials defined in the HTTP header. If authentication is not required, specify without a value as `<GSA_password></GSA_password>`.

Element	Description
description	`<description>`*`text`*`</description>` Optional. A textual description of *`what`* the OneBox module does. This value is important if you plan to make the modules that you develop available for others to use. The description can contain up to 512 characters.
trigger	`<trigger>`*`keyword_or_regex`*`</trigger>` Optional. The trigger expression determines when the search appliance invokes the module. For information on regular expression syntax, see “Triggering When the Query Matches a Regular Expression” on page 11. You can specify multiple trigger values by separating each value with an OR symbol or a pipe (|). For example, you can specify `<trigger> bug|CR </trigger>`. The `trigger` element has the following required attribute: • `triggerType="null"|"keyword"|"regex"` Required if the `<trigger>` element is present. The `triggerType` can have one of the following values: • `null`—Trigger the keyword on every search. The OneBox module then appears in all search results. • `keyword`—Trigger only when a search string starts with the indicated keyword or keywords. • `regex`—Trigger when a search string matches the regular expression statement. For information on the regular expression syntax, see “Triggering When the Query Matches a Regular Expression” on page 11.
collection	`<collection>`*`collection_name`*`</collection>` Optional. Specifies the collection that is the provider for an internal OneBox module. If the `type` attribute of the onebox element is set to `internal`, this element must contain a collection name. The collection name can be up to 32 characters in length.
providerURL	`<providerURL>`*`url`*`</providerURL>` Optional. A fully qualified URL to which the GET request is passed for an external OneBox module. If the `type` attribute of the onebox element is set to `external`, this element must contain a URL. If the URL uses the HTTPS protocol, the search appliance creates a secure (SSL) session with the provider.
resultsTemplate	`<resultsTemplate>`*`child_elements`*`</resultsTemplate>` Optional. The search appliance uses the XSLT template to convert the XML response from the provider into HTML for the end user. The top level element must be the `<xsl:template>` element which will be matched to generate the HTML results. The `<xsl:template>` element must include a `name` attribute, and must not include have the `match` attribute, which interferes with other stylesheet operations in the search appliance front end. You can include other `<xsl:template>` elements within the stylesheet.

Call Parameters

A GET request from a search appliance is sent to the external provider as specified in the OneBox module definition. The table below lists the parameters in a GET call.

Parameter	Required	Description	Type of Value
`apiMaj`	required	API major version number. Changes in this value may break compatibility.	Integer (1, 2, 3...)
`apiMin`	required	API minor version number. Changes in this value do not break compatibility.	Integer (0, 1, 2, 3...)
`authType`	required	The authentication mechanism used to provide user-specific information.	One of the following values: `none`, `basic`, `ldap`, or `sso`
`dateTime`	optional	Date and time of the query on the calling search appliance.	Text, UTC format
`ipAddr`	optional	The IP address of the originating user.	IP address
`lang`	required	The language of the user's browser where the query originated.	Text, two-character language code, such as EN, JP, or DE
`oneboxName`	required	Name of the OneBox module. Must match the OneBox Module definition specified in the Admin Console.	Text, no spaces
`password`	optional	The password for the user if HTTP Basic authentication is being used.	Text
`P0, P1, P2, ..., Pn`	required	Match groups from the regular expression evaluation (if applicable)	Text
`query`	required	The query string from the user's query to a search appliance. The query string does not include the trigger term.	UTF-8 encoded and URL-escaped text
`userName`	optional	The user identity for secure or personalized results from a provider: • If `authType=basic`, *`userName`* is the username of the search user. • If `authType=ldap`, *`userName`* is the distinguished name (DN) from the LDAP server. • If `authType=sso`, *`userName`* is the user of the cookie used by a provider.	Text

Certain parameters can be suppressed if the appropriate attributes are set on the `<onebox>` element. Access control settings are only passed if the OneBox module is configured to include user-level access control. If the provider is configured to use basic authentication between the search appliance and the provider (`<GSA_username>` and `<GSA_password>` elements are defined, see the element "security" on page 23) then the GET request will include these parameters in the HTTP header.

OneBox Results Schema Reference

Results are returned to a search appliance in the form of XML that conforms to the OneBox Results schema.

Results Definition Schema

The results definition schema file is available in the `oneboxresults.xsd` file, which is included in the OneBox simulator download (http://google-developers.appspot.com/search-appliance/download/downloadsdk).

Note: Do not set the `id=` attributes that appear in the `oneboxresults.xsd` file. The search appliance does not use any of the `id` attributes.

Your OneBox module definition XML file must conform to this schema.

Syntax

```
<OneBoxResults>
  <resultCode>result_code</resultCode>
  <Diagnostics>failure_reason</Diagnostics>
  <provider>provider_name</provider>
  <searchTerm>query_escape</searchTerm>
  <totalResults>total_results_escape</totalResults>
  <title>
    <urlText>results_title</urlText>
    <urlLink>results_url</urlLink>
  </title>
  <IMAGE_SOURCE>image_url</IMAGE_SOURCE>
  <MODULE_RESULT>
    <U>url</U>
    <Title>title</Title>
    <Field name="name1">value1</Field>
    <Field name="name2">value2</Field>
    <Field name="nameN">valueN</Field>
  </MODULE_RESULT>
</OneBoxResults>
```

OneBoxResults Elements

The `OneBoxResults` element supports the following child elements:

Element	Description
resultCode	`<resultCode>`*`result_code`*`</resultCode>` Optional, a single use of this tag is permitted. The return code is from the OneBox provider. The value `success` is assumed if no value is returned, and results are processed only on success. Although this element is optional, it is good practice to always return a result code. The value can be one of the following: • `success` • `lookupFailure` • `securityFailure` • `timeout`
Diagnostics	`<Diagnostics>`*`failure_reason`*`</Diagnostics>` Optional, a single use of this tag is permitted. If a lookup fails, the provider uses this element to send diagnostic information and sets the `resultCode` attribute to a value other than `success`. It is not illegal to send a diagnostic alone, or with a `resultCode` of `success`, but the diagnostic message may be logged differently depending on the implementation. The failure reason string can be up to 256 characters in length.
provider	`<provider>`*`provider_name`*`</provider>` Optional, a single use of this tag is permitted. The name of the provider. If this is an internal provider, specify the collection name. The name need not match the name provided in the OneBox module definition, and can be more descriptive than that name. This string can be a brand, URL, or other identifying information. The provider name can be up to 128 characters in length.
title	`<title>`*`child-elements`*`</title>` Optional, a single use of this tag is permitted. The title of the result, consisting of a line of text and a link to the full result set from the provider. If one of the child elements is present, both must be present. The `title` element contains the following child elements: • `<urlText>`*`results_title`*`</urlText>` Required if you specify the `<title>` element. The title text for the results, for example, `Search Results`. The title can be up to 40 characters in length. • `<urlLink>`*`results_url`*`</urlLink>` Required if you specify the `<title>` element. The URL to the results.

Element	Description
IMAGE_SOURCE	`<IMAGE_SOURCE>`*`image_url`*`</IMAGE_SOURCE>` Optional. A link to the image that displays in the OneBox results.
MODULE_RESULT	`<MODULE_RESULT>`*`child_elements`*`</MODULE_RESULT>` Optional. Describes the results that the provider sends to the search appliance. The search appliance supports up to eight `MODULE_RESULT` blocks from external providers, or up to three from an internal provider. The absence of this element means that there were no search results found for the query. The `MODULE_RESULT` element has the following child elements: • `<U>`*`url`*`</U>` Optional. The URL for the results. • `<Title>`*`module_title`*`</Title>` Optional, only one title is permitted in a `MODULE_RESULT` block. The title of the results. • `<Field name="`*`field_name`*`" [`*`any`*`] >`*`field_value`*`</Field>` Name-value pairs for each returned item. For internal OneBox providers, you can have any number of `Field` elements. For an external Onebox provider only, you can add up to 8 `<Field>` elements. This element has the following attributes: • `name="`*`text`*`"`—The name attribute allows you to give the field a name for results formatting and processing. The default XSLT template uses this attribute to format results. • `[`*`any`*`]`—You can add additional to the `Field` element for formatting results.

Index

Symbols
&proxystylesheet 19

A
access control 7, 8, 17
Admin Console 8
attributes
  suppressDateTime 23
  suppressIPAddr 23
  trigger type 10
  type 23
  userAuth 17
authentication 8, 17

B
Basic HTTP authentication 8

C
certificate validation 17
collection element 12, 24
Content Sources > OneBox Modules page 6, 8
contextual data 18
Custom KeyMatch OneBox 4
customer-onebox.xsl 20

D
date of query 18
description element 9, 24
Diagnostics element 18, 27

E
elements
  collection 12, 24
  description 9, 24
  Diagnostics 18, 27
  field 14
  globals 22
  GoogleEnterpriseSources 22
  GSA_password 17
  GSA_username 17
  IMAGE_SOURCE 28
  MODULE_RESULT 18, 28
  ModulesPerFrontEnd 22
  name 9, 23
  onebox 9, 18, 22
  OneBoxResults 27
  provider 27
  providerURL 12, 24
  resultCode 18, 27
  resultsTemplate 14, 24
  security 17, 23
  title 27
  trigger 9, 24
errors 18
example OneBox module 5
external provider
  adding a secure connection 17
  description 6
  methods for defining 7
  response 13
  search process 12
  URL as https 17
  using 12

F
Field elements 14

G
general information component 4
globals element 22
Google OneBox Simulator 7
GoogleEnterpriseSources element 22

gsa_default_stylesheet.en.xsl 21
GSA_password element 17
GSA_username element 17

H
HTML
  limit on output 14
  transforming results from XML 14
HTTP Basic authentication 17
HTTP GET command 13, 25

I
IMAGE_SOURCE element 28
internal provider
  adding a secure connection 17
  calling 12
  description 6
  method for defining 7
  response 13
IP address of user's machine 18

K
keyword trigger 10

L
language of the query 18
LDAP authentication 8, 17
lookup failures 18

M
MODULE_RESULT element 18, 28
ModulesPerFrontend element 22

N
name element 9, 23
NTLM HTTP authentication 17

O
onebox element 9, 18, 22
OneBox module
  description 4–6
  example 5
  maximum results 22
  merge output with results 20
  planning 7
  results schema 26
  returning results 10
  schema 22
  steps to deploy 6
OneBox Simulator 4, 6, 16, 19, 22
OneBox Style Template 8
onebox.xsd file 22
OneBoxResults element 27
oneboxresults.xsd 26

P
Perl Compatible Regular Expressions library 11
provider
  component 4
  element 27
providerURL element 24

R
regular expression trigger 11
resultCode element 18, 27
Results Template component 4
results, formatting 14
resultsTemplate element 14, 24

S
Search > Search Features > Front Ends page 6
secure connection 17
security component 4
security element 17, 23
single sign-on systems 8
SSL connection 17
suppressDateTime attribute 23
suppressIPAddr attribute 23

T
time of query 18
title element 27
trigger
  component 4
  description 9
  element 9, 24
  keywords 10
  regular expressions 11
triggerType attribute 10
type attribute 23

U
userAuth attribute 17

W
wrapper in XSLT 16

X
XML
  configuration file 9–28
  reference 21–28
  results 19
  transform results to HTML 14
XSLT
  tools 7
  wrapper code 16
XSLT template 14
  default 15
  example 15
  verifying 16