RightNow® February '08

Integration Manual

February 15, 2008

RIGHT NOW TECHNOLOGIES

Web address: http://rightnow.com
Email address: info@rightnow.com

Contents

1

Introduction

RightNow helps businesses deliver exceptional customer experiences that drive competitive advantage and business growth, while reducing operation costs. With our newest offering, RightNow February '08, you can deliver great experiences to every customer all the time.

Using RightNow's customer service, sales, marketing, and feedback solutions, you can guarantee that your customers and front-line employees—customer service agents, marketers, and salespeople—have the information they need when they need it. We call it "knowledge at the point of action," and we deliver it through a combination of our intuitive knowledge foundation (iKnow) that dynamically learns from every customer interaction, our suite of front-line action applications that facilitate knowledge delivery across all your organization's channels and touch points, and Day1 Advantage, our results-based engagement model, which ensures that you start with results and build on success.

RightNow February '08 delivers real-time actionable knowledge, guiding customers and employees to make the right decisions and take the right actions for their buying, selling, and servicing needs—right now.

RightNow Service

RightNow's industry-leading customer service and support solution delivers high-value, consistent customer experiences across multiple customer service channels. Using RightNow Service, you can provide your customers with fast and accurate answers from phone, email, web, and chat requests. RightNow Service puts knowledge at the fingertips of your customer service agents to quickly and consistently help customers and enables your customers to help themselves with powerful and intuitive web and voice self-service capabilities.

RightNow Service is seamlessly integrated with RightNow Marketing, RightNow Sales, RightNow Analytics, and RightNow Feedback, enabling your organization to capture high-value insights from customer service interactions to drive better marketing experiences and product development decisions.

RightNow Marketing

Marketing is often the first point of contact with a customer; as a result, that first experience is crucial to how the customer views your organization. RightNow Marketing is an email and campaign marketing solution that ensures high-value customer experiences across your marketing touch points.

Using multi-stage marketing campaigns, you can quickly target and deliver the right information and product offers to the right recipients at the right moment. By providing your customers and prospects with what they need when they need it, you not only create a great experience, you also optimize the effectiveness and cost-efficiency of your marketing programs.

RightNow Marketing is seamlessly integrated with RightNow Service, RightNow Sales, RightNow Analytics, and RightNow Feedback, so your organization can act on every new lead in a timely, appropriate manner and provide your marketing team with more accurate, complete, and up-to-date customer data.

RightNow Feedback

RightNow makes it easy for you to find out what your customers really think—by asking them at the right time and in an appropriate manner. RightNow Feedback is a customer survey tool for gathering information about your customers' experiences. The resulting information will help your organization improve customer experiences and increase customer loyalty.

RightNow Feedback also unifies all of your organization's feedback programs into a single enterprise feedback management solution that supports your organization's complete feedback strategy. Surveys can target diverse internal and external audiences for broad purposes such as improving business processes, ensuring quality compliance, improving customer and employee satisfaction, and more.

The key to any organization's long-term success is knowing what customers expect today and will demand tomorrow. Using our flexible and robust survey functionality, you can capture and measure feedback across all touch points in real time and take immediate action on that feedback.

RightNow Sales

RightNow's sales automation solution enables sales teams to capitalize on every opportunity to maximize sales performance and provide a superior customer or prospect experience. RightNow Sales provides comprehensive tools to quickly and effectively manage opportunities, contacts, leads, and tasks; optimize analysis of opportunities and deal pipelines; analyze and track performance of individuals and teams; and automate quote generation.

Seamlessly integrated with RightNow Service, RightNow Marketing, RightNow Analytics, and RightNow Feedback, RightNow Sales can assist your organization in building sustainable, long-term relationships with customers by understanding their needs and ensuring that those needs are met—before, during, and after the sale.

RightNow Analytics

To deliver a great customer experience, you need to know what you are doing right and what you need to do better. That requires full visibility into all of your customer touch points across customer service, sales, marketing, and feedback activities. You also have to be able to deliver timely, actionable analytics information to managers and decision-makers across your entire organization. With RightNow Analytics—our business analytics software—you can capture, analyze, and distribute information about customer interactions with ease and flexibility.

RightNow Analytics is embedded throughout RightNow, providing your organization with a unified view of all analytics across all channels. With over 300 standard reports and the ability to create custom reports and dashboards, you can easily measure your most critical performance metrics and quickly respond to changing conditions and customer needs.

About this manual

This manual is intended for administrators responsible for carrying out integrations in RightNow February '08. It contains information and procedures for implementing pass-through authentications, external event handlers, or XML API integrations.

Refer to the *RightNow Administrator Manual* for an overview of the RightNow Console and configuration procedures for those areas common to all RightNow products, including RightNow Service, RightNow Marketing, RightNow Feedback, and RightNow Sales. Refer to the *RightNow User Manual* for an overview of the RightNow Console and information and procedures for performing tasks associated with areas that are common to all RightNow products.

Chapter 2, Integration Overview—Contains a description of each type of integration.

Chapter 3, XML Integration—Contains information for implementing an integration using XML to access RightNow's API and update the database.

Chapter 4, Event Handlers—Contains information for implementing event handlers to define custom processes for managing your incidents, contacts, organizations, answers, and opportunities.

Chapter 5, Pass-Through Authentication—Contains information for integrating RightNow Service with an external customer validation source to allow your customers to automatically log in to RightNow Service from an external web page.

Appendix A, Pair Names—Describes the pairs available to be used in each XML API function.

Appendix B, Source Codes—Describes the source codes to be used in the source_lvl1, source_lvl2, and tbl pairs in each XML API function.

Documentation conventions

As you work with RightNow documentation, you will notice certain conventions used to convey information. To help you become familiar with these conventions, the following table contains examples and descriptions of the conventions used.

Convention	Description
Path: Common Configuration>Double-Click Staff Accounts	Identifies the path to open an administration item. The administration option appears first, followed by the mouse action. **Note:** All paths appear immediately before figures in the documentation.
Path: Answers>Double-Click Report>Right-Click Answer>Open>Answer	Identifies the path to open a record from a report. The navigation list appears first, followed by the mouse actions and the menu selection.
<angle brackets> as in: http://<your_interface>.custhelp.com/	Indicates variable information specific to your RightNow application.
Asterisk (*) preceding field names in tables	Indicates that the field is required. You cannot save a record, report, or file until you fill in all required fields. **Note:** In RightNow, required fields are flagged with an asterisk, or the field name appears in red text, or both.

RightNow February '08 documentation

RightNow Technologies offers manuals, guides, and documents to help you install, administer, and use RightNow products, including RightNow Service™, RightNow Marketing™, RightNow Sales™, and RightNow Feedback™. Our documentation is written for users who have a working knowledge of their operating system and web browsers and are familiar with standard conventions such as using menus and commands to open, save, and close files.

RightNow Administrator Manual—Contains procedures for configuring options common to RightNow Service, RightNow Marketing, RightNow Feedback, and RightNow Sales. This manual addresses staff management, common communications, custom fields, customizable menus, workspaces, navigation sets, monetary configuration, business rules, system configuration, database administration, contact upload, multiple interfaces, screen pops, computer telephony integration (CTI), and the external suppression list.

RightNow User Manual—Contains procedures common to all staff members, including customer service agents, marketing personnel, and sales representatives. This manual addresses organization and contact records, tasks, notifications, and computer telephony integration (CTI).

RightNow Analytics Manual—Contains procedures for working with RightNow Analytics, including generating standard reports and creating custom reports and dashboards. Also included are descriptions of the elements used to build custom reports and dashboards, including styles, chart styles, color schemes, images, and text fields.

RightNow Service Administrator Manual—Contains procedures for configuring RightNow Service. This manual addresses service level agreements, standard text and answer variables in the content library, product linking, the end-user interface, Offer Advisor, RightNow Live, RightNow Wireless, and incident archiving.

RightNow Service User Manual—Contains procedures for customer service agents working with RightNow Service. This manual addresses incidents, incident archiving, Offer Advisor, RightNow Live, answers, the accessibility interface, and the end-user interface.

Standalone End-User Manual—Contains a description of all the pages on the end-user interface in RightNow Service and the unique features on each page. This standalone manual describes the 8.2 end-user interface, which is an option in RightNow February '08.

RightNow Marketing User Manual—Contains procedures for staff members working with RightNow Marketing. This manual addresses audiences, the content library, mailings, and campaigns.

RightNow Sales Administrator and User Manual—Contains procedures for the RightNow administrator and all staff members working with RightNow Sales. Procedures for the RightNow administrator include adding sales process options and quote templates, and configuring Outlook integration and disconnected access. Procedures for sales staff members include working with opportunities, quotes, Outlook integration, and disconnected access.

RightNow Feedback User Manual—Contains procedures for all staff members working with RightNow Feedback. This manual addresses audiences, the content library, questions, and surveys.

RightNow Made Easy: An Administrator's How-To Guide—Contains basic procedures for the RightNow administrator to configure all common areas in RightNow and all RightNow products, including RightNow Service, RightNow Marketing, RightNow Sales, and RightNow Feedback. This streamlined how-to guide gives administrators the basic steps to set up and configure all areas in RightNow, one task at a time, and complements the array of published RightNow manuals and documentation.

RightNow Made Easy: A User's How-To Guide—Contains the basic procedures for tasks that staff members perform on a regular or daily basis. With how-to instructions for each RightNow product, customer service agents, marketing personnel, and sales representatives can quickly and efficiently complete routine tasks as they work with customers and prospects. The user's how-to guide combines several RightNow user manuals into one easy-to-use resource.

RightNow February '08 Release Notes—Contains a brief description of the new and expanded features in RightNow February '08, including features common to all products and those specific to RightNow Service, RightNow Marketing, RightNow Sales, and RightNow Feedback.

RightNow HMS Guide—Contains upgrade instructions for customers hosted by RightNow Technologies.

RightNow SmartConversion Guide—Contains procedures for upgrading from RightNow CRM 7.5 to RightNow February '08.

RightNow Smart Client Installation Guide—Contains procedures for installing the RightNow Smart Client on staff members' workstations using the RightNow Click-Once installer or the RightNow Smart Client Setup Wizard.

RightNow Integration Manual—Contains procedures for integrating the RightNow knowledge base with external systems, including help desks, data mining, and data reporting systems. Contact your RightNow account manager to obtain this manual.

Tip	For a comprehensive list of all RightNow documentation, refer to http://community.rightnow.com/customer/documentation.

2

Integration Overview

RightNow has all the tools you need to create a fully integrated customer service solution. There are three ways to integrate RightNow with other applications:

- XML API
- Event handlers
- Pass-through authentication

This overview provides a brief description of each integration method to assist you in deciding which method best suits your integration needs. For detailed information about the types of integration, refer to each method's chapter in the manual.

You must be a non-hosted customer to implement event handlers. If you are a hosted customer, you must contact your RightNow account manager to perform these functions. Both hosted and non-hosted customers may contact their account manager for assistance from Professional Services in planning and implementing an integration. To learn more about the services provided, visit our web site at:

```
http://rightnow.com
```

To follow the procedures in this manual, on-premise customers must be using the latest version of RightNow, available for download on our web site.

Caution The API functions should be used by experienced programmers only. Misuse of the API could result in damage to your RightNow site or database. We recommend that you first test your integration on a non-production site. If you require assistance, contact your RightNow account manager.

XML API

XML integration allows you to interact directly with the API through the use of XML. Using XML integration, you can create, update, delete, get, and search on accounts, answers, contacts, hierarchical menus, incidents, meta-answers, opportunities, organizations, quotes, SLA instances, and task instances in your RightNow database. You can also run SQL queries on any table of your RightNow database to retrieve information.

RightNow provides two methods for accessing the XML API. You can use HTTP POST or send an XML-formatted email to perform XML API functions. Posting XML allows real-time interaction with RightNow.

Use XML integration when you want direct access to the RightNow database. XML integration can also be used when an external application has the ability to create and send XML-formatted emails or post XML directly to RightNow. You should have experience with XML and familiarity with RightNow functions before attempting to perform an XML integration. For more information, refer to Chapter 3, "XML Integration," on page 17.

Event handlers

An event handler is implemented when a specific event occurs within RightNow. An event handler can either execute a script (external event) or email data (email integration) to a specified email address when the event occurs. The following events are supported:

- An incident is created, updated, or deleted
- An answer is created, updated, or deleted
- A contact is created, updated, or deleted
- An organization is created, updated, or deleted
- An opportunity is created, updated, or deleted
- A business rule is matched

These events facilitate execution of a program or email transmission when information is modified within RightNow. The external event program is passed the data related to the update. For example, this function could be used to update contact information in an external system every time contact information is updated within RightNow.

Use event handlers when you want real-time synchronization with an external system or want to update data external to RightNow. Using external events requires programming experience and familiarity with RightNow functions. For more information, refer to Chapter 4, "Event Handlers," on page 101.

Pass-through authentication

You can integrate RightNow Service with an external customer validation source to allow your end-users to automatically log in to RightNow Service from an external web page by passing the necessary login parameters in the URL of any appropriate end-user page (home.php, std_alp.php, std_adp.php). By using this integration method, you can allow contacts to have one login name and password for RightNow Service, as well as an external system.

Use this integration method when you want to use an external customer validation source to log in contacts to RightNow Service. This method requires programming experience and familiarity with RightNow Service functions. For more information, refer to Chapter 5, "Pass-Through Authentication," on page 109.

3

XML Integration

You can use XML (Extensible Markup Language) to access RightNow's API and update the database. Through XML integration, you can perform many tasks normally accomplished through the RightNow user interface, such as creating, updating, deleting, retrieving and searching records in your RightNow database using either of the following methods:

- **Sending XML data using the POST method**—When posting data using this method, the XML is immediately passed to RightNow and parsed by a PHP script. A record is then instantly created, updated, deleted, retrieved, or searched for in the RightNow database. For additional information, refer to "Using the POST method" on page 94.
- **Sending an XML-formatted email**—When sending an XML-formatted email, the utility *techmail* will identify an email as having XML through a trigger word or phrase in the subject line. The email will then be parsed by a PHP script to retrieve the data. For additional information, refer to "Sending an XML-formatted email" on page 95.

Caution The XML API functions should be used by experienced programmers only. Misuse of the API could result in damage to your RightNow site or database. We recommend that you first test your integration on a non-production site. If you require assistance, please contact your RightNow account manager.

This chapter describes the XML tags used by the RightNow API, provides descriptions of the basic API functions, and contains information on posting XML through a URL or sending an XML-formatted email.

XML tags

The data sent to RightNow is identified by a series of XML tags defined in this chapter. The tags organize data so it can be parsed by RightNow and handled appropriately. There are four basic tags used when accessing the API through XML:

- <connector>
- <function>
- <parameter>
- <pair>

Each tag is used in the following example code:

```
<connector>
    <function name="incident_update" id="incident_id">
        <parameter name="args">
            <pair name="i_id" type="integer">19283</pair>
            <pair name="ref_no" type="string">061031-000227</pair>
            <pair name="status" type="pair">
                <pair name="id" type="integer">4</pair>
                <pair name="type" type="integer">4</pair>
            </pair>
            <pair name="updated" type="time">1164451523</pair>
            <pair name="updated_by" type="integer">23</pair>
            <pair name="use_smime" type="integer">1</pair>
            <pair name="wf_flag" type="integer">0</pair>
       </parameter>
  </function>
</connector>
```

These tags are described in the following sections, along with descriptions of their attributes and types.

<connector> tag

The <connector> tag is the root element of the XML code. It contains all function tags. The <connector> tag can use the following attributes:

- **ret_type**—This attribute specifies either http or email as the type of return. For example:
  `<connector ret_type="http">`
  Or

```
<connector ret_type="email">
```

If the ret_type is set to http, the XML return value will be sent to the http requester. If the ret_type is set to email, an email containing the return value will be sent. If no ret_type is specified, http will be used by default.

Note Specifying `ret_type="http"` does not allow you to send an XML return to a specific http location or URL.

- **ret_email**—This attribute specifies the email address to send return values to if ret_type is email. For example:

```
<connector ret_type="email" ret_email="jdoe@example.com">
```

When return values are sent to an email address or URL, they appear in the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<connector_ret>
     <function name="incident_create">
          <ret_val name="i_id">7345</ret_val>
     </function>
</connector_ret>
```

This example returns the i_id of an incident created through the API with the incident_create function. The automatically assigned i_id of the new incident, 7345, is specified by the <ret_val> tag.

<function> tag

The <function> tag contains each API call and contains the following attributes:

- **name**—This attribute specifies the name of the API function you want to call. For example:

```
<function name="incident_create">
```

- **id**—This attribute specifies a string used to apply return values to. The string can be used later to have the return value replace a variable. For example:

```
<function name="contact_create" id="contact_id">
```

For information about using variables in the ID attribute, refer to "Passing variable IDs" on page 87.

<parameter> tag

The parameters described in Table 1 can be specified using the name attribute in the <parameter> tag. The datatype should also be specified using the type attribute. For example:

```
<parameter name="args" type="pair"> <pair_data> </parameter>
```

Table 1: Parameter Description

Parameter	Description
ac_id	This parameter defines the report ID number in the search function.

args	This parameter indicates that pair data will follow the <parameter> tag. For example:
	`<parameter name="args" type="pair">`
	`  <pair name="i_id" type="integer">19283</pair>`
	`</parameter>`

max_rows	This parameter defines the maximum number of records that should be returned by the search when using the search API functions.
sql	This parameter defines the SQL statement for the SQL query API functions.

<pair> tag

The <pair> tag contains data used by the API function. It describes the database field and the value to add to the RightNow database. The <pair> tag can have the following attributes:

- **name**—This attribute defines the pair name that the enclosed data pertains to. Pair names for the API are described in Appendix A, "Pair Names," on page 125. For example:

  ```
  <pair name="title" type="string">Title</pair>
  ```

- **type**—This attribute determines whether the pair is a pair, integer, time, or string. For example:

  ```
  <pair name="c_id" type="integer">3</pair>
  ```

Table 2 describes the four pair types.

Table 2: Pair Type Description

Type	Description
integer	A positive or negative 4-byte integer.
string	A string of characters that cannot contain any NULLs.
time	A field that is the same type as the UNIX date_t, generally a long integer that is the number of seconds since the UNIX Epoch date (00:00:00 UTC January 1, 1970).
pair	A pair that contains additional data within a pair; also called a "nested pair."

Using special characters

When passing data through XML, there are certain characters that cannot be used because they are misinterpreted by the XML language as it is parsed in RightNow. These special characters should always be encoded when used as a parameter value in your XML code. Table 3 shows the special characters and their required format.

Table 3: Special Characters

Character	Format in XML
&	&amp;
"	&quot;
’	' or '
<	&lt;
>	&gt;

XML API functions

When using XML to integrate RightNow with an external system, you can use several API functions to perform actions on accounts, answers, contacts, hierarchical menus, incidents, meta-answers, opportunities, organizations, quotes, and tasks. An XML API function is also available for searching in RightNow.

Table 4 lists the available XML API functions along with a description of the function and its required parameters and pairs.

Table 4: XML API Functions

Function	Description	Required Parameters/Pairs
Account Functions		
acct_create	An account API function used to add an account to the database. Refer to page 30.	• The args parameter • An array of pair data containing name, login, display_name, profile_id, def_curr_id, seq, and country_id pairs
acct_destroy	An account API function used to delete an account from the database. Refer to page 31.	• The args parameter • An array of pair data containing seq, group_id, and acct_id pairs
acct_move	An account API function used to move an account in the database. Refer to page 32.	• The args parameter • An array of pair data containing id, oldseq, newseq, oldparent, and np_lvl_id pairs
acct_update	An account API function used to update an existing account in the database. Refer to page 33.	• The args parameter • An array of pair data containing the acct_id pair

Table 4: XML API Functions (Continued)

Function	Description	Required Parameters/Pairs
Answer Functions		
ans_create	An answer API function used to add an answer to the database. Refer to page 34.	• The args parameter • An array of pair data containing summary, status, access_mask, type, and lang_id pairs
ans_destroy	An answer API function used to delete answer data from the database. Refer to page 38.	• The args parameter • An array of pair data containing the a_id pair
ans_get	An answer API function used to retrieve an answer from the database. Refer to page 38.	• The args parameter • An array of pair data containing the a_id pair
ans_update	An answer API function used to update an existing answer in the database. Refer to page 39.	• The args parameter • An array of pair data containing the a_id pair
Contact Functions		
contact_create	A contact API function used to add a contact to the database. Refer to page 40.	• The args parameter • An array of pair data containing the state pair
contact_destroy	A contact API function used to delete a contact from the database. Refer to page 42.	• The args parameter • An array of pair data containing the c_id pair
contact_get	A contact API function used to retrieve a contact from the database. Refer to page 42.	• The args parameter • An array of pair data containing the c_id pair
contact_update	A contact API function used to update an existing contact in the database. Refer to page 43.	• The args parameter • An array of pair data containing the c_id pair

Table 4: XML API Functions (Continued)

Function	Description	Required Parameters/Pairs
mailing_send_to_contact	A contact API function used with contact_create or contact_update to send a new or updated contact a transactional mailing or survey. Refer to page 43. **Note:** This function should only be used if RightNow Marketing or RightNow Feedback is enabled.	• The c_id and mailing_id pairs
Flow Function		
flow_execute	A flow API function used to enter contacts in campaign flows at specified entry points. Refer to page 44. **Note:** This function should only be used if RightNow Marketing is enabled.	• The args parameter and the c_id, flow_id, and shortcut pairs
Hierarchical Menu Functions		
css_category_create	A hierarchical menu API function used to create category customizable menu items in the database. Refer to page 45.	• The args parameter • An array of pair data containing, seq, label, lbl_item, parent, lvl_id (1-6), desc, and vis pairs
css_disposition_create	A hierarchical menu API function used to create disposition customizable menu items in the database. Refer to page 45.	• The args parameter • An array of pair data containing, seq, label, lbl_item, parent, lvl_id (1-6), desc, and vis pairs
css_product_create	A hierarchical menu API function used to create product customizable menu items in the database. Refer to page 45.	• The args parameter • An array of pair data containing, seq, label, lbl_item, parent, lvl_id (1-6), desc, and vis pairs

Table 4: XML API Functions (Continued)

Function	Description	Required Parameters/Pairs
css_category_destroy	A hierarchical menu API function used to destroy category customizable menu items in the database. Refer to page 47.	• The args parameter • An array of pair data containing id, seq, and parent pairs
css_disposition_destroy	A hierarchical menu API function used to destroy disposition customizable menu items in the database. Refer to page 47.	• The args parameter • An array of pair data containing id, seq, and parent pairs
css_product_destroy	A hierarchical menu API function used to destroy product customizable menu items in the database. Refer to page 47.	• The args parameter • An array of pair data containing id, seq, and parent pairs
css_category_move	A hierarchical menu API function used to move category customizable menu items in the database. Refer to page 47.	• The args parameter • An array of pair data containing id, old_seq, new_seq, old_parent, np_lvl_id, lvl_id, old_lvl, and new_lvl pairs
css_disposition_move	A hierarchical menu API function used to move disposition customizable menu items in the database. Refer to page 47.	• The args parameter • An array of pair data containing id, old_seq, new_seq, old_parent, np_lvl_id, lvl_id, old_lvl, and new_lvl pairs
css_product_move	A hierarchical menu API function used to move product customizable menu items in the database. Refer to page 47.	• The args parameter • An array of pair data containing id, old_seq, new_seq, old_parent, np_lvl_id, lvl_id, old_lvl, and new_lvl pairs

Table 4: XML API Functions (Continued)

Function	Description	Required Parameters/Pairs
css_category_ update	A hierarchical menu API function used to update an existing category customizable menu in the database. Refer to page 48.	• The args parameter • An array of pair data containing id and parent pairs.
css_disposition_ update	A hierarchical menu API function used to update an existing disposition customizable menu in the database. Refer to page 48.	• The args parameter • An array of pair data containing id and parent pairs.
css_product_ update	A hierarchical menu API function used to update an existing product customizable menu in the database. Refer to page 48.	• The args parameter • An array of pair data containing id and parent pairs.
Incident Functions		
incident_create	An incident API function used to add an incident to the database. Refer to page 49.	• The args parameter • An array of pair data containing subject, status, interface_id, lang_id, and contact pairs
incident_destroy	An incident API function used to delete an incident from the database. Refer to page 51.	• The args parameter • An array of pair data containing the i_id pair
incident_get	An incident API function used to retrieve a specific incident from the database. Refer to page 52.	• The args parameter • An array of pair data containing the i_id pair
incident_update	An incident API function used to update an existing incident in the database. Refer to page 52.	• The args parameter • An array of pair data containing the i_id pair

Table 4: XML API Functions (Continued)

Function	Description	Required Parameters/Pairs
Lookup Function		
lookup_id_for_name	A function used to look up the code number of a field in RightNow. Refer to page 91.	• The args parameter • An array of pair data containing lk_tbl and lk_str pairs
Meta-Answer Functions		
meta_ans_create	A meta-answer API function used to add a meta-answer to the database. Refer to page 53.	• The args parameter • An array of pair data
meta_ans_destroy	A meta-answer API function used to delete meta-answer data from the database. Refer to page 55.	• The args parameter • An array of pair data containing the m_id pair
meta_ans_update	A meta-answer API function used to update an existing meta-answer in the database. Refer to page 56.	• The args parameter • An array of pair data containing the m_id pair
Opportunity Functions		
opp_create	An opportunity API function used to add an opportunity to the database. Refer to page 57.	• The args parameter • An array of pair data containing the name and status pairs
opp_destroy	An opportunity API function used to delete an opportunity from the database. Refer to page 59.	• The args parameter • An array of pair data containing the op_id pair
opp_get	An opportunity API function used to retrieve an opportunity from the database. Refer to page 59.	• The args parameter • An array of pair data containing the op_id pair
opp_update	An opportunity API function used to update an existing opportunity in the database. Refer to page 60.	• The args parameter • An array of pair data containing the op_id pair

Table 4: XML API Functions (Continued)

Function	Description	Required Parameters/Pairs
Organization Functions		
org_create	An organization API function used to add an organization to the database. Refer to page 64.	• The args parameter • An array of pair data containing the name and state pairs
org_destroy	An organization API function used to delete an organization from the database. Refer to page 65.	• The args parameter • An array of pair data containing the org_id pair
org_get	An organization API function used to retrieve an organization from the database. Refer to page 65.	• The args parameter • An array of pair data containing the org_id pair
org_update	An organization API function used to update an existing organization in the database. Refer to page 66.	• The args parameter • An array of pair data containing the org_id pair
Purchased Product Function		
pur_prod_create	A purchased product API function used to create purchased products in the database for use by RightNow Sales and Offer Advisor. Refer to page 66.	• The args parameter • An array of pair data containing the label and oa_exclude pairs
Sales Product Functions		
sa_prod_create	A sales product API function used to create sales products in the database for use by RightNow Sales and Offer Advisor. Refer to page 68.	• The args parameter • An array of pair data containing the desc, label, disabled, and seq pairs
sa_prod_destroy	A sales product API function used to delete sales products in the database. Refer to page 69.	• The args parameter • An array of pair data containing the product_id and seq pairs

Table 4: XML API Functions (Continued)

Function	Description	Required Parameters/Pairs
sa_prod_update	A sales product API function used to update existing sales products in the database. Refer to page 69.	• The args parameter • An array of pair data containing the product_id pair
Search Function		
search	A search API function used to search for any records in the database using an existing view. Refer to page 70.	• The args parameter • An array of pair data containing the ac_id pair
SQL Query Functions		
sql_get_int	A SQL query API function used to retrieve an integer value from the database. Refer to page 74.	• SQL parameter and statement
sql_get_str	A SQL query API function used to retrieve a string from the database. Refer to page 75.	• SQL parameter and statement
sql_get_dttm	A SQL query API function used to retrieve a datetime value from the database. Refer to page 76.	• SQL parameter and statement
Task Functions		
task_create	A task API function used to add a task to the database. Refer to page 77.	• The args parameter • An array of pair data containing assgn_acct_id, tbl, status, and name pairs
task_destroy	A task API function used to delete a task from the database. Refer to page 78.	• The args parameter • An array of pair data containing the task_id pair
task_get	A task API function used to retrieve a task from the database. Refer to page 78.	• The args parameter • An array of pair data containing the id pair

Table 4: XML API Functions (Continued)

Function	Description	Required Parameters/Pairs
task_update	A task API function used to update an existing task in the database. Refer to page 78.	• The args parameter • An array of pair data containing the task_id pair

Unless otherwise specified in "Error codes" on page 96, positive return values indicate success, and negative return values indicate an error. If an invalid parameter is used with the XML API get functions, a blank return value will be returned.

Note The XML API get functions do not return data fields with NULL values.

Account API

The account API functions (acct_create, acct_destroy, acct_move, and acct_update) allow you to create, delete, move, or update information within the *accounts* table. You can act on all standard database fields of the *accounts* table, as well as some specialized information, such as staff account custom fields.

To access staff accounts from the RightNow Console, click Common Configuration and double-click Staff Accounts under Staff Management.

acct_create

The acct_create function is used to add a staff account to the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Important The API will automatically generate an acct_id for the account that is consistent with existing accounts in the database.

The account will be populated with data specified in the pair list. A brief description of all *accounts* table fields and their corresponding pair names can be found in Appendix A, "Pair Names," on page 125.

Example:

```
<connector>
<function name="acct_create">
     <parameter name="args" type="pair">
```

```
          <pair name="name" type="pair">
               <pair name="first" type="string">Chad</pair>
               <pair name="last" type="string">Jones</pair>
          </pair>
          <pair name="display_name" type="string">Chad Jones</pair>
          <pair name="email" type="pair">
               <pair name="addr" type="string">cj@example.com</pair>
          </pair>
          <pair name="def_curr_id" type="integer">1</pair>
          <pair name="seq" type="integer">3</pair>
          <pair name="profile_id" type="integer">2</pair>
          <pair name="country_id" type="integer">1</pair>
          <pair name="notif_pending" type="integer">0</pair>
          <pair name="group_id" type="integer">1</pair>
          <pair name="login" type="string">cjones</pair>
          <pair name="attr" type="integer">0</pair>
     </parameter>
</function>
</connector>
```

This example creates an account in the *accounts* table and the account ID is automatically returned by the function.

acct_destroy

The acct_destroy function is used to delete an existing account in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="acct_destroy">
     <parameter name="args" type="pair">
          <pair name="acct_id" type="integer">3</pair>
          <pair name="seq" type="integer">3</pair>
          <pair name="group_id" type="integer">100061</pair>
     </parameter>
</function>
</connector>
```

This example deletes account ID 3 from the database.

acct_move

The acct_move function is used to move an account from one group to another and to change the sequence of an account within a group. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs. The type attribute for each of these parameters is integer. The value of old_lvl and new_lvl must be 2.

Example:

```
<connector>
<function name="acct_move">
     <parameter name="args" name="pair">
          <pair name="id" type="integer">16</pair>
          <pair name="old_lvl" type="integer">2</pair>
          <pair name="new_lvl" type="integer">2</pair>
          <pair name="old_seq" type="integer">5</pair>
          <pair name="new_seq" type="integer">2</pair>
          <pair name="old_parent" type="integer">100061</pair>
          <pair name="np_lvl_id" type="pair">
               <pair name="lvl_id1" type="integer">100068</pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example moves account ID 16 from the fifth position in the hierarchy to the second position in the hierarchy and reassigns it from group 100061 to group 100068. Group IDs correspond to folder IDs in the *folders* table.

Note If an account is referenced by a rule, it cannot be moved.

acct_update

The acct_update function is used to update the information associated with an existing staff account in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Note If RightNow Sales is enabled and you want to change an account's territory, you must pass the old territory with the old_terr pair and the new territory ID with the terr_id pair. You will also need to specify whether to update the account's opportunities by setting the upd_opt pair to one of the following values.

1—Update all opportunities

2—Update only active opportunities

3—Update no opportunities

The API will set any fields supplied in the pair list. Any staff account fields missing from the pair list will not be altered in the database.

Example:

```
<connector>
<function name="acct_update" id="myid">
     <parameter name="args" name="pair">
          <pair name="acct_id" type="integer">22</pair>
          <pair name="name" type="pair">
               <pair name="last" type="string">Moore</pair>
          </pair>
          <pair name="display_name" type="string">Susan Moore</pair>
          <pair name="email" type="pair">
               <pair name="addr" type="string">smoore@example.com
               </pair>
          </pair>
          <pair name="login" type="string">smoore</pair>
          <pair name="custom_field" type="pair">
               <pair name="cf_item1" type="pair">
                    <pair name="cf_id" type="integer">1</pair>
                    <pair name="val_str" type="string">1096045800
                    </pair>
               </pair>
               <pair name="cf_item2" type="pair">
                    <pair name="cf_id" type="integer">2</pair>
```

```
                    <pair name="val_str" type="string">Updated to
                    change spelling of last name.</pair>
                </pair>
            </pair>
        </parameter>
</function>
</connector>
```

This example updates the account with an acct_id of 22 to have a last name of "Moore," a display name of "Susan Moore," an email address of "smoore@example.com," and a login of "smoore." It also updates two custom fields. The codes for custom field types are located in Table 12 on page 80.

Answer API

The answer API functions (ans_create, ans_destroy, ans_get, and ans_update) allow you to create, delete, retrieve, or update information from the *answers* table. You can act on all standard database fields of the *answers* table, as well as some specialized information, such as answer custom fields.

Answers can be created and managed through the RightNow Console, and may be designated for viewing by end-users. On the end-user pages, answers that have been designated for viewing by end-users appear on the Find Answers and Answer pages.

ans_create

The ans_create function is used to add an answer to the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Important The API will automatically generate an a_id for the answer that is consistent with existing answers in the database.

The answer will be populated with data specified in the pair list. A brief description of all *answers* table fields and their corresponding pair names can be found in Appendix A, "Pair Names," on page 125.

If you want to assign access levels to the answer, you must pass them through the access_mask pair. The access mask pair is a decimal value computed from a binary bitmap of your access levels. Use the following steps to compute access_mask.

To compute the access mask:

1 Run the following SQL query to determine the maximum value for access IDs in your database:

```
SELECT max(access_id) FROM ans_access
```

This is the number of digits in the binary bitmap of your access levels.

2 Run the following SQL query to display the access levels in your database:

```
SELECT a.access_id, l.label FROM ans_access a, labels l WHERE
l.label_id=a.access_id AND l.fld = 1 and l.tbl = 11
```

The query will return a set of data like the example in Table 5.

Table 5: Example SQL Data Set

access_id	label
1	Everyone
2	Help
3	Human Resources
6	Admin–Finance
7	Information Systems
32	Product Development
33	Hosting
34	Product Management
63	Sales–NW
65	Sales–SW
66	Sales–NE
67	Sales–SE
94	Marketing
95	Technical Publications

3 Create a four grids with three columns. The first grid will contain the access IDs from one to 31. The second grid will contain the access IDs from 32 to 63. The third grid will contain the access IDs from 64 to 93. The fourth grid will contain the access IDs from 94 to

124. If you want the answer to be viewable by end-users with the access level, place a one in the third column next to that access level. Place zeros in the remaining cells in the third row as shown in the following grids.

Table 6: Example Grid

Access ID	Access Level	On/Off
1	Everyone	0
2	Help	0
3	Human Resources	1
4	Not Used	0
5	Not Used	0
6	Admin–Finance	1
7	Information Systems	0

Table 7: Example Grid

Access ID	Access Level	On/Off
32	Product Development	0
33	Product Management	1
34	Hosting	0

Table 8: Example Grid

Access ID	Access Level	On/Off
63	Sales–NW	0
64	Not Used	0
65	Sales–SW	0
66	Sales–NE	0
67	Sales–SE	0

Table 9: Example Grid

Access ID	Access Level	On/Off
94	Marketing	1
95	Technical Publications	0

In this example, the answer would be viewable by end-users with the following access levels: Human Resources, Admin–Finance, Product Management, and Marketing.

4 For each grid, use the values in column 3, from bottom to top to create a binary number. The binary numbers in this example are 0100100, 010, 0000, and 01.

5 Convert the binary number to decimal. In this example the decimal values are 36, 2, 0, and 1.

6 Add leading zeros to the decimal values to make them ten-digit numbers and place them one after the other. For example, 0000000360000000002000000000000000001. This is the value of access_id.

Example:

```
<connector>
<function name="ans_create">
     <parameter name="args" type="pair">
          <pair name="access_mask" type="string">
          0000000360000000002000000000000000001</pair>
          <pair name="assigned" type="pair">
               <pair name="acct_id" type="integer">4</pair>
               <pair name="group_id" type="integer">100061</pair>
          </pair>
          <pair name="type" type="integer">1</pair>
          <pair name="m_id" type="integer">25</pair>
          <pair name="lang_id" type="integer">1</pair>
          <pair name="description" type="string">How do I send a
          picture with my new camera phone?</pair>
          <pair name="summary" type="string">How do I send a
          picture with my new camera phone?</pair>
          <pair name="solution" type="string">Dear Valued Customer:
          Simply use the picture taking ability of your camera phone
          to take a photo. You can save the pictures on your phone
          up to the limit of the storage space on your camera and
```

```
            then send them to any World Mobile customer, upload them to
            your online photo album at World Mobile, or send them to an
            email address. To send to another World Mobile member, dial
            their number and select the Send Picture option on
            your phone.</pair>
            <pair name="status" type="pair">
                 <pair name="id" type="integer">4</pair>
                 <pair name="type" type="integer">4</pair>
            </pair>
      </parameter>
</function>
</connector>
```

This example creates an answer associated with meta-answer ID 25, sets the access mask, description, summary, solution, language, and status fields for the answer, and assigns it to the staff member with the account ID of 4. The function will return the a_id number.

ans_destroy

The ans_destroy function is used to delete an existing answer in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="ans_destroy">
      <parameter name="args" type="pair">
            <pair name="a_id" type="integer">33</pair>
            <pair name="wf_flag" type="integer">1</pair>
            <pair name="ee_flag" type="integer">1</pair>
      </parameter>
</function>
</connector>
```

This example deletes answer ID number 33, applies business rules, and executes an external event if one is specified in the EE_ANS_DELETE_HANDLER configuration setting.

ans_get

The ans_get function is used to retrieve the contents of an answer from the *answers* table. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="ans_get">
     <parameter name="args" type="pair">
          <pair name="id" type="integer">33</pair>
          <pair name="sub_tbl" type='pair'>
               <pair name="tbl_id" type="integer">164</pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example retrieves the details for the answer with ID number 33 from the database.

Note The XML API ans_get function does not return data fields with NULL values.

ans_update

The ans_update function is used to update the information associated with an existing answer in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs. The API will set any fields supplied in the pair list, including custom fields. Any answer fields missing from the pair list will not be altered in the database.

If you want to assign access levels to the answer, you must pass them through the access_mask pair. The access mask pair is a decimal value computed from a binary bitmap of your access levels. For more information, refer to the procedure "To compute the access mask:" on page 35.

Example:

```
<connector>
<function name="ans_update">
     <parameter name="args" type="pair">
          <pair name="a_id" type="integer">15</pair>
          <pair name="status" type="pair">
               <pair name="id" type="integer">4</pair>
               <pair name="type" type="integer">4</pair>
          </pair>
          <pair name="access_mask" type="string">
          00000000036000000000200000000000000000002</pair>
          <pair name="wf_flag" type="integer">0</pair>
     </parameter>
```

```
</function>
</connector>
```

This example updates the answer with an a_id of 15 to have a status and status type of Public (code 4) and an access level of Everyone. Business rules will not be run for the answer update.

Contact API

The contact API functions (contact_create, contact_destroy, contact_get, and contact_update) allow you to create, delete, retrieve, or update information from the *contacts* table. You can act on all standard database fields of the *contacts* table, as well as some specialized information, such as contact custom fields.

A contact is a customer who has a record in your RightNow knowledge base. Contacts have a customer account which allows them to log in and submit incidents, subscribe to answer notifications, and view their recently submitted incidents.

Note Note entries in contacts use a unique pair structure. For information, refer to "Adding Notes" on page 85.

contact_create

The contact_create function is used to add a contact to the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

The contact will be populated with data specified in the pair list. A brief description of all *contacts* table fields and their corresponding pair names can be found in Appendix A, "Pair Names," on page 125. For a listing of sources, refer to Appendix B, "Source Codes," on page 177.

Note For records created or updated through the XML API, you should always allow the API to set the source levels. The API will automatically set source_lvl1 to 32007 and source_lvl2 to 6001. Setting the sources to other values can have a detrimental effect on your data. If you choose to set sources to other values, be sure you carefully test your work and the results.

You must include a first name, a last name, and the states for the contact in your XML. The email address of the contact is not required under certain configurations, such as call center applications. In this case, enabling TC_CT_EMAIL_REQD would require the email address when using the contact_create function.

Example:

```
<connector>
<function name="contact_create">
     <parameter name="args" type="pair">
          <pair name="name" type="pair">
               <pair name="first" type="string">Joe</pair>
               <pair name="last" type="string">Smith</pair>
          </pair>
          <pair name="email" type="pair">
               <pair name="addr" type="string">js@example.com</pair>
               <pair name="cert" type="string"></pair>
          </pair>
          <pair name="state" type="pair">
               <pair name="css" type="integer">1</pair>
               <pair name="ma" type="integer">1</pair>
               <pair name="sa" type="integer">0</pair>
          </pair>
          <pair name="ee_flag" type="integer">1</pair>
          <pair name="note" type="pair">
               <pair name="note_item1" type="pair">
                    <pair name="action" type="integer">1</pair>
                    <pair name="seq" type="integer">1</pair>
                    <pair name="text" type="string">Created through
                    the XML API</pair>
               </pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example creates a contact, Joe Smith, who is associated with a Service state and a Marketing state, but not with a Sales state. An external event is executed if one is specified in the EE_CONTACT_INSERT_HANDLER configuration setting. A note will be added to the contact record. This function will return the contact ID from the database.

contact_destroy

The contact_destroy function is used to delete an existing contact in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Important Deleting a contact also results in the deletion of all incidents and opportunities associated with the contact.

Example:

```
<connector>
<function name="contact_destroy">
     <parameter name="args" type="pair">
          <pair name="c_id" type="integer">94</pair>
          <pair name="ee_flag" type="integer">1</pair>
     </parameter>
</function>
</connector>
```

This example deletes the contact with contact ID number 94 from the database and executes an external event if one is specified in the EE_CONTACT_DELETE_HANDLER configuration setting.

contact_get

The contact_get function is used to retrieve a record from the *contacts* table. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="contact_get">
     <parameter name="args" type="pair">
          <pair name="id" type="integer">9</pair>
          <pair name="sub_tbl" type="pair">
               <pair name="tbl_id" type="integer">164</pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example retrieves the contact details and notes (table ID of 164) for the contact with ID number 9.

Note The XML API contact_get function does not return data fields with NULL values.

contact_update

The contact_update function is used to update the information associated with an existing contact in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

The API will set any fields supplied in the pair list, including custom fields. Any contact fields missing from the pair list will not be altered in the database.

Example:

```
<connector>
<function name="contact_update">
     <parameter name="args" type="pair">
          <pair name="c_id" type="integer">9</pair>
          <pair name="password" type="string">newpassword</pair>
          <pair name="email" type="pair">
               <pair name="addr" type="string">mib@test.com</pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example updates the contact with ID number 9. The contact's password is changed to newpassword, and the email address is changed to mib@test.com.

mailing_send_to_contact

The mailing_send_to_contact function is available when RightNow Marketing or RightNow Feedback is enabled. This function can be used to send a transactional mailing or survey to a contact. This function has four parameters: c_id, mailing_id, flow_id, and scheduled. Refer to Table 4 on page 22 for a list of required parameters.

Example:

```
<connector>
<function name="mailing_send_to_contact">
     <parameter name="args" type="pair">
          <pair name="c_id" type="integer">84</pair>
```

```
          <pair name="mailing_id" type="integer">12</pair>
          <pair name="flow_id" type="integer">5</pair>
          <pair name="scheduled"  type="time">1207571400</pair>
          <pair name='trigger' type='pair'>
               <pair name='id' type='integer'>207</pair>
               <pair name='tbl' type='integer'>1</pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example would send the transactional mailing or survey with the ID of 12 to the contact with the ID of 84 and associate it with the campaign flow with the ID of 5 and the incident with the ID of 207. The email will be sent the first time the mailer daemon runs after 12:30 PM on April 7, 2007 (UNIX timestamp of 1207571400).

Note If the scheduled pair is not included, the mailing or survey will be sent the next time the mailer daemon runs.

Flow API

The flow API function flow_execute allows you to enter contacts into campaign flows. Flows are multistep processes based on business logic and are used in marketing campaigns and surveys.

flow_execute

The flow_execute function can be used when RightNow Marketing is enabled. The function is used to send a specified contact through a campaign flow starting at a flow entry point. This function has two components: the args parameter and an array of pairs. The shortcut parameter corresponds to the Shortcut ID field of an entry point in a campaign flow. A flow may have multiple entry points; however, each entry point's shortcut ID will be unique. Refer to Table 4 on page 22 for a list of required parameters and pairs.

Example:

```
<connector>
<function name="flow_execute">
     <parameter name="args" type="pair">
          <pair name="c_id" type="integer">2</pair>
          <pair name="flow_id" type="integer">4</pair>
          <pair name="shortcut" type="string">EP1</pair>
     </parameter>
```

```
</function>
</connector>
```

This example sends the contact with an ID of 2 through the flow with an ID of 4, starting at the entry point named EP1.

Hierarchical menu API

The hierarchical menu API functions (css_product_<action>, css_category_<action>, and css_disposition_<action>) allow you to create, delete, move, and update customizable menus in RightNow. For example, in RightNow Service you can act on products, categories, or dispositions. You can act on all standard database fields of the *hier_menus* table.

The hierarchical menu will be populated with data specified in the pair list. A brief description of all *hier_menus* table fields and their corresponding pair names can be found in Appendix A, "Pair Names," on page 125.

css_product_create, css_category_create, and css_disposition_create

The hierachical menu create functions (css_product_create, css_category_create, and css_disposition_create) are used to add hierarchical menu items to the RightNow database. The parameter and pair structures of the three create functions are identical. The functions have two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Important The API will automatically generate an ID for the hierarchical menu item that is consistent with existing objects in the database.

Because the parameter and pair structures of the three create functions are identical, only one example is provided: css_product_create. The only change required for css_category_create and css_disposition_create is the function name.

Example:

```
<connector>
<function name="css_product_create">
     <parameter name="args" type="pair">
          <pair name="parent" type="pair">
               <pair name="lvl_id1" type="integer">1</pair>
               <pair name="lvl_id2" type="integer">2</pair>
          </pair>
          <pair name="seq" type="integer">1</pair>
          <pair name="label" type="pair">
```

```
                <pair name="lbl_item" type="pair">
                    <pair name="label" type="string">Calling Plans
                    </pair>
                    <pair name="lang_id" type="integer">1</pair>
                </pair>
                <pair name="lbl_item" type="pair">
                    <pair name="label" type="string">Planes de Llamada
                    </pair>
                    <pair name="lang_id" type="integer">2</pair>
                </pair>
            </pair>
            <pair name="desc" type="pair">
                <pair name="lbl_item" type="pair">
                    <pair name="label" type="string">This product folder
                    lists the different wireless plans available within
                    the organization.
                    </pair>
                    <pair name="lang_id" type="integer">1</pair>
                </pair>
            </pair>
            <pair name="vis" type="pair">
                <pair name="vis_item0" type="pair">
                    <pair name="admin" type="integer">1</pair>
                    <pair name="enduser" type="integer">1</pair>
                    <pair name="intf_id" type="integer">1</pair>
                </pair>
                <pair name="vis_item1" type="pair">
                    <pair name="admin" type="integer">0</pair>
                    <pair name="enduser" type="integer">0</pair>
                    <pair name="intf_id" type="integer">2</pair>
                </pair>
            </pair>
        </parameter>
</function>
</connector>
```

This example creates a top-level product in English and Spanish, Wireless Plans and Planes de Llamada, populates the description of the product, allows administration and end-user visibility on the interface with the ID of 1, and prohibits visibility on the interface with the ID of 2.

css_product_destroy, css_category_destroy, and css_disposition_destroy

The hierarchical menu destroy functions (css_product_destroy, css_category_destroy, and css_disposition_destroy) are used to delete an existing object (for example, a product or category) from a hierarchical menu. The parameter and pair structures of the three destroy functions are identical. The functions have two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Because the parameter and pair structures of the three destroy functions are identical, only one example is provided: css_product_destroy. The only change required for css_category_destroy and css_disposition_destroy is the function name.

Example:

```
<connector>
<function name="css_product_destroy">
     <parameter name="args" type="pair">
          <pair name="id" type="integer">112</pair>
          <pair name="seq" type="integer">1</pair>
          <pair name="parent" type="pair">
               <pair name="lvl_id1" type="integer">1</pair>
               <pair name="lvl_id2" type="integer">2</pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example deletes the product with the ID of 112 from the database.

css_product_move, css_category_move, and css_disposition_move

The hierarchical menu move functions (css_product_move, css_category_move, and css_disposition_move) are used to move objects in hierarchical menus. For example, you could move a top-level product to be a lower-level product under another top-level product. The parameter and pair structures of the three move functions are identical. The functions have two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Because the parameter and pair structures of the three move functions are identical, only one example is provided: css_product_move. The only change required for css_category_move and css_disposition_move is the function name.

Example:

```
<connector>
<function name="css_product_move">
```

```
     <parameter name="args" type="pair">
          <pair name="id" type="integer">18</pair>
          <pair name="old_seq" type="integer">3</pair>
          <pair name="new_seq" type="integer">4</pair>
          <pair name="old_lvl" type="integer">3</pair>
          <pair name="new_lvl" type="integer">3</pair>
          <pair name="old_parent" type="integer">14</pair>
          <pair name="np_lvl_id" type="pair">
               <pair name="lvl_id1" type="integer">1</pair>
               <pair name="lvl_id2" type="integer">13</pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example moves the product with the ID of 18 to the third level (new_lvl=3) of a hierarchical menu, placing it under the hierarchical menu item with the ID of 13 (lvl_id2=13), along with its associated lower-level hierarchical menu items.

css_product_update, css_category_update, and css_disposition_update

The hierarchical menu update functions (css_product_update, css_category_update, and css_disposition_update) are used to update the information associated with an existing hierarchical menu item in the RightNow database. The parameter and pair structures of the three update functions are identical. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="css_product_update">
     <parameter name="args" type="pair">
          <pair name="id" type="integer">7</pair>
          <pair name="label" type="pair">
               <pair name="lbl_item" type="pair">
                    <pair name="label" type="string">Family Calling
                    Plans</pair>
                    <pair name="lang_id" type="integer">1</pair>
               </pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example updates the name the product with the ID of 7 to be Family Calling Plans.

Incident API

The incident API functions (incident_create, incident_destroy, incident_get, and incident_update) allow you to create, delete, retrieve, or update information from the *incidents* table. You can act on all standard database fields of the *incidents* table, as well as some specialized information, such as incident custom fields and incident threads.

An incident is a question or request for help from an end-user through any of the channels into RightNow Service, such as Ask a Question, email, Live chat, site or answer feedback, or the API.

incident_create

The incident_create function is used to add an incident to the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs. In addition, you should also include a thread. If you are using organizations, you should pass the organization ID number (org_id). Depending on your configuration, you may also need to specify other fields, such as products and categories.

Note The API will automatically generate an i_id for the incident that is consistent with existing incidents in the database.

Thread entries in incidents use a unique pair structure. For more information, refer to "Adding thread entries" on page 82.

The incident will be populated with data specified in the pair list. A description of all *incidents* table fields and their corresponding pair names can be found in Appendix A, "Pair Names," on page 125. When creating an incident, the source of the incident is a required element. For a listing of sources, refer to Appendix B, "Source Codes," on page 177.

Note For records created or updated through the XML API, you should always allow the API to set the source levels. The API will automatically set source_lvl1 to 32007 and source_lvl2 to 6001. Setting the sources to other values can have a detrimental effect on your data. If you choose to set sources to other values, be sure you carefully test your work and the results.

Example:

```
<connector>
<function name="incident_create">
```

```
<parameter name="args" type="pair">
     <pair name="assigned" type="pair">
          <pair name="acct_id" type="integer">4</pair>
          <pair name="group_id" type="integer">100061</pair>
     </pair>
     <pair name="contact" type="pair">
          <pair name="ic_item1" type="pair">
               <pair name="c_id" type="integer">7</pair>
               <pair name="prmry" type="integer">1</pair>
          </pair>
          <pair name="ic_item2" type="pair">
               <pair name="c_id" type="integer">9</pair>
               <pair name="prmry" type="integer">0</pair>
          </pair>
          <pair name="ic_item2" type="pair">
               <pair name="c_id" type="integer">11</pair>
               <pair name="prmry" type="integer">0</pair>
          </pair>
     </pair>
     <pair name="cat" type="pair">
          <pair name="lvl_id1" type="integer">7</pair>
          <pair name="lvl_id2" type="integer">8</pair>
     </pair>
     <pair name="interface_id" type="integer">1</pair>
     <pair name="lang_id" type="integer">1</pair>
     <pair name="org_id" type="integer">4</pair>
     <pair name="prod" type="pair">
          <pair name="lvl_id1" type="integer">2</pair>
          <pair name="lvl_id2" type="integer">13</pair>
     </pair>
     <pair name="queue_id" type="integer">3</pair>
     <pair name="source_upd" type="pair">
          <pair name="source_lvl1" type="integer">32007</pair>
          <pair name="source_lvl2" type="integer">6001</pair>
     </pair>
     <pair name="status" type="pair">
          <pair name="id" type="integer">1</pair>
          <pair name="type" type="integer">1</pair>
     </pair>
     <pair name="thread" type="pair">
```

```
                <pair name="thread_entry" type="pair">
                    <pair name="entry_type" type="integer">3</pair>
                    <pair name="note" type="string">I want to send a
                    picture I have taken with my camera phone through
                    email. How can I do this?
                    </pair>
                    <pair name="seq" type="integer">1</pair>
                </pair>
            </pair>
            <pair name="subject" type="string">How do I send a picture
            with my new camera phone?</pair>
            <pair name="ee_flag" type="integer">1</pair>
    </parameter>
</function>
</connector>
```

In this example, an incident is created for the contact with an ID of 7. Two secondary contacts with contact IDs 9 and 11 are also associated with the incident. The incident is unresolved (status code 1). The first-level product is set to code 2 and the second-level product is set to code 13. The first-level category is set to code 7 and the second-level category is set to code 8. In addition, a contact thread is created in the incident. The function returns the i_id number for the incident. An external event will be executed if one is specified in the EE_INC_INSERT_HANDLER configuration setting.

incident_destroy

The incident_destroy function is used to delete an existing incident. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="incident_destroy">
    <parameter name="args" type="pair">
        <pair name="i_id" type="integer">21</pair>
        <pair name="ee_flag" type="integer">1</pair>
    </parameter>
</function>
</connector>
```

This example deletes the incident with the ID of 21 from the database and executes an external event.

incident_get

The incident_get function is used to retrieve the contents of the *incidents* table. The function has two components: an args parameter and an array of pair data. The sub_tbl and tbl_id pairs are used to return thread entries. The tbl_id pair should always be set to 18 (the ID of the *threads* table). Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="incident_get">
     <parameter name="args" type="pair">
          <pair name="id" type="integer">1539</pair>
          <pair name="sub_tbl" type="pair">
               <pair name="tbl_id" type="integer">18</pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example retrieves the incident details from the incident with the ID of 1539 from the database. All incident API pairs are returned using this function, including incident threads.

Note The XML API incident_get function does not return data fields with NULL values.

incident_update

The incident_update function is used to update the information associated with an existing incident in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

The API will set any fields supplied in the pair list, including custom fields and incident threads. Any incident fields missing from the pair list will not be altered in the database.

Important Incident threads cannot be updated. If you try to update an existing thread, a new thread will be added instead.

Example:

```
<connector>
<function name="incident_update">
     <parameter name="args" type="pair">
          <pair name="i_id" type="integer">7</pair>
```

```
            <pair name="status" type="pair">
                 <pair name="id" type="integer">2</pair>
                 <pair name="type" type="integer">2</pair>
            </pair>
            <pair name="assigned" type="pair">
                 <pair name="acct_id" type="integer">4</pair>
                 <pair name="group_id" type="integer">100061</pair>
            </pair>
            <pair name="custom_field" type="pair">
                 <pair name="cf_item1" type="pair">
                      <pair name="cf_id" type="integer">4</pair>
                      <pair name="val_str" type=”string”>Roaming</pair>
                 </pair>
            </pair>
       </parameter>
</function>
</connector>
```

This example updates the incident with the ID of 7 to assign the incident to the staff member with ID number 4. The status is set to solved (status=2) and a custom field (cf_id=4) is set to the value Roaming.

Meta-answer API

The meta-answer API functions (meta_ans_create, meta_ans_destroy, and meta_ans_update) allow you to create or alter information from the *meta_answers* table. You can act on all standard database fields of the *meta_answers* table.

Meta-answers are groups of sibling answers that solve the same question, but present information in different formats, either in another language or at different levels of detail. The meta-answer defines the products and categories assigned to that answer. However, you must update each answer separately to define content and visibility settings.

meta_ans_create

The meta_ans_create function is used to add a meta-answer to the RightNow database. The function has two components: the args parameter and an array of pair data. Products and categories are also required if they are enabled in your configuration.

Important The API will automatically generate a meta-answer ID for the meta-answer that is consistent with existing meta-answers in the database.

The meta-answer will be populated with data specified in the pair list. A brief description of all *meta_answers* table fields and their corresponding pair names can be found in Appendix A, "Pair Names," on page 125.

Using meta_ans_create, you can associate up to six levels of products and categories to a meta-answer by using multiple hier_item pairs within a products or categories pair array.

Example:

```
<connector>
<function name="meta_ans_create">
     <parameter name="args" type="pair">
          <pair name="notes" type="string">This is a note.</pair>
          <pair name="products" type="pair">
               <pair name="hier_item" type="pair">
                    <pair name="hm" type="pair">
                         <pair name="lvl_id1" type="integer">1</pair>
                    </pair>
               </pair>
               <pair name="hier_item" type="pair">
                    <pair name="hm" type="pair">
                         <pair name="lvl_id1" type="integer">3</pair>
                         <pair name="lvl_id2" type="integer">12</pair>
                    </pair>
               </pair>
               <pair name="hier_item" type="pair">
                    <pair name="hm" type="pair">
                         <pair name="lvl_id1" type="integer">5</pair>
                         <pair name="lvl_id2" type="integer">22</pair>
                         <pair name="lvl_id3" type="integer">33</pair>
                    </pair>
               </pair>
          </pair>
          <pair name="categories" type="pair">
               <pair name="hier_item" type="pair">
                    <pair name="hm" type="pair">
                         <pair name="lvl_id1" type="integer">13</pair>
                    </pair>
               </pair>
               <pair name="hier_item" type="pair">
                    <pair name="hm" type="pair">
                         <pair name="lvl_id1" type="integer">15</pair>
```

```
                    <pair name="lvl_id2" type="integer">42</pair>
               </pair>
          </pair>
          <pair name="hier_item" type="pair">
               <pair name="hm" type="pair">
                    <pair name="lvl_id1" type="integer">18</pair>
                    <pair name="lvl_id2" type="integer">45</pair>
                    <pair name="lvl_id3" type="integer">51</pair>
               </pair>
          </pair>
     </pair>
</parameter>
</function>
</connector>
```

In this example, a meta-answer is created in RightNow. The meta-answer is associated with three first-level products (IDs of 1, 3, and 5), two second-level products (IDs of 12 and 22), and one level-three product (ID of 33). The meta-answer is also associated with three first-level categories (IDs of 13, 15, and 18), two second-level categories (IDs of 42 and 45), and a third-level category (ID of 51). In addition, the meta-answer is associated with a linked product (ID of 4). The value of the m_id is automatically created and returned from the database.

Note When defining products and categories with multiple levels, the lvl_id pairs nested within the hm pair must match the hierarchy of your products and categories. For example, in the previous example, the category with ID 51 must be a third-level category that is associated with the second-level category with ID 45 which must be associated with the first-level category with ID 18.

meta_ans_destroy

The meta_ans_destroy function is used to delete a meta-answer. The function has two components: The args parameter and the the m_id pair. A valid m_id of an existing meta-answer must be supplied.

Caution Deleting a meta-answer will delete all answers associated with it.

Example:

```
<connector>
<function name="meta_ans_destroy">
     <parameter name="args" type="pair">
```

```
          <pair name="m_id" type="integer">25</pair>
     </parameter>
</function>
</connector>
```

This example deletes the meta-answer with an ID number of 25 from the database.

meta_ans_update

The meta_ans_update function is used to update the information associated with an existing meta-answer in the RightNow database. The function has two components: the args parameter and an array of pair data. A valid m_id of an existing meta-answer must be supplied.

The API will set any fields supplied in the pair list, and any meta-answer fields missing from the pair list will not be altered in the database.

Example:

```
<connector>
<function name="meta_ans_update">
     <parameter name="args" type="pair">
          <pair name="m_id" type="integer">25</pair>
          <pair name="categories" type="pair">
               <pair name="hier_item" type="pair">
                    <pair name="hm" type="pair">
                         <pair name="lvl_id1" type="integer">18</pair>
                    </pair>
               </pair>
               <pair name="hier_item" type="pair">
                    <pair name="hm" type="pair">
                         <pair name="lvl_id1" type="integer">13</pair>
                         <pair name="lvl_id2" type="integer">42</pair>
                    </pair>
               </pair>
               <pair name="hier_item" type="pair">
                    <pair name="hm" type="pair">
                         <pair name="lvl_id1" type="integer">15</pair>
                         <pair name="lvl_id2" type="integer">45</pair>
                         <pair name="lvl_id3" type="integer">51</pair>
                    </pair>
               </pair>
          </pair>
     </parameter>
</function>
```

```
</connector>
```

This example updates the meta-answer with a meta-answer ID of 25 to specify three additional category associations.

Opportunity API

The opportunity API functions (opp_create, opp_destroy, opp_get, and opp_update) allow you to create, delete, retrieve, or update information in the *opportunities* and *quotes* tables. You can act on all standard database fields of these tables, as well as some specialized information, such as custom fields.

Opportunities are records in RightNow Sales that contain information on a specific sale or a pending deal that is tracked and maintained in the knowledge base.

Note Through the opportunity API, you can retrieve, update, and delete quotes; however, you cannot create quotes. Quotes must be created through the RightNow Console.

opp_create

The opp_create function is used to add an opportunity to the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Note The API will automatically generate an op_id for the opportunity that is consistent with existing opportunities in the database.

Note entries in opportunities use a unique pair structure. For information, refer to "Adding Notes" on page 85.

The opportunity will be populated with data specified in the pair list. A brief description of all *opportunities* table fields and their corresponding pair names can be found in Appendix A, "Pair Names," on page 125. A description of all opportunity source codes can be found in Appendix B, "Source Codes," on page 177.

Note For records created or updated through the XML API, you should always allow the API to set the source levels. The API will automatically set source_lvl1 to 32007 and source_lvl2 to 6001. Setting the sources to other values can have a detrimental effect on your data. If you choose to set sources to other values, be sure carefully test your work and the results.

To create a hierarchy of territories or sales representatives, use the terr_lvl<1-12>_id and acct_lvl<1-11>_id pairs. The top-level territory is specified using terr_lvl1_id, the second-level territory is specified using terr_lvl2_id, and so on. The territory hierarchy in the opportunity must match the territory hierarchy.

To set the sales representative, you must use the assgn_acct_id pair, along with the acct_lvl<1-11>_id pair. The top-level sales representative, or manager, is specified using acct_lvl1_id, the second-level sales representative is specified using acct_lvl2_id, and so on. The account hierarchy in the opportunity must match the account hierarchy.

You can assign multiple secondary contacts to an opportunity using the contact pair along with an array of pair data. Separate contacts are specified using the oc_item<#> pair. The first contact is specified by oc_item0, the second contact is specified by oc_item1, and so on. You must set a contact role for each contact and indicate whether it is the primary contact.

Example:

```
<connector>
<function name="opp_create">
     <parameter name="args" type="pair">
          <pair name="updated_by" type="integer">2</pair>
          <pair name="status" type="pair">
               <pair name="id" type="integer">9</pair>
               <pair name="type" type="integer">6</pair>
          </pair>
          <pair name="contact" type="pair">
               <pair name="oc_item0" type="pair">
                    <pair name="c_id" type="integer">1</pair>
                    <pair name="cr_id" type="integer">1</pair>
                    <pair name="prmry" type="integer">1</pair>
               </pair>
               <pair name="oc_item1" type="pair">
                    <pair name="c_id" type="integer">3</pair>
                    <pair name="cr_id" type="integer">2</pair>
                    <pair name="prmry" type="integer">0</pair>
               </pair>
          </pair>
          <pair name="org_id" type="integer">1</pair>
          <pair name="name" type="string">Fall Clearance Sale</pair>
          <pair name="summary" type="string">40% off on all camera phones
          for existing customers.</pair>
     </parameter>
</function>
```

```
</connector>
```

This example creates an opportunity, Fall Clearance Sale, and associates two contacts (one primary and one secondary) with the opportunity. This function will automatically return the opportunity ID from the database.

opp_destroy

The opp_destroy function is used to delete an existing opportunity in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Tip If you are destroying an opportunity, it is not necessary to destroy the associated quotes; all associated quotes will be automatically destroyed.

Example:

```
<connector>
<function name="opp_destroy">
     <parameter name="args" type="pair">
          <pair name="op_id" type="integer">116</pair>
     </parameter>
</function>
</connector>
```

This example deletes the opportunity with the ID number 116 from the database.

opp_get

The opp_get function is used to retrieve a record from the *opportunities* table and associated records from the *quotes* table. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

If you want to retrieve information about associated quotes, use the same pair structure that you would use to retrieve note information.

Example:

```
<connector>
<function name="opp_get">
     <parameter name="args" type="pair">
          <pair name="id" type="integer">9</pair>
          <pair name="sub_tbl" type="pair">
               <pair name="tbl_id" type="integer">163</pair>
               <pair name="tbl_id" type="integer">12</pair>
          </pair>
```

```
     </parameter>
</function>
</connector>
```

This example retrieves the opportunity information for the opportunity with the ID number of 9 from the database. It also retrieves information for notes (tbl_id=163) and associated quotes (tbl_id=12).

Note The XML API opp_get function does not return data fields with NULL values.

opp_update

The opp_update function is used to update the information associated with an existing opportunity in the RightNow database, including quote information. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs. If you are updating a quote, the action pair must be set to 2.

Note Note entries in opportunities use a unique pair structure. For information, refer to "Adding Notes" on page 85.

The API will set any fields supplied in the pair list, including custom fields. Any opportunity fields not present in the pair list will not be altered in the database.

Note If another contact is added or one of the contacts is deleted from the contact list, the entire new list needs to be passed because the API deletes the existing contact list and inserts the new list if there is any change. If there is no change, opp2contact does not need to be set on update.

Example:

```
<connector>
<function name="opp_update">
     <parameter name="args" type="pair">
          <pair name="op_id" type="integer">568</pair>
          <pair name="status" type="pair">
               <pair name="id" type="integer">11</pair>
               <pair name="type" type="integer">8</pair>
          </pair>
          <pair name="closed_value" type="pair">
               <pair name="curr_id" type="integer">1</pair>
               <pair name="rate_id" type="integer">1</pair>
```

```
                <pair name="val" type="string">99000</pair>
            </pair>
            <pair name="qt" type="pair">
                <pair name="qt_item" type="pair">
                    <pair name="action" type="integer">2</pair>
                    <pair name="quote_id" type="integer">767</pair>
                    <pair name="forecast" type="integer">1</pair>
                </pair>
            </pair>
        </parameter>
</function>
</connector>
```

This example updates the opportunity with ID number 568, changing the opportunity's status to an ID of 11 (Closed) and setting the closed value to $99,000. Additionally, the Rep Forecast field will be updated with the value of the associated quote (forecast=1).

Organization API

The organization API functions (org_create, org_destroy, org_get, and org_update) allow you to create, delete, retrieve, or update information from the *orgs* table. You can act on all standard database fields of the *orgs* table, as well as some specialized information, such as custom fields.

Contacts can be associated with organizations in RightNow. By associating contacts with organizations, contacts and staff members can view all incidents submitted by an organization and allow administrators to assign an SLA instance to all contacts in an organization.

By including SLA pairs, you can also issue and terminate SLAs when creating and updating organizations. For information on assigning and terminating SLAs, refer to "Creating and deleting SLA instances" on page 85.

Multiple addresses

An organization can have several types of addresses, including a billing and shipping address. When passing address information using the org_create or org_update function, a unique pair structure is used. Table 10 describes the default address types that can be associated with each organization.

Table 10: Address Type Descriptions

Address Type (oat_id)	ID
Billing	1
Shipping	2

The following example shows the pair for each the billing and the shipping address.

Example:

```
<connector>
<function name="org_update">
<parameter name="args" type="pair">
     <pair name="org_id" type="integer">27</pair>
     <pair name="oaddr" type="pair">
          <pair name="oaddr_item1" type="pair">
               <pair name="oat_id" type="integer">1</pair>
               <pair name="addr" type="pair">
                    <pair name="street" type="string">12345 Maple Way
                    </pair>
                    <pair name="city" type="string">Bozeman</pair>
                    <pair name="prov_id" type="integer">32</pair>
                    <pair name="postal_code" type="string">59718</pair>
                    <pair name="country_id" type="integer">1</pair>
               </pair>
          </pair>
          <pair name="oaddr_item2" type="pair">
               <pair name="oat_id" type="integer">2</pair>
               <pair name="addr" type="pair">
                    <pair name="street" type="string">4321 Oak Street
                    </pair>
                    <pair name="city" type="string">Belgrade</pair>
                    <pair name="prov_id" type="integer">32</pair>
                    <pair name="postal_code" type="string">59714</pair>
```

```
                    <pair name="country_id" type="integer">1</pair>
                </pair>
            </pair>
        </pair>
</parameter>
</function>
</connector>
```

org_create

The org_create function is used to add an organization to the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Important The API will automatically generate an org_id for the organization that is consistent with existing organizations in the database.

The organization will be populated with data specified in the pair list. A brief description of all *orgs* table fields and their corresponding pair names can be found in Appendix A, "Pair Names," on page 125. A brief description of all organization source codes can be found in Appendix B, "Source Codes," on page 177.

Note Note entries in organizations use a unique pair structure. For information, refer to "Adding Notes" on page 85.

Example:

```
<connector>
<function name="org_create">
     <parameter name="args" type="pair">
               <pair name="name" type="string">The River Deep</pair>
               <pair name="login" type="string">riverdeep</pair>
               <pair name="password" type="string">wh1tewat3r</pair>
               <pair name="state" type="pair">
                    <pair name="css" type="integer">1</pair>
                    <pair name="ma" type="integer">0</pair>
                    <pair name="sa" type="integer">0</pair>
               </pair>
               <pair name="ee_flag" type="integer">1</pair>
     </parameter>
</function>
</connector>
```

This example creates an organization, The River Deep, with a login of "riverdeep" and a password of "wh1tewat3r." The organization state is set to Service. The function will return the organization ID number and execute an external event if there is a value in the EE_ORG_INSERT_HANDLER configuration setting.

org_destroy

The org_destroy function is used to delete an existing organization in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Caution Deleting an organization will result in the deletion of all contacts, incidents, and opportunities associated with the organization.

Example:

```
<connector>
<function name="org_destroy">
     <parameter name="args" type="pair">
          <pair name="org_id" type="integer">8</pair>
          <pair name="ee_flag" type="integer">1</pair>
     </parameter>
</function>
</connector>
```

This example deletes the organization with ID number 8 and executes an external event if one is specified in the EE_ORG_DELETE_HANDLER configuration setting.

org_get

The org_get function is used to retrieve a record from the *orgs* table. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="org_get">
     <parameter name="args" type="pair">
          <pair name="id" type="integer">7</pair>
          <pair name="sub_tbl" type="pair">
               <pair name="tbl_id" type="integer">163</pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example retrieves the organization details with ID number 7 from the database.

Note The XML API org_get function does not return data fields with NULL values.

org_update

The org_update function is used to update the information associated with an existing organization in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Note Note entries in organizations use a unique pair structure. For information, refer to "Adding Notes" on page 85.

The API will set any fields supplied in the pair list, including custom fields. Any organization fields missing from the pair list will not be altered in the database.

Example:

```
<connector>
<function name="org_update">
     <parameter name="args" type="pair">
          <pair name="org_id" type="integer">9</pair>
          <pair name="password" type="string">newpassword</pair>
     </parameter>
</function>
</connector>
```

This example changes the password of the organization with ID number 9 to "newpassword."

Purchased product API

The purchased product API function, pur_prod_create, allows you to insert information in to the *purchased_products* table. The information in the *purchased_products* table is used by RightNow Marketing and Offer Advisor. You can act on all standard database fields of the *purchased_products* table, as well as some specialized information, such as custom fields.

pur_product_create

The pur_prod_create function is used to add a purchased product to the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="pur_prod_create">
     <parameter name="args">
          <pair name="pp_item1" type="pair">
               <pair name="c_id" type="integer">8</pair>
               <pair name="campaign_id" type="integer">4</pair>
               <pair name="finalized_by" type="integer">11</pair>
               <pair name="license_end" type="time">1288544400</pair>
               <pair name="license_start" type="time">1162328858</pair>
               <pair name="mailing_id" type="integer">8</pair>
               <pair name="notes" type="string">Replaces 1YR400U</pair>
               <pair name="oa_c_id" type="integer">311</pair>
               <pair name="op_id" type="integer">25</pair>
               <pair name="org_id" type="integer">4</pair>
               <pair name="price" type="pair">
                    <pair name="curr_id" type="integer">1</pair>
                    <pair name="rate_id" type="integer">1</pair>
                    <pair name="val" type="string">1288</pair>
               </pair>
               <pair name="product_id" type="integer">23</pair>
               <pair name="purchase_date" type="time">1162328858</pair>
               <pair name="quote_id" type="integer">1</pair>
               <pair name="serial_number" type="string">SN420ASD</pair>
          </pair>
     </parameter>
</function>
</connector>
```

This example adds a complete record to the *purchased_products* table.

Sales product API

The sales product API functions (sa_prod_create, sa_prod_destroy, and sa_prod_update) allow you to create, delete, or update information in the *sa_products* table. The information in the *sa_products* table is used by Offer Advisor. You can act on all standard database fields of the *sa_products* table, as well as some specialized information, such as custom fields.

Sales products are used by RightNow Sales to identify items or services sold by an organization. Sales products can be added to quotes and promotions.

sa_prod_create

The sa_prod_create function is used to add a sales product to the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Important The API will automatically generate a product_id that is consistent with existing sales products in the database.

Example:

```
<connector>
<function name="sa_prod_create">
     <parameter name="args">
          <pair name="desc" type="pair">
               <pair name="lbl_item" type="pair">
                    <pair name="label" type="string">One year contract
                    with 400 daytime minutes and unlimited evening and
                    weekend minutes</pair>
                    <pair name="fld" type="integer">2</pair>
                    <pair name="lang_id" type="integer">1</pair>
                    <pair name="tbl" type="integer">93</pair>
               </pair>
          </pair>
          <pair name="label" type="pair">
               <pair name="lbl_item" type="pair">
                    <pair name="label" type="string">1 Yr - 400 Airtime
                    Unlimited Nights and Weekends</pair>
                    <pair name="fld" type="integer">1</pair>
                    <pair name="lang_id" type="integer">1</pair>
                    <pair name="tbl" type="integer">93</pair>
               </pair>
          </pair>
          <pair name="disabled" type="integer">0</pair>
          <pair name="id" type="string">1YR400U</pair>
          <pair name="oa_exclude" type="integer">0</pair>
          <pair name="seq" type="integer">37</pair>
     </parameter>
</function>
</connector>
```

This example creates a new sales product in the product catalog with a name, description, and ID.

sa_prod_destroy

The sa_prod_destroy function is used to delete a sales product from the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="sa_prod_destroy">
     <parameter name="args">
          <pair name="product_id" type="integer">5</pair>
     </parameter>
</function>
</connector>
```

This example deletes the sales product with the product_id of 5 from the RightNow database.

sa_prod_update

The sa_prod_update function is used to update the information associated with an existing sales product in the RightNow database. The function has two components: an args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Example:

```
<connector>
<function name="sa_prod_update">
     <parameter name="args">
          <pair name="product_id" type="integer">5</pair>
          <pair name="disabled" type="integer">1</pair>
          <pair name="oa_exclude" type="integer">1</pair>
          <pair name="folder_id" type="integer">100343</pair>
     </parameter>
</function>
</connector>
```

This example updates the sales product with the product_id of 5, disabling the sales product, removing it from use by Offer Advisor, and moving it to a new folder.

Search API

The XML search API function can be used to search for records in RightNow, including incidents, answers, contacts, organizations, opportunities, quotes, and tasks.

The function has two components: the args and ac_id parameters, but filters associated with the report can also be applied. All fixed filters defined in the report specified by the ac_id are applied to the query that is run by the XML search API function and any run-time filters with default values are also included. To narrow your search, you can use the run-time selectable filters created in the view.

Note The ac_id must be defined on the current interface to work correctly. To find the ac_id for a desired report, refer to "Finding code numbers" on page 89.

The run-time selectable filters are applied by passing the search_args pair. To identify the filter, set the name pair to the name of the filter (the name is specified by you when you create the filter in the report). The value is passed using the compare_val pair. The format of the data in the compare_val pair depends on the type of operator specified in the filter. Table 11 describes the type of data to use in the compare_val pair for each type of operator.

Important If you use percent-encoding reserved characters in a search string, the characters must be percent encoded (also called URL encoded). For example, if you use the percent symbol (%) as a wildcard in a search string, the percent sign must be percent-encoded.

Table 11: Operators Description

If the operator is...	Then data in the compare_val pair is...
=, !=, <, <=, >, >=, like, not like, is null, != or null, not like or null	A number or string. For example, to search for the word "roaming" within a string (such as an incident subject), you would use the following: `<pair name="compare_val" type="string">%25roaming%25</pair>` Where %25 is the percent-encoded representation of the % wildcard.

Table 11: Operators Description (Continued)

If the operator is...	Then data in the compare_val pair is...

in list, not in list	A list of numbers separated by semicolons.
	For example, to search for two statuses (IDs are 4 and 5), you would use the following:
	`<pair name="compare_val" type="string">4;5 </pair>`
	
	When searching for products and categories, you must specify the level the code ID is associated with. The format is <level>.<ID>. For example, to search for a product (ID is 2) and two of its lower-level products (IDs are 9 and 12), you would use the following:
	`<pair name="compare_val" type="string"> 1.2;2.9;2.12</pair>`
	
	To search for something that has product=5 and sub-level NULL, you specify "2.u5", which says that the level two ID should be NULL and the level 1 ID should be 5. You can combine this with others as follows:
	`<pair name="compare_val" type="string"> 1.2;2.u5</pair>`
	In the example listed above, "1.2;2.u5" would equate to "prod_lvl1=2 OR (prod_lvl1=5 AND prod_lvl2 IS NULL)."
	
	If you want to specify that the product should be NULL, you use "1.u0", which is a special case, since the level 1 values have no parents.
	
	Note: Six levels of products and categories are supported. For example, "1.2;3.22;4.u35" would search everything with "prod_lvl1=2 or prod_lvl3=22 or (prod_lvl3=35 and prod_lvl4 is null)."
	
	In the previous example, the "4.u35" describes that prod_lvl4 should equal something. In this particular case, the "u35" describes that prod_lvl4 should be null, but the parent should be 35 (which means prod_lvl3=35).
	
	In other words, if the string was "1.9;4.u23", it would expand to prod_lvl1=9 OR (prod_lvl4 is NULL and prod_lvl3=23).

Table 11: Operators Description (Continued)

If the operator is...	Then data in the compare_val pair is...
between	A set of two numbers, separated by a pipe (|). For example, to search for answers with an ID between 1 and 50, you would use the following: `<pair name="compare_val" type="string">1|50 </pair>`

You can use the max_rows parameter to pass the maximum number of rows returned by the search. The upper limit of the allowed number passed in this parameter is set by the configuration setting VRL_HARD.

Note If your view output data length is set to more than 4000 characters, the XML search API will truncate the return results at the 4000 character limit.

Examples:

The following example produces a default result set defined by the referenced answer view ID.

```
<connector>
<function name="search">
     <parameter name="ac_id" type="integer">100026</parameter>
</function>
</connector>
```

The following example shows a search by product and a range of answer IDs. In this example, a filter named Product must exist in the report with the ac_id of 100026.

```
<connector>
<function name="search">
     <parameter name="args" type="pair">
          <pair name="search_args" type="pair">
               <pair name="search_field1" type="pair">
                    <pair name="name" type="string">Product</pair>
                    <pair name="compare_val" type="string">1.2;2.9;2.12;
                    </pair>
               </pair>
               <pair name="search_field2" type="pair">
                    <pair name="name" type="string">a_id</pair>
                    <pair name="compare_val" type="string">1|50
                    </pair>
```

```
                </pair>
            </pair>
        </parameter>
        <parameter name="ac_id" type="integer">100026</parameter>
        <parameter name="max_rows" type="integer">5</parameter>
</function>
</connector>
```

In this example, the function searches for all answers with a product ID of 2, a second-level product ID of 9 or 12, and an ID between the range of 1 and 50 and returns the values according to the report with the ID (ac_id) of 1000026.

Example result set:

The following is an example of a set of results from an answer search.

```
<connector_ret>
<function name="search" id="">
     <row id="1">
          <col id="1">1</col>
          <col id="2">How do I email a photo with my camera phone?</col>
          <col id="3">en_US</col>
          <col id="4">Everyone</col>
          <col id="5">Public</col>
          <col id="6">Mary Smith</col>
          <col id="7">1036594069</col>
          <col id="8">100</col>
          <col id="9">1</col>
          <col id="10">1</col>
          <col id="11" />
          <col id="12">1128020129</col>
          <col id="13">1</col>
          <col id="14">0</col>
     </row>
     <row id="2">
          <col id="1">2</col>
          <col id="2">What will it cost for me to upgrade to your business
          plan?</col>
          <col id="3">en_US</col>
          <col id="4">Everyone</col>
          <col id="5">Public</col>
          <col id="6">Mary Smith</col>
          <col id="7">1036594069</col>
```

```
            <col id="8">100</col>
            <col id="9">2</col>
            <col id="10">1</col>
            <col id="11" />
            <col id="12">1128022171</col>
            <col id="13">1</col>
            <col id="14">0</col>
      </row>
</function>
</connector_ret>
```

The above result set shows a search return value containing two rows, or two matching records for a search. Each row relates directly to a row in the specified report.

SQL query API

The SQL query API functions (sql_get_int, sql_get_str, and sql_get_dttm) allow read-only access to the RightNow database through the XML API. These functions will return a single value from the database. The ID attribute and the sql parameter must be supplied for these functions.

When using any of the sql_get functions, if more than one value meets the criteria of the SQL statement, only the first value to match the criteria will be returned. For this reason, you should not use an SQL statement like `SELECT * from <table>` because only the first value in the table will be returned; however, SQL statements like `SELECT COUNT(*) from <table>` or `SELECT MAX(acct_id) FROM accounts` would work well because they only return a single value.

Important If you use percent-encoding reserved characters in a search string, the characters must be percent encoded (also called URL encoded). For example, if you use the percent symbol (%) as a wildcard in a search string. the percent sign must be percent-encoded.

The terminating semicolon is implied for all SQL statements.

sql_get_int

The sql_get_int function is used to execute a SELECT statement against the RightNow database when the result is an integer, such as an account ID from the *accounts* table or a count of records in the *incidents* table. The function has one component: the sql parameter. A single integer will be returned.

Example:

```
<connector>
<function name="sql_get_int" id="sql_str">
     <parameter name="sql" type="string">
          SELECT acct_id FROM accounts WHERE login = 'susan'
     </parameter>
</function>
</connector>
```

Return:

```
<?xml version="1.0" encoding="UTF-8" ?>
<connector_ret>
     <function name="sql_get_int" id="sql_str">
          <ret_val name="rv" type="integer">8</ret_val>
     </function>
</connector_ret>
```

This example runs an SQL query to find the account ID for the account with a login of "susan" and returns the integer "8."

sql_get_str

The sql_get_str function is used to execute a SELECT statement against the RightNow database when the result is a string, such as the login name from the *accounts* table. The function has one component: the sql parameter. A single string will be returned.

Example:

```
<connector>
<function name="sql_get_str" id="sql_str">
     <parameter name="sql" type="string">
          SELECT login FROM accounts WHERE acct_id = 10
     </parameter>
</function>
</connector>
```

Return:

```
<?xml version="1.0" encoding="UTF-8" ?>
<connector_ret>
     <function name="sql_get_str" id="sql_str">
          <ret_val name="rv" type="string">archie</ret_val>
     </function>
</connector_ret>
```

This example runs an SQL query to find the account login for the account with the ID of 10 and returns the string "archie."

sql_get_dttm

The sql_get_dttm function is used to execute a SELECT statement against the RightNow database when the result is a datetime, such as the password expiration time from the *accounts* table. The function has one component: the sql parameter. A single datetime will be returned in UNIX date_t format (the number of seconds since the UNIX Epoch date).

Example:

```
<connector>
<function name="sql_get_dttm">
     <parameter name="sql" type="string">
          SELECT password_exp FROM accounts WHERE acct_id = 10
     </parameter>
</function>
</connector>
```

Return:

```
<?xml version="1.0" encoding="UTF-8" ?>
<connector_ret>
     <function name="sql_get_dttm">
          <ret_val name="rv" type="time">1174314061</ret_val>
     </function>
</connector_ret>
```

This example runs an SQL query to find the password expiration time for the account with the ID of 10 and returns the value "1174314061."

Task API

The task API functions (task_create, task_destroy, task_get, and task_update) allow you to create, update, delete, or retrieve a task from the *tasks* table.

Tasks are actions or activities scheduled to be completed within a specified time. Tasks can be standalone, or they can be associated with answers, campaigns, contacts, documents, incidents, mailings, opportunities, organizations, surveys, and stages in a sales strategy.

task_create

The task_create function is used to add a task to the RightNow database. The function has two components: the args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

Important The API will automatically generate a task_id for the task. If no name is specified for the task, it will be named New Task.

The task will be populated with data specified in the pair list. A brief description of all *task* table fields and their corresponding pair names can be found in Appendix A, "Pair Names," on page 125.

Example:

```
<connector>
     <function name="task_create">
          <parameter name="args" type="pair">
               <pair name="assgn_acct_id" type="integer">2</pair>
               <pair name="tbl" type="integer">87</pair>
               <pair name="status" type="pair">
                    <pair name="id" type="integer">1</pair>
                    <pair name="type" type="integer">1</pair>
               </pair>
               <pair name="name" type="string">Schedule Follow-Up Call
               </pair>
               <pair name="due_date" type="time">1176045800</pair>
          </parameter>
     </function>
</connector>
```

This example creates a task with the name "Schedule Follow-Up Call." The API will automatically generate a task_id.

task_destroy

The task_destroy function is used to delete an existing task in the RightNow database. The function has two components: the args parameter, and the task_id pair. A valid task_id of an existing task must be supplied; otherwise, the function will abort with an error message.

Example:

```
<connector>
<function name="task_destroy">
     <parameter name="args" type="pair">
          <pair name="task_id" type="integer">7</pair>
     </parameter>
</function>
</connector>
```

This example deletes the task with the ID number 7 from the database.

task_get

The task_get function is used to retrieve a record from the *tasks* table. This function has two components: the args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs. A valid task ID number must be supplied in the id pair; otherwise, a blank value will be returned. If the function executes without error, the task details will be returned.

Example:

```
<connector>
<function name="task_get">
     <parameter name="args" type="pair">
          <pair name="id" type="integer">7</pair>
     </parameter>
</function>
</connector>
```

This example retrieves the task details with ID number 7 from the database.

Note The XML API task_get function does not return data fields with NULL values.

task_update

The task_update function is used to update the information associated with an existing task in the RightNow database. The function has two components: the args parameter and an array of pair data. Refer to Table 4 on page 22 for a list of required pairs.

The API will set any fields supplied in the pair list, including custom fields. Any task fields omitted from the pair list will not be altered in the database.

Example:

```
<connector>
<function name="task_update">
     <parameter name="args" type="pair">
          <pair name="task_id" type="integer">1109</pair>
          <pair name="name" type="string">Schedule Wrap-Up Call</pair>
     </parameter>
</function>
</connector>
```

This example updates the task name to "Schedule Wrap-Up Call."

Additional actions

When using many of the XML API functions, you can perform other actions using nested pairs, including:

- Setting custom fields
- Creating thread and note entries
- Creating and terminating SLA instances
- Passing variables between functions
- Finding code numbers

The following sections contain information on using these pairs, along with information on finding code values for fields.

Setting custom fields

You can set custom field values for custom fields when creating or updating answers, contacts, incidents, opportunities, quotes, sales products, or tasks with the XML API. Passing custom field data through the API is different than interacting with standard database fields. To set a custom field using a create or update function, each custom field must be specified within a custom_field pair and a nested cf_item pair containing an array of pairs, including the cf_id, data_type, and val_<type> pairs. The val_<type> pairs are also called "value pairs."

Using cf_id pairs

The cf_id pair specifies the code of the custom field. In this pair, the name should be set to cf_id with a type of integer, and the value of the pair will be the code number of the custom field. For information on finding the code value of a custom field, refer to "Finding code numbers" on page 89.

Using data_type pairs

The data_type pair specifies the code of the custom field data type. The possible data types and the corresponding codes are specified in Table 12.

Table 12: Custom Field Data Types and Codes

Code	Data Type
1	Menu
2	Radio

Table 12: Custom Field Data Types and Codes (Continued)

Code	Data Type
3	Integer
4	Date/Time
5	Text field
6	Text area
7	Date

Using value pairs

The value pair specifies the value you want the custom field set to. There are three value pairs: val_int, val_str, and val_time. Each value pair is used for different data types, as specified by the data_type pair. Each pair must also has a corresponding type which must be correctly specified in the XML tag. These three pairs are described in Table 13.

Table 13: Value Pairs Description

Pair name	Description	Type
val_int	Use this pair for menu, radio, and integer custom fields (data types 1, 2, and 3).	integer
val_str	Use this pair for text and text area custom fields (data types 5 and 6).	string
val_time	Use this pair for date/time and date custom fields (data types 4 and 7).	time

Important Date and date/time custom fields must be configured with a type of time, and the value must be in UNIX date_t format; that is, a long integer that is the number of seconds since the UNIX Epoch date (00:00:00 UTC January 1, 1970).

Examples:

The following example shows how to set two custom fields, a text field and a text area, where the custom field with cf_id code of 37 is set to "Main St. Branch," and the custom field with the cf_id code of 38 is set to "Model 80801-DS9."

```
<pair name="custom_field" type="pair">
     <pair name="cf_item1" type="pair">
          <pair name="cf_id" type="integer">37</pair>
          <pair name="data_type" type="integer">5</pair>
          <pair name="val_str" type="string">Main St. Branch</pair>
     </pair>
     <pair name="cf_item2" type="pair">
          <pair name="cf_id" type="integer">38</pair>
          <pair name="data_type" type="integer">6</pair>
          <pair name="val_str" type="string">Model 80801-DS9</pair>
     </pair>
</pair>
```

The following example shows how to set a radio button custom field and a date/time custom field, where the value pair is set to 1 for "yes," or 0 for "no."

```
<pair name="custom_field" type="pair">
     <pair name="cf_item5" type="pair">
          <pair name="cf_id" type="integer">41</pair>
          <pair name="data_type" type="integer">2</pair>
          <pair name="val_int" type="integer">0</pair>
     </pair>
     <pair name="cf_item6" type="pair">
          <pair name="cf_id" type="integer">42</pair>
          <pair name="data_type" type="integer">4</pair>
          <pair name="val_time" type="time">1096045800</pair>
     </pair>
</pair>
```

Adding thread entries

An incident can contain a threaded conversation between staff members and customers. A sales contact, sales opportunity, or sales organization can contain threaded entries by staff members only. You can create threads with the XML API create and update functions by using nested pairs which allow you to specify the type of thread that is associated with the incident. The following pairs are required for create and update functions: entry_type, note, and seq.

Table 14 describes the thread types that can be associated with threads and notes.

Table 14: Thread Entry Types Description

Thread Entry Type	ID
Note	1
Staff	2
Contact	3
Contact Proxy	4
RightNow Live	5
Rule Response	6
Rule Response Note	7
Sales Note	8
Sales Customer Email	9
Sales Email	10
Sales Phone	11

Table 15 describes the table types that can be associated with threads and notes.

Table 15: Channel Types Description

Channel Type	ID
Service Mailbox	1
Marketing Mailbox	2
Phone	3
Fax	4

Table 15: Channel Types Description (Continued)

Channel Type	ID
Postal	5
Service End-User Pages	6
RightNow Live	8

The following example shows two threads, the first from a customer, the second from a staff member. This example is typical of thread entries used in the incident_create function.

Example:

```
<pair name="thread" type="pair">
     <pair name="thread_entry1" type="pair">
          <pair name="c_id" type="integer">1423</pair>
          <pair name="channel" type="integer">3</pair>
          <pair name="entered" type="time">1096045800</pair>
          <pair name="entry_type" type="integer">3</pair>
          <pair name="note" type="string">How do I access voice mail?
          </pair>
          <pair name="seq" type="integer">1</pair>
     </pair>
     <pair name="thread_entry2" type="pair">
          <pair name="acct_id" type="integer">11</pair>
          <pair name="c_id" type="integer">1423</pair>
          <pair name="channel" type="integer">3</pair>
          <pair name="entered" type="time">1096060222</pair>
          <pair name="entry_type" type="integer">2</pair>
          <pair name="note" type="string">Thank you for your inquiry. One
          of our agents will respond to you as soon as possible.</pair>
          <pair name="seq" type="integer">2</pair>
     </pair>
</pair>
```

Important Incident threads cannot be updated. If you try to update an existing thread, a new thread will be added instead.

Adding Notes

Contacts, organizations, and opportunities can have notes associated with them. You can create notes with the XML API create and update functions by using nested pairs which allow you to specify the type of note that is associated with the record. For a description of channel types that can be used for notes, refer to Table 15 on page 83.

Note The action pair should always have a value of 1.

The following contact_update example shows a note entry.

Example:

```
<connector>
<function name="contact_update">
     <parameter name="args" type="pair">
          <pair name="c_id" type="integer">7</pair>
          <pair name="note" type="pair">
               <pair name="note_item1" type="pair">
                    <pair name="action" type="integer">1</pair>
                    <pair name="seq" type="integer">1</pair>
                    <pair name="text" type="string">Updated through
                    the XML API</pair>
               </pair>
          </pair>
     </parameter>
</function>
</connector>
```

Creating and deleting SLA instances

A service level agreement (SLA) instance is an assigned SLA, assigned by an agent from the RightNow Console or through the XML API and associated with a contact or organization. For additional information on SLAs, see the *RightNow Service Administrator Manual*.

By using the slai pair and associated nested pairs in the contact_create, contact_update, org_create, and org_update functions, you can create or delete an SLA instance within the *sla_instances* table. The slai pair must contain an array of pair data, including the slai_item<#> pair and the action, owner_tbl, owner_id, sla_id, and activedate nested pairs.

Important The action pair is required to create or terminate an SLA instance. The value of the action pair determines whether the SLA instance is created or terminated; set this value to 1 to create an instance (used in contact_create or org_create functions) or 3 to delete an instance (used in contact_update or org_update functions). The owner_id corresponds with the c_id of the contact or org_id of the organization the SLA is associated with. The owner_tbl corresponds with the table ID of the table the owner_id is associated with; contact-associated SLAs will have a table ID of 2 and organization-associated SLAs will have a table ID of 3.

When updating an SLA instance, the slai_id pair must be included; however, when creating an SLA instance, the API will automatically generate an slai_id for the SLA instance that is consistent with existing SLA instances in the database, so you should not include the slai_id pair when creating an SLA instance.

The SLA instance will be populated with data specified in the pair list. A brief description of all *sla_instances* table fields and their corresponding pair names can be found in Appendix A, "Pair Names," on page 125.

The following example shows an array of pair data that would create an SLA instance when used in a contact_create or org_create function.

Example:

```
<pair name="slai" type="pair">
     <pair name="slai_item1" type="pair">
          <pair name="action" type="integer">1</pair>
          <pair name="activedate" type="time">1097683200</pair>
          <pair name="expiredate" type="time">1129219200</pair>
          <pair name="inc_chat" type="integer">10</pair>
          <pair name="inc_csr" type="integer">10</pair>
          <pair name="inc_email" type="integer">10</pair>
          <pair name="inc_total" type="integer">40</pair>
          <pair name="inc_web" type="integer">10</pair>
          <pair name="sla_id" type="integer">1</pair>
          <pair name="sla_set" type="integer">1</pair>
          <pair name="state" type="integer">2</pair>
```

```
     </pair>
</pair>
```

This example creates an SLA instance in the *sla_instances* table, sets the owner_id to 4, the owner_tbl to 3 (the *orgs* table).

The following example shows an array of pair data that would terminate an SLA instance when used in a contact or organization function.

```
<pair name="slai" type="pair">
     <pair name="slai_item1" type="pair">
          <pair name="action" type="integer">3</pair>
          <pair name="slai_id" type="integer">61</pair>
     </pair>
</pair>
```

This example terminates the SLA instance with the slai_id of 61.

Passing variable IDs

When you use multiple XML functions in the same XML file, the XML API allows you to store newly created record IDs in a variable to be used later in your XML. To create a variable, define the variable using the id attribute in the function tag as shown in the following example.

```
<function name="org_create" id="organization_id">
```

In this example, the org_id assigned to the new organization will be stored in the variable organization_id. This variable can be called later in your XML by replacing the org_id with the variable $organization_id.

The following example shows how you can create a contact and also create an incident associated with that contact in the same XML file by creating and passing the variable contact_id.

Example:

```
<connector>
<function name="contact_create" id="contact_id">
     <parameter name="args" type="pair">
          <pair name="name" type="pair">
               <pair name="first" type="string">Joe</pair>
               <pair name="last" type="string">Smith</pair>
          </pair>
          <pair name="email" type="pair">
               <pair name="addr" type="string">js@example.com</pair>
               <pair name="cert" type="string"></pair>
```

```
            </pair>
            <pair name="state" type="pair">
                 <pair name="css" type="integer">1</pair>
                 <pair name="ma" type="integer">1</pair>
                 <pair name="sa" type="integer">0</pair>
            </pair>
            <pair name="ee_flag" type="integer">1</pair>
       </parameter>
  </function>
  <function name="incident_create">
       <parameter name="args" type="pair">
            <pair name="contact_id" type="pair">$contact_id</pair>
            <pair name="assigned" type="pair">
                 <pair name="acct_id" type="integer">4</pair>
                 <pair name="group_id" type="integer">100061</pair>
            </pair>
            <pair name="contact" type="pair">
                 <pair name="ic_item1" type="pair">
                      <pair name="c_id" type="integer">7</pair>
                      <pair name="prmry" type="integer">1</pair>
                 </pair>
                 <pair name="ic_item2" type="pair">
                      <pair name="c_id" type="integer">9</pair>
                      <pair name="prmry" type="integer">0</pair>
                 </pair>
                 <pair name="ic_item1" type='pair'>
                      <pair name="c_id" type='integer'>11</pair>
                      <pair name="prmry" type='integer'>0</pair>
                 </pair>
            </pair>
            <pair name="cat" type="pair">
                 <pair name="lvl_id1" type="integer">7</pair>
                 <pair name="lvl_id2" type="integer">8</pair>
            </pair>
            <pair name="interface_id" type="integer">1</pair>
            <pair name="lang_id" type="integer">1</pair>
            <pair name="org_id" type="integer">4</pair>
            <pair name="prod" type='pair'>
                 <pair name="lvl_id1" type="integer">2</pair>
                 <pair name="lvl_id2" type="integer">13</pair>
```

```
            </pair>
            <pair name="queue_id" type="integer">3</pair>
            <pair name="status" type="pair">
                 <pair name="id" type="integer">1</pair>
                 <pair name="type" type="integer">1</pair>
            </pair>
            <pair name="thread" type="pair">
                 <pair name="thread_entry" type="pair">
                      <pair name="entry_type" type="integer">3</pair>
                      <pair name="note" type="string">I want to send a
                      picture I have taken with my camera phone through
                      email. How can I do this?
                      </pair>
                      <pair name="seq" type="pair">1</pair>
                 </pair>
            </pair>
            <pair name="subject" type="string">How do I send a picture
            with my new camera phone?</pair>
            <pair name="ee_flag" type="integer">1</pair>
       </parameter>
</function>
</connector>
```

In this example, the contact_id variable is set when the c_id is returned by the contact_create function. When the incident is created, the c_id of the newly created contact is passed using the variable $contact_id in the c_id pair.

Finding code numbers

You will frequently need to use code numbers in your XML to identify items such as products, categories, custom fields, and staff accounts. RightNow provides two easy ways to look up the codes for these types of fields: mouseover functionality and the lookup_id_for_name function.

Using the mouseover function

You can use RightNow mouseover functionality to look up many of the code numbers you need. Simply mouse over a profile, group, staff account, contact type, country, state, province, organization address type, service product, service category, incident disposition, incident status, answer status, answer access level, billable task, SLA, or custom field.

Figure 1 shows the mouseover function for staff accounts. In this example, Tim Johnson's account ID number (or code) is 3. This number is used to identify Tim when creating or updating records in RightNow.

Path: Common Configuration>Double-Click Staff Accounts

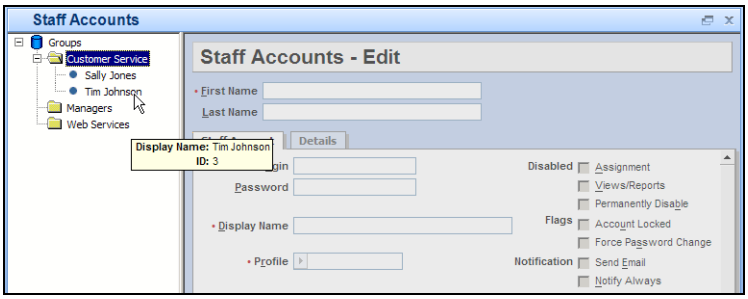

Figure 1: Mousing Over a Staff Account

Figure 2 shows the mouseover function for a custom field menu item. In this example, a menu item (Prepay) within the answer custom field Calling Plan is being referenced. The mouseover function shows that the menu item Prepay is associated with ID number (or code) 8, which is used to identify the Prepay menu item ID when creating or updating records in RightNow.

Path: Service>Double-Click Answer Custom Fields

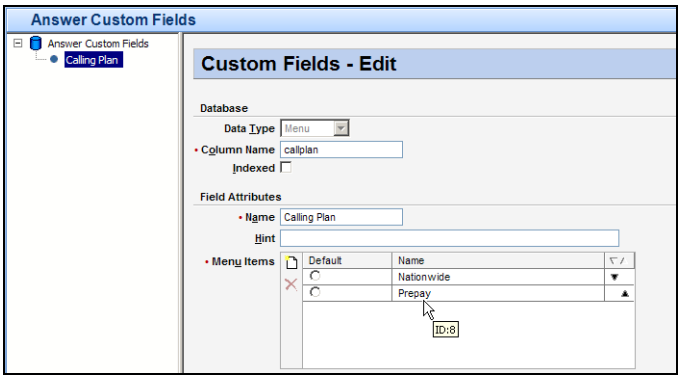

Figure 2: Mousing Over a Custom Field Menu Item

Most record tabs on the RightNow Console display the Info button in the upper right-hand corner of the tab. When you click the Info button, record details are displayed, including the record ID number. Figure 3 shows the details displayed when you mouse over the Info button when editing an organization. In this example, the organization ID is the code number. The Info button can be used when editing most records, such as contacts, opportunities, campaigns, and tasks.

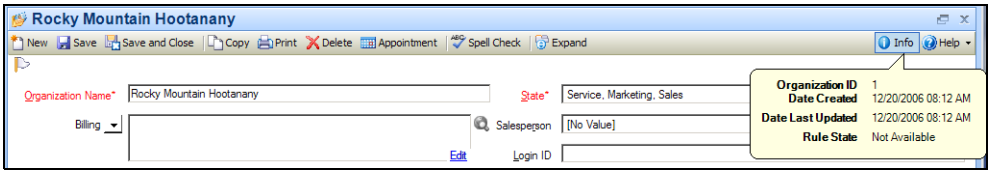

Figure 3: Displaying Record Information

Finding IDs in analytics

When exploring reports, you can view the report ID (ac_id) by displaying the ID column in the explorer details. For information on customizing explorer details, refer to *RightNow Administrator Manual*. Figure 4 displays the Reports Explorer with the ID column visible.

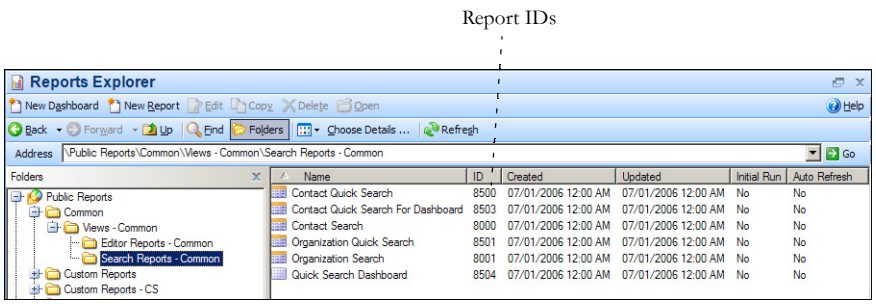

Figure 4: Finding the Report ID

Using the lookup_id_for_name function

In addition to using the mouseover functionality and the Info button, you can also use an XML function, lookup_id_for_name, which will find the code number of an item and return the value by email or in a variable used later in your XML. (For information on passing the parameter to another function, refer to "Passing variable IDs" on page 87.) This function passes two pairs, name and tbl, nested under an args parameter. The name pair is used to pass

the name of the string. The tbl pair is used to pass the number of the table the code item belongs to. The numbers of each table are listed in Table 16, along with the field looked up by the function.

Table 16: Table Numbers for tbl Parameter

Table Name	Number	Lookup Field
accounts	24	login
categories (hier_menu)	14	name
contacts	2	email, email_alt1, email_alt2
dispositions	37	name
incidents	1	ref_no
menu_items	20	name
orgs	3	name
products (hier_menu)	13	name

The following example shows lookup_id_for_name being used to find the ID number of a product. The returned value is then passed to the incident_update function. For information on passing variables, refer to "Passing variable IDs" on page 87.

Example:

```
<connector ret_type="email">
<function name="lookup_id_for_name" id="prodid">
     <parameter name="args" type="pair">
          <pair name="tbl" type="integer">13</pair>
          <pair name="name" type="string">Cell Phones</pair>
     </parameter>
</function>
<function name="incident_update">
     <parameter name="args" type="pair">
          <pair name="prod_lvl1" type="integer">$prodid</pair>
          <pair name="i_id" type="integer">9</pair>
     </parameter>
</function>
</connector>
```

In this example, the function looks up the code number for the product Cell Phones, and uses a variable, prodid, to use this code to update the incident with ID number 9.

Implementing code for the XML API

The following sections describe the two methods you can use to implement code for use with the RightNow XML API, and using the XML API log.

Using the POST method

When using the POST method, the XML is immediately sent to RightNow and parsed by the PHP script (*parse.php*). Record data is then instantly created, updated, or deleted in the RightNow database. The *parse.php* script is located at:

```
http://<your_domain>/cgi-bin/<your_interface>.cfg/php/xml_api/parse.php
```

To develop the integration, you will need to create code operating independently or within the HTML on a separate web page to post the XML data. The posted data must pass two parameters: xml_doc and sec_string. The xml_doc parameter contains the entire set of XML data, including the <connector> and </connector> tags and all XML contained within the tags. The sec_string parameter should specify the XML trigger phrase specified in the II_SEC_WEB_STRING configuration setting (refer to Table 17 on page 95).

Note The encoding of *parse.php* is set to UTF-8, and any XML document passed to the parser must also be UTF-8 encoded.

A simple way to use the POST method to send XML to RightNow is to create a web form using HTML, as shown in the following example:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<form method="POST" action="http://<your_domain>/cgi-bin/
<your_interface>.cfg/php/xml_api/parse.php" name="XML Form">
     <h2>XML Data</h2>
     <textarea cols="80" name="xml_doc" rows="20"></textarea>
          <br><br>
     <h2>Security String</h2>
     <input name="sec_string" size="10" value="xml">
          <br><br>
     <input type="submit" value="Submit" name="B1">
     <input type="reset" value="Reset" name="B2">
</form>
```

In this example, a web form with two text boxes (for XML data and to pass the security string) is created. The security string text box is prepopulated with the value specified in the II_SEC_WEB_STRING configuration setting (in this case, "xml"). You can then use this web page to enter XML data and submit it to *parse.php*.

You can also post XML data to RightNow using this method but without using a web page, by directly opening a socket connection to *parse.php*. You can accomplish this using any scripting language, such as PHP. Using this method, you must establish a connection with RightNow, and then use POST to pass your XML data.

RightNow Technologies Professional Services can assist you in determining which XML integration method best suits your needs and then implementing the method. For more information, contact your RightNow account manager.

Sending an XML-formatted email

You can add, update, delete, and retrieve data or perform a search or lookup function through the RightNow API by sending an XML-formatted email to a RightNow mailbox. The email must have a trigger word or phrase in the subject line that is specified in the RightNow configuration settings. When RightNow receives the email, the utility *techmail* will identify it as XML through the trigger word or phrase. The email will then be parsed by a PHP script to retrieve the data.

Important XML-formatted email messages must be in plain text.

You must configure RightNow to identify email that contains data in XML format. Through the configuration setting II_SEC_EMAIL_STR, you can specify a value for a trigger phrase to be used in the subject line of the email. The value specified by this configuration setting must be matched exactly, including case, to identify the email as XML. You can also configure RightNow to send an email message if there are any errors during the XML integration.

The configuration settings are located under RightNow Common>External Events>Incoming Integration and are detailed in Table 17.

Table 17: Incoming Integration Configuration Settings

Configuration Setting	Description
II_EMAIL_ERROR_ADDR	Specifies the email address to send XML API error data. Default is blank.

Table 17: Incoming Integration Configuration Settings (Continued)

Configuration Setting	Description
II_SEC_EMAIL_STR	Specifies the subject line of the email to be compared for validation of the XML source for email integrations. Default is blank.
II_SEC_WEB_STR	Specifies the security variable to be compared for validation of the XML source for integrations that use the POST method. Default is blank.

Error codes

When an XML API function fails, an error code will be returned. Error codes can be used to troubleshoot your integration. The tables in the section include the existing error codes for the following functions: contact_create, contact_update, flow_execute, mailing_send_to_contact, org_create, and org_update.

Note Generally, positive return values indicate success, and negative return values indicate an error. If an invalid parameter is used with the XML API get functions, a blank return value will be returned.

Table 18 describes the error codes for the contact_create function.

Table 18: Error Codes for contact_create

Error Code	Description
-100079801	Invalid email address
-100079802	Invalid ID number
-100079803	Invalid login

Table 19 describes the error codes for the contact_update function.

Table 19: Error Codes for contact_update

Error Code	Description
-100089801	Invalid email address

Table 19: Error Codes for contact_update (Continued)

Error Code	Description
-100089802	Invalid ID number
-100089803	Invalid login

Table 20 describes the error codes for the flow_execute function.

Table 20: Error Codes for flow_execute

Error Code	Description
-100049751	Contact update failed
-100049752	Flow failed
-100049753	Flow is not set to Launched

Table 21 describes the error codes for the mailing_send_to_contact function.

Table 21: Error Codes for mailing_send_to_contact

Error Code	Description
-100030001	No available mailing message
-100030002	No valid email addresses
-100030003	Mail generate error
-100030004	Filter error
-100030005	Suppressed email address
-100030006	Excluded from audience
-100030007	Exceeded recency limit
-100030008	Exceeded frequency limit

Table 21: Error Codes for mailing_send_to_contact (Continued)

Error Code	Description
-100030009	Opted out

Table 22 describes the error codes for the org_create function.

Table 22: Error Codes for org_create

Error Code	Description
-100059802	Invalid ID number
-100059803	Invalid login
-100059851	Invalid parent organization
-100059853	Organization destination changed

Table 23 describes the error codes for the org_update function.

Table 23: Error Codes for org_update

Error Code	Description
-100069802	Invalid ID number
-100069803	Invalid login

Using the XML API log

The XML API log allows you to view a record of all XML functions passed to your RightNow site though the API. Each function that was performed through the XML API is listed, along with the IP address that passed the function, and the date and time it was performed. All functions are listed, regardless of whether an error occurred during the processing of the function.

The XML API log is accessed through the Common Configuration section of the RightNow Console. You can use the log to track activity through the XML API and ensure security is maintained by monitoring the IP addresses that pass functions to your site.

Path: Common Configuration>Double-Click XML API Log

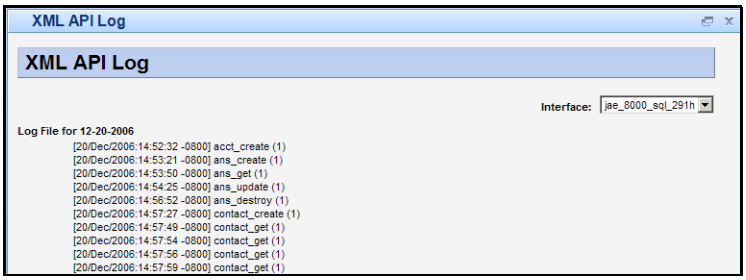

Figure 5: XML API Log

4

Event Handlers

Through the RightNow event handlers feature, you can define custom processes for managing your incidents, contacts, organizations, answers, and opportunities. For example, if you need to maintain your own incident database, you can create an event handler that automatically copies new incidents from RightNow to your external database. These types of event handlers are ideal for building real-time interfaces between RightNow and third-party software applications, such as those used in external help desks, call centers, or reporting systems.

The following types of external events are supported by RightNow:

- **Insert events**—This type of event occurs whenever a customer or staff member creates a new incident, answer, contact, organization, or opportunity.
- **Update events**—This type of event occurs whenever a customer or a staff member updates an existing incident, answer, contact, organization, or opportunity.
- **Delete events**—This type of event occurs whenever a customer or a staff member deletes an existing incident, answer, contact, organization, or opportunity.

There are two ways to handle external events in RightNow. You can specify the location of a PHP script that directs the handling of an event (external events), or you can email the event data to a specified mailbox (email integration). This chapter contains procedures for both of these methods.

Important You must contact RightNow Technologies Professional Services to develop and implement external events. This is a chargeable service. For more information, contact your RightNow account manager.

External event handlers

The external event handlers can be enabled to run a specified PHP script when an incident, answer, contact, organization, or opportunity is created, updated, or deleted. When an external event occurs, a data file (CSV) is created. The data is then handled under the direction of the PHP script specified in your configuration settings. For example, you can create a script that will export specified incident data to an external Oracle database.

Important You must contact RightNow Technologies Professional Services to develop and implement external events. This is a chargeable service. For more information, contact your RightNow account manager.

Enabling external events

Enabling external events requires configuring the insert, update, or delete handlers. The event handler configuration settings are located under RightNow Common>External Events. These settings are described in Table 24.

Table 24: External Events Configuration Settings

Setting	Usage
EE_INC_DELETE_HANDLER	Specifies the relative path name of a PHP script to be used to externally process incident delete events. If no handler is specified, no external action is taken. Default is blank.
EE_INC_INSERT_HANDLER	Specifies the relative path name of a PHP script to be used to externally process incident insert events. If no handler is specified, no external action is taken. Default is blank.
EE_INC_UPDATE_HANDLER	Specifies the relative path name of a PHP script to be used to externally process incident update events. If no handler is specified, no external action is taken. Default is blank.
EE_CONTACT_DELETE_ HANDLER	Specifies the relative path name of a PHP script to be used to externally process contact delete events. If no handler is specified, no external action is taken. Default is blank.

Table 24: External Events Configuration Settings (Continued)

Setting	Usage
EE_CONTACT_INSERT_ HANDLER	Specifies the relative path name of a PHP script to be used to externally process contact insert events. This is used to provide an interface for third-party call management systems or other third-party systems. If no handler is specified, no external action is taken. Default is blank.
EE_CONTACT_UPDATE_ HANDLER	Specifies the relative path name of a PHP script to be used to externally process contact update events. This is used to provide an interface for third-party call management systems or other third-party systems. If no handler is specified, no external action is taken. Default is blank.
EE_ANS_DELETE_HANDLER	Specifies the relative path name of a PHP script to be used to externally process answer delete events. This is used to provide an interface for third-party call management systems or other third-party systems. If no handler is specified, no external action is taken. Default is blank.
EE_ANS_INSERT_HANDLER	Specifies the relative path name of a PHP script to be used to externally process answer insert events. This is used to provide an interface for third-party call management systems or other third-party systems. If no handler is specified, no external action is taken. Default is blank.
EE_ANS_UPDATE_HANDLER	Specifies the relative path name of a PHP script to be used to externally process answer update events. This is used to provide an interface for third-party call management systems or other third-party systems. If no handler is specified, no external action is taken. Default is blank.
EE_ORG_DELETE_HANDLER	Specifies the relative path name of a PHP script to be used to externally process organization delete events. This is used to provide an interface for third-party call management systems or other third-party systems. If no handler is specified, no external action is taken. Default is blank.
EE_ORG_INSERT_HANDLER	Specifies the relative path name of a PHP script to be used to externally process organization insert events. This is used to provide an interface for third-party call management systems or other third-party systems. If no handler is specified, no external action is taken. Default is blank.

Table 24: External Events Configuration Settings (Continued)

Setting	Usage
EE_ORG_UPDATE_HANDLER	Specifies the relative path name of a PHP script to be used to externally process organization update events. This is used to provide an interface for third-party call management systems or other third-party systems. If no handler is specified, no external action is taken. Default is blank.
EE_OPP_DELETE_HANDLER	Specifies the relative path name of a PHP script used to externally process opportunity delete events. This is used to provide an interface for third party call management systems or other third party systems. If no handler is specified, no external action is taken. Default is blank.
EE_OPP_INSERT_HANDLER	Specifies the relative path name of a PHP script used to externally process opportunity insert events. This is used to provide an interface for third party call management systems or other third party systems. If no handler is specified, no external action is taken. Default is blank.
EE_OPP_UPDATE_HANDLER	Specifies the relative path name of a PHP script used to externally process opportunity update events. This is used to provide an interface for third party call management systems or other third party systems. If no handler is specified, no external action is taken. Default is blank.

Developing external events

When you activate an insert, update, or delete handler, the PHP script you develop to handle the event becomes an extension of RightNow. Event handlers must be written in PHP.

When developing event handlers, RightNow will:

1 Create a CSV file. The file format will be the same as that produced by the *kexport* utility. The data field names and actual data provided will be determined according to the template file. For information about *kexport*, refer to the *RightNow Administrator Manual*.

2 Execute the appropriate PHP script for handling the insert, update, or delete event. The names of the temporary files containing the incident, answer, contact, organization, or opportunity data are passed using the variable ee_file_name.

Important Your event handler must reside in the <cgi-bin>/<interface>.cfg/scripts/ext_evt directory.

3 Wait for the event handler to terminate, and then continue with normal processing.

Your custom event handler will:

1 Retrieve from the ee_file_name variable the name of the temporary file containing incident, answer, contact, organization, or opportunity data.

2 Open and parse the temporary files to retrieve incident, answer, contact, organization, or opportunity data.

3 Perform any custom processing.

4 Delete the temporary files.

5 Terminate and return control to RightNow within an acceptable amount of time so as not to degrade overall system performance.

Email integration

The email integration in RightNow allows you to email data to a mailbox when an incident, answer, contact, organization, or opportunity is created, updated, or deleted. When the event occurs, an email is sent immediately to the specified mailbox with the incident, answer, contact, organization, or opportunity data.

Enabling email integration requires configuring the insert, update, or delete handlers. The email integration configuration settings are located under RightNow Common>External Events>Email Integration. Use these settings to specify the email address to which you want to send email event data.

The email integration configuration settings are described in Table 25.

Table 25: Email Integration Configuration Settings

Setting	Description
EI_INC_DELETE_ADDR	Specifies the email address to receive incident delete data. If no address is specified, no external action is taken. Default is blank.
EI_INC_INSERT_ADDR	Specifies the email address to receive incident insert data. If no address is specified, no external action is taken. Default is blank.
EI_INC_UPDATE_ADDR	Specifies the email address to receive incident update data. If no address is specified, no external action is taken. Default is blank.
EI_CONTACT_DELETE_ADDR	Specifies the email address to receive contact delete data. If no address is specified, no external action is taken. Default is blank.
EI_CONTACT_INSERT_ADDR	Specifies the email address to receive contact insert data. If no address is specified, no external action is taken. Default is blank.
EI_CONTACT_UPDATE_ADDR	Specifies the email address to receive contact update data. If no address is specified, no external action is taken. Default is blank.
EI_ANS_DELETE_ADDR	Specifies the email address to receive answer delete data. If no address is specified, no external action is taken. Default is blank.

Table 25: Email Integration Configuration Settings (Continued)

Setting	Description
EI_ANS_INSERT_ADDR	Specifies the email address to receive answer insert data. If no address is specified, no external action is taken. Default is blank.
EI_ANS_UPDATE_ADDR	Specifies the email address to receive answer update data. If no address is specified, no external action is taken. Default is blank.
EI_ORG_DELETE_ADDR	Specifies the email address to receive organization delete data. If no address is specified, no external action is taken. Default is blank.
EI_ORG_INSERT_ADDR	Specifies the email address to receive organization insert data. If no address is specified, no external action is taken. Default is blank.
EI_ORG_UPDATE_ADDR	Specifies the email address to receive organization update data. If no address is specified, no external action is taken. Default is blank.
EI_OPP_INSERT_ADDR	Specifies the email address to receive opportunity insert data. If no address is specified, no external action is taken. Default is blank.
EI_OPP_UPDATE_ADDR	Specifies the email address to receive opportunity update data. If no address is specified, no external action is taken. Default is blank.
EI_OPP_DELETE_ADDR	Specifies the email address to receive opportunity delete data. If no address is specified, no external action is taken. Default is blank.

Creating templates for email integration

When using the email integration in RightNow, you can create template files that specify the data sent by email following an event. You can create up to five template files and upload them to the *integration files* directory in File Manager. For more information about uploading files through the File Manager, refer to the *RightNow Administrator Manual*.

To upload a file to the *integration files* directory in File Manager, the file name must be in the following format:

- *incident.tmpl*—This template determines the data sent when an incident event (create, update, or delete) occurs.
- *ans.tmpl*—This template determines the data sent when an answer event (create, update, or delete) occurs.
- *contact.tmpl*—This template determines the data sent when a contact event (create, update, or delete) occurs.
- *org.tmpl*—This template determines the data sent when an organization event (create, update, or delete) occurs.
- *opp.tmpl*—This template determines the data sent when an opportunity event (create, update, or delete) occurs.

The template file will contain three components. The first line of the template specifies the reply-to address of the email. The second line of the template specifies the subject of the email. The remaining lines determine the content of the email. These lines can contain actual text, as well as variable information designated in pipes (|). Any text contained in pipes should be in the format table_name.column_name.

Important You can specify any field definition columns in the table related to the external event (*answers, contacts, incidents, orgs,* or *opportunities*). You can also define output for any table directly related to the external event table. For example, you can require contact output in the *incident.tmpl* file because a contact should be directly related to each incident.

The following is an example of an *incident.tmpl* file:

```
jsmith@example.com
Email Integration
Reference Number: |incidents.ref_no|
Subject: |incidents.subject|
Product: |incidents.prod_lvl1|
         |incidents.prod_lvl2|
```

In this example, the reply-to address of the email will be "jsmith@example.com" and the subject line of the email will be "Email Integration." The body of the email will look like the following:

```
Reference Number: 010620-000003
Subject: Incident Title
Product: Integration
```

5

Pass-Through Authentication

You can integrate RightNow Service with an external customer validation source to allow your customers to automatically log in to RightNow Service from an external web page. The external source supplies login parameters to RightNow Service by placing them in the URL of the Support Home page. In this way, customers will not have to provide customer login data twice if you are using an external customer validation source. The contact information will also be shared between the external source and RightNow Service, so contacts can be created and updated during the login to RightNow Service.

To perform this integration, customers must be redirected when attempting to access or log in to RightNow Service. When the login parameters are passed back to RightNow Service, the customer will be logged in if the information passed is sufficient to identify an existing contact or create a new contact. An existing contact is identified by matching the email field and login field of the *contacts* table in the database. When an existing contact is found, the customer is logged in as that contact and is updated if any additional or new contact information is passed to RightNow Service.

If an existing contact is not found, a new contact is created from the data provided and the customer is logged in to RightNow Service as the new contact. If the contact information passed does not contain all required fields to create a new contact, RightNow Service can be configured to redirect the customer to an alternate URL.

Important When using pass-through authentication, the configuration setting EGW_AUTO_CUST_CREATE should be set to No to prevent contact records from being created through email before they are created by a pass-through authentication event. This will help eliminate login issues caused by mismatched user names and passwords.

If you set EGW_AUTO_CUST_CREATE to No, you should also modify the message base NOT_REG_EMAIL_MSG to direct new end-users to your portal site to register and create an account.

Although contacts can be created and updated through the pass-through authentication integration, deletion of contacts must be handled by manually deleting the contact from the knowledge base though the RightNow Console or another integration method, such as the XML API.

Note Contact your RightNow account manager for assistance in customizing your pass-through authentication beyond the procedures detailed in this chapter (for example, securing pass-through authentication strings beyond Base 64 encoding standards).

Refer to Figure 6 for assistance in designing your login integration. This figure can help you determine the process used by RightNow Service when pass-through authentication is used.

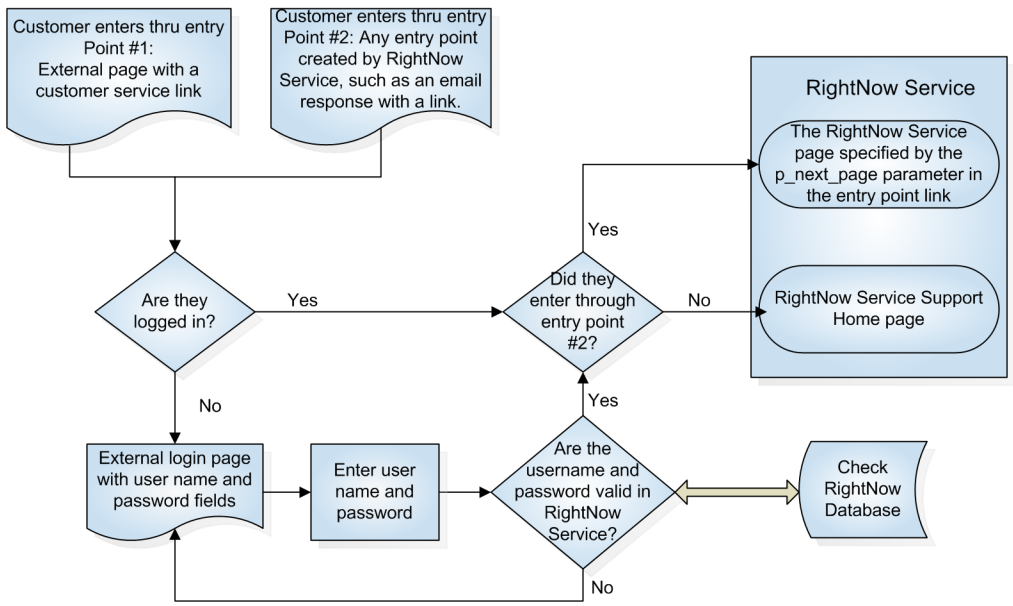

Figure 6: Pass-Through Authentication Flow Chart

Configuring RightNow Service

Before you can perform a pass-through authentication integration with an external source, you must configure RightNow Service to prevent customers from accessing specific options without a proper login. Then you must redirect the login to the URL of your external validation source.

Requiring a login to RightNow Service

To integrate RightNow Service with an external customer validation source, RightNow Service can be configured to require a login to the end-user interface (excluding the Site Feedback page). This ensures that contact information is passed directly to the login page and prevents customers from accessing their account information through the end-user pages. However, customers will also be required to log in when clicking the link in an incident email to respond or update their incident.

To configure RightNow Service to require a login:

1. Click Common Configuration.
2. Double-click Settings under System Configuration.
3. Select RightNow User Interface.
4. Click End-User Interface>Support Home Page Display>SHP_PASSWD_REQD, and click Yes for the value (No is the default).
5. Click the Update button followed by the Commit and Exit button to save your changes and return to the General Configuration Menu.

Redirecting the RightNow Service login

A configuration setting must also be enabled when using the integration to specify the URL to which a customer is redirected if attempting to log in to RightNow Service, or if the external login information supplied to RightNow Service is not adequate to create a new account or use an existing account. When a URL value is specified for this configuration setting, the passed login parameters must provide data for the minimum required fields needed to log in to RightNow Service (p_userid, p_passwd) or create a new contact in RightNow Service (p_userid, p_passwd, p_email.addr). (In most cases, it is recommended that you pass back all URL parameters to RightNow Service that RightNow Service passed during the redirection.) Even if the configuration setting TC_CT_EMAIL_REQD is disabled, the specified fields are still required. If the required fields are not passed, the customer is redirected to the specified

URL. You can create a new site at this URL to either inform the customer that their access is denied or create a form to gather additional required information and re-pass the parameters to RightNow Service.

Important If additional required contact custom fields have been created, these will also need to be passed to create a new account.

Note RightNow Service will automatically append your customer's session ID information to the URL when the customer is redirected through the end-user pages. The specified page must be configured to accept the session ID.

To configure RightNow Service to redirect the login:

1 Click Common Configuration.

2 Double-click Settings under System Configuration.

3 Select RightNow User Interface.

4 Click My Stuff>Security>MYSEC_EXT_LOGIN_URL.

5 Type the desired URL in the Value text box and click the Update button.

6 Click the Commit and Exit button to save your changes and return to the General Configuration Menu.

Note URLs sent to contacts via email (for example, a link to update the incident) will use the URL specified in the MYSEC_EXT_LOGIN_URL configuration setting.

If you are passing a non-blank password via p_passwd in a pass-through authentication event and EU_CUST_PASSWD_ENABLED is disabled, the pass-through authentication event will fail. It is recommended that you enable EU_CUST_PASSWD_ENABLED when using pass-through authentication and use TC_CT_PASSWD_DISP to control the look and feel of contact passwords on the administration side of RightNow. TC_CT_PASSWD_DISP does not affect pass-through authentication.

If you use pass-through authentication to add a contact record, but the email address already exists in the knowledge base (associated with another contact record), RightNow will add the record; however, it will append ".0001" to the end of the email address. This will also occur if you try to update a contact record in order to change the email address, but the email address already exists in the knowledge base. If this occurs repeatedly with the same email address, the appended number will automatically increment (for example, .0002, .0003, etc.).

Implementing a customer login script

To develop a login parameters integration, you will need to embed code within your login script to format a URL that will pass data from your external validation source to RightNow Service. The embedded code can be written in any scripting language, including PHP, JSP, or ASP. The login parameters from the external validation source must be encoded using Base 64 encoding and placed in the RightNow Service URL from the desired page. In addition to using the Base 64 function, certain characters must also be replaced in the URL, as shown in "PHP Example:" on page 118 (+ becomes _, / becomes ~, and = becomes *).

Note You must use a login script for every link from your web site to RightNow Service. If contacts exit the RightNow Service end-user pages and re-enter later in their session, they will not be automatically logged in. Therefore, we recommend that all links to the end-user interface contain pass-through data.

The following format should be used:

UNIX:

```
http://<your_domain>/cgi-bin/<your_interface>.cfg/php/enduser/
entry.php?p_li=<encoded login parameters>
```

Windows:

```
http://<your_domain>/scripts/<your_interface>.cfg/php.exe/enduser/
entry.php?p_li=<encoded login parameters>
```

Note You can replace entry.php with any end-user page in RightNow Service (for example, std_alp.php), or use the p_next_page parameter to return the customer to their original RightNow Service page. Refer to "PHP Example:" on page 118.

The parameters to be passed to RightNow Service are detailed in Table 26.

Table 26: Parameter Descriptions

Parameter	Description
p_userid	This parameter represents the login field in the *contacts* table of the RightNow database. This field is required to log in and create a new contact, and cannot be updated via pass-through authentication.

Table 26: Parameter Descriptions (Continued)

Parameter	Description
p_passwd	This parameter represents the password field in the *contacts* table of the RightNow database (limited to 20 characters). This field is required to log in and create a new contact, or log in as an existing contact, and cannot be updated via pass-through authentication. The value can be NULL. **Note:** We recommend that the password specified in the *contacts* table be different than that stored in your external database. This is because the customer's RightNow Service password cannot be updated later by the external system, since the password is used as a verification field by RightNow Service. Therefore, to prevent customers who change their password in your external system from being locked out of the RightNow Service end-user pages, you should create a different password when the contact is created, and use this password consistently to log in the customer to RightNow. One way to accomplish this is to use a constant value for all contact passwords and use the value each time a customer logs in. You could also encrypt the contact's user id and use the encryption as the contact's password. Each time the customer's login parameters are passed to RightNow Service, you can use your encryption script to pass the valid password.
p_email.addr	This parameter represents the email field in the *contacts* table in the RightNow database. This field is required to log in and create a new contact. **Note:** The value of this field must be unique.
p_title	This parameter represents the title field in the *contacts* table in the RightNow database.
p_name.first	This parameter represents the first_name field in the *contacts* table in the RightNow database.
p_name.last	This parameter represents the last_name field in the *contacts* table in the RightNow database.
p_alt_name.first	This parameter represents the alt_first_name field in the *contacts* table in the RightNow database.
p_alt_name.last	This parameter represents the alt_last_name field in the *contacts* table in the RightNow database.
p_email_alt1.addr	This parameter represents the email_alt1 field in the *contacts* table in the RightNow database.

Table 26: Parameter Descriptions (Continued)

Parameter	Description
p_email_alt2.addr	This parameter represents the email_alt2 field in the *contacts* table in the RightNow database.
p_addr.street	This parameter represents the street field in the *contacts* table in the RightNow database.
p_addr.city	This parameter represents the city field in the *contacts* table in the RightNow database.
p_addr.postal_code	This parameter represents the postal_code field in the *contacts* table in the RightNow database. This field may not contain special characters (for example, 59715-1111 should be passed as 597151111).
p_addr.country_id	This parameter represents the country_id field in the *contacts* table in the RightNow database. This field should be passed as a country's ID number. To find the value of menu items, refer to "Finding code numbers" on page 89.
p_addr.prov_id	This parameter represents the prov_id field in the *contacts* table in the RightNow database. This field should be passed as a state or province's ID number. To find the value of menu items, refer to "Finding code numbers" on page 89.
p_ph_office	This parameter represents the ph_office field in the *contacts* table in the RightNow database.
p_ph_mobile	This parameter represents the ph_mobile field in the *contacts* table in the RightNow database.
p_ph_fax	This parameter represents the ph_fax field in the *contacts* table in the RightNow database.
p_ph_asst	This parameter represents the ph_asst field in the *contacts* table in the RightNow database.
p_ph_home	This parameter represents the ph_home field in the *contacts* table in the RightNow database.

Table 26: Parameter Descriptions (Continued)

Parameter	Description
p_ccf_*	The parameter p_ccf_* represents a contact custom field in RightNow. The * should be replaced with the number of the cf_id for the contact custom field. If this is a menu custom field, the numbers (not the actual text) for each menu item must be specified as the value in the integration login code. To find the value of menu items, refer to "Finding code numbers" on page 89.
p_li_expiry	This parameter represents the time the login session will last before expiring. When the session expires, the contact will be required to resubmit their login on the page specified by the MYSEC_EXT_LOGIN_URL configuration setting. Your login form should calculate the expiration timestamp and pass it back to RightNow Service. **Note:** If the p_li_expiry parameter is used in combination with the p_redirect parameter, the p_li_expiry parameter is overridden, and the value in the MYSEC_SESSION_ID_EXP configuration setting is used instead.
p_li_passwd	This parameter represents the string specified in the MYSEC_LI_PASSWD configuration setting. **Note:** This parameter is required if the MYSEC_LI_PASSWD configuration setting contains a value.
p_org_id	This parameter represents an organization ID to associate with a contact. To find the value of menu items, refer to "Finding code numbers" on page 89. **Note:** You must manually assign any service level agreements (SLA) that you want to associate with the organization, including those controlling privileged access. You can do this through RightNow's administration interface.
p_redirect	This parameter is added to the p_li variable and is used to remove the p_li variable from the URL. When p_redirect is set to 1, the p_li variable will be replaced by the p_sid variable in the URL when the user logs into the site. This will prevent secure information from being passed in the p_li variable if the end-user copies and pastes the URL from their browser and emails it to someone.
p_state.css	This parameter represents the contact's state for RightNow Service. • 0—Disabled • 1—Enabled

Table 26: Parameter Descriptions (Continued)

Parameter	Description
p_state.ma	This parameter represents the contact's state for RightNow Marketing. • 0—Disabled • 1—Enabled
p_state.sa	This parameter represents the contact's state for RightNow Sales. • 0—Disabled • 1—Enabled

The following examples show how to generate a form to pass login parameters to RightNow Service using PHP and ASP.Net code. You can retain all query_string parameters and append key-value pair parameters per the following examples.

Note To understand these scripts better, it will help to replace certain variables. Replace <your_domain> with the domain name used by your RightNow site, <your_interface> with your interface name, and <li_password> with the string specified in MYSEC_LI_PASSWD. In addition, specify "cgi-bin" and "php" for UNIX or "scripts" and "php.exe" for Windows.

Caution The following examples are for illustrative purposes only, and will be improperly formatted if you attempt to cut and paste directly from the following text.

PHP Example:

```
<?php
  header("Content-type: text/html; charset=UTF-8");
//
//NOTE: It is necessary to overwrite the html header to explicitly define
//the character set that will be used to encode PTA user data.  It is not
//possible to do this with a meta equiv tag because a header already
//exists, and the existing header will take precedence.  If the character
//set is not defined here, it is likely that the browser of the end user
//will select another default encoding (such as Latin-1). Data encoded
//using character sets other than UTF-8 can cause server-side SQL errors
//when the server is attempting to parse the user data coming from PTA.
//
// ***** THIS IS JUST AN EXAMPLE AND NOT INTENDED FOR PRODUCTION USE *****
```

```
//Use this script to see an illustrated example of how login integration
//is supposed to work. This script will generate a form that requests a
//login/password and other optional information. It submits this data
//back to itself (with $li_reentry set), sets up the appropriate
//parameters (important ones passed in from RNW) and redirects
//back to RNW.
// ------------------------------------------------------------
// Site specific variables

$script_name = 'li.php';
$domain = '<your_domain>';
$script_dir = '<cgi-bin or scripts>';
$interface = '<your_interface>';
$mysec_li_passwd = '<li_password>';
$php_bin = '<php or php.exe>';
// ------------------------------------------------------------
// Function definitions

function urlsafe_encode(&$str)
{
     return(strtr(base64_encode($str),
          array('+' => '_', '/' => '~', '=' => '*')));
}

function urlsafe_decode(&$str)
{
     return(base64_decode(strtr($str,
          array('_' => '+', '~' => '/', '*' => '='))));
}
// ------------------------------------------------------------
// Process the form & redirect

if ($li_reentry) {
     $li_data = array(
          'p_userid'     => $li_userid,
          'p_passwd'     => $li_passwd,
          'p_email.addr'      => $li_email,
          'p_name.first' => $li_first_name,
          'p_name.last'  => $li_last_name,
     // sample text contact custom field (custom_fields.cf_id== 1)
```

```
            'p_ccf_1'      => $li_ccf_1,
        // sample menu contact custom field (custom_fields.cf_id == 3)
            'p_ccf_3'      => intval($li_ccf_3),
        // p_li_passwd must match the MYSEC_LI_PASSWD config setting
            'p_li_passwd'  => $mysec_li_passwd
        );

        // set up the $p_li variable
        while (list($key, $val) = each($li_data))
            $p_li .= sprintf("%s%s=%s", $p_li ? '&' : '', $key,
            $val);
        $p_li = urlsafe_encode($p_li);

        // retain all the important query_string parameters passed in from
        //RNW (excluding the special cases and the li_* form parameters)
        while (list($key, $val) = each($HTTP_GET_VARS)) {
            if (($key != 'p_next_page') &&
                ($key != 'p_li') &&
                (substr($key, 0, 3) != 'li_'))
                $parms .= sprintf("&%s=%s", $key,
                urlencode($val));
        }

        // default next page to support home
        if (!isset($p_next_page))
            $p_next_page = "home.php";
        // redirect back to RNW
        header("Location: http://$domain/$script_dir/      \
        $interface.cfg/$php_bin/enduser/$p_next_page?p_li  \
        =$p_li$parms");
        exit;
    }

    // ------------------------------------------------------------
    // Display the form
    ?>
    <html>
    <body>

    <h2> Login Integration </h2>
```

```
<form action="<? print($script_name) ?>">
<input type="hidden" name="li_reentry" value="1">
<?
// retain all the important query_string parameters passed in from RNW
while (list($key, $val) = each($HTTP_GET_VARS)) {
     print("<input type=\"hidden\" name=\"$key\"     \
     value=\"$val\">\n");
}
?>
Login: <input type="text" name="li_userid"><br />
Password: <input type="password" name="li_passwd"><br />
Email: <input type="text" name="li_email"><br />
First Name: <input type="text" name="li_first_name"><br />
Last Name: <input type="text" name="li_last_name"><br />
Contact Custom 1: <input type="text" name="li_ccf_1"><br />
Contact Custom 3: <input type="text" name="li_ccf_3"><br />
<input type="submit">
</form>

</body>
</html>
```

ASP.Net Example:

```
Imports System
Imports System.Text
' ***** THIS IS ONLY AN EXAMPLE AND NOT INTENDED FOR PRODUCTION USE *****
'Use this script to see an illustrated example of how login integration
'is supposed to work.

Public Class login
     Inherits System.Web.UI.Page

Web Form Designer Generated Code
     Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
     System.EventArgs)
     Handles MyBase.Load
          Dim redirectLink, URLParams As String
          Try
               'LDAPService is an internal web service to look up user
               'info.
               Dim ldapWebService As New LDAPService.services
               Dim aUser As LDAPService.userInfo
               aUser = ldapWebService.GetUser(getUserName())
               'begin forming url
               redirectLink = "http://<your_domain>/cgi-bin/
               <your_interface>.cfg/php/enduser/" & getNextPage()
               'add parameters
               URLParams = "p_userid=" & aUser.userid &
     "&p_passwd=blank&p_email.addr=" & aUser.email & "&p_name.first=" &
     aUser.first_name & "&p_name.last=" & aUser.last_name
               'convert URLParams to a byte array
               Dim asciiEncoding As Encoding = Encoding.ASCII
               Dim byteArray(asciiEncoding.GetByteCount(URLParams)) As
               Byte
               byteArray = asciiEncoding.GetBytes(URLParams)
               'convert the byte array to a base64 string
               URLParams = Convert.ToBase64String(byteArray)

          Catch ex As Exception
               lblError.Text = "Error. Cannot log in. Unknown user."
          End Try
          Response.Redirect(redirectLink & "?p_li=" & URLParams, True)
     End Sub
```

```
    Private Function getNextPage() As String
        'get p_next_page parameter from request
        If Len(Request.Params("p_next_page")) > 0 Then
            Return Request.Params("p_next_page")
        Else
            Return "home.php"
        End If
    End Function

    Private Function getUserName() As String
        'gets the user name.
        Try
            Dim theUserName As String
            theUserName = Request.ServerVariables("AUTH_USER")
            Return theUserName
        Catch ex As Exception
            lblError.Text = "Error. Cannot log in. Unknown user."
        End Try
    End Function
End Class
```

Appendix A
Pair Names

This appendix describes the pairs available to be used in the public APIs. Each table contains the pairs available for the API, a description of the pairs, the pair type, and visibility of the pair for the different function types. The visibility indicators are:

- C—Visible for create functions
- D—Visible for delete functions
- G—Visible for get functions
- U—Visible for update functions

For example, if an account API pair has the visibility indicators C, G, and U, you can use that pair in the ans_create, ans_get, and ans_update functions; however, you can not use the pair in the ans_delete function.

Note Visibility indicators are not applicable to the search, css_category_move, css_disposition_move, css_product_move, or flow_execute functions.

Account API

The pairs described in the following table are available to use in account functions.

Table 27: Account Pairs

Name	Use	Type	Visibility
acct_id	The ID number of the account.	integer	CDGU
acd_group	The automatic call distribution group associated with a staff account.	string	CGU
acd_passwd	The automatic call distribution password associated with a staff account.	string	CGU
alt_name	The alternate name of the staff account. Uses the following nested pairs.	pair	CGU
first_name	The alternate first name of a staff account.	string	CGU

Table 27: Account Pairs (Continued)

Name	Use	Type	Visibility
last_name	The alternate last name of a staff account.	string	CGU
attr	A bitmap that determines the attribute statuses of the account. • 0—Fully enabled • 1—Assignment to the staff member is disabled • 2—Views and reports are disabled • 4—Account locked • 8—Force password change • 32—Permanently disabled	integer	CGU
country_id	The default country associated with a staff account.	integer	CGU
custom_field	A custom field associated with a staff account. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	CGU
def_curr_id	The default currency associated with a staff account.	integer	CGU
display_name	The display name associated with a staff account.	string	CGU
eas_id	The agent skills ID associated with a staff account.	string	CGU
email	The email address and security information associated with a staff account. Uses the following nested pairs.	pair	CGU
addr	The email address associated with the account.	string	CGU
cert	The S/MIME account certificate associated with the account.	string	CGU
email_notif	The email notification flag associated with a staff account.	integer	CGU

Table 27: Account Pairs (Continued)

Name	Use	Type	Visibility
group_id	The group ID associated with a staff account.	integer	CDG
last_event_id	The ID number of the last event.	integer	GU
login	The login associated with a staff account.	string	CGU
mgr	The management hierarchy. Uses the nested pairs lvl_id1 through lvl_id12.	pair	CGU
lvl_id<1–12>	The pair data defining the management hierarcy.	integer	CGU
name	The name associated with the staff account. Uses the following nested pairs.	pair	CGU
first	The first name associated with a staff account.	string	CGU
last	The last name of a staff account.	string	CGU
notif_cache	Cached notifications.	string	CGU
notif_pending	Indicates if notifications are pending. • 0—Not pending • 1—Pending	integer	CGU
old_terr	The old territory associated with a staff account.	integer	CU
passwd_history	The previous password for the account (encrypted).	string	G
password_text	The password associated with a staff account.	string	CGU
password_exp	The password expiration date of a staff account.	time	G
phone	The phone number associated with a staff account.	string	CGU

Table 27: Account Pairs (Continued)

Name	Use	Type	Visibility
phone_alt_1	The first alternative phone number associated with a staff account.	string	CGU
phone_alt_2	The second alternative phone number associated with a staff account.	string	CGU
profile_id	The profile ID associated with a staff account.	integer	CGU
seq	The sequence listing within a group folder that is associated with a staff account.	integer	CGU
signature	The signature associated with a staff account.	string	CGU
source_upd	The source of the account. Uses the following nested pairs.	pair	CGU
lvl_id1	The level-one source of the answer. Refer to Appendix B, "Source Codes," on page 177.	integer	CGU
lvl_id2	The level-two source of the answer. Refer to Appendix B, "Source Codes," on page 177.	integer	CGU
sp_dial	The speed dial items associated with a staff account. Uses the following nested pairs.	pair	CGU
sd_item	A speed dial entry. Uses the following nested pairs.	pair	CGU
name	The name of the speed dial entry.	string	CGU
phone	The phone number of the speed dial entry.	string	CGU
terr_id	The territory ID associated with a staff account.	integer	CGU
timezone	The default timezone associated with the staff account.	integer	CGU
upd_opt	The flag that updates opportunities when changing the territory of a staff account.	integer	CU

Answer API

The pairs described in the following table are available to use in answer functions.

Table 28: Answer Pairs

Name	Use	Type	Visibility
a_id	The ID number of the answer.	integer	CGUD
access_mask	The answer access of the answer. The access level determines which end-users can view the answer.	string	CGU
admin_last_access	The last time the answer was accessed by an administrator.	time	G
assigned	The staff member assigned to the answer. Uses the following nested pairs.	pair	CGU
acct_id	The ID number of the staff member assigned to the answer.	integer	CGU
group_id	The ID number of the staff group assigned to the answer.	integer	CGU
ault_solved_count	The long-term solved count for administrative users.	integer	CGU
aust_solved_count	The short-term solved count for administrative users.	integer	CGU
banner	The flag information associated with the answer.	pair	CGU
acct_id	The ID number of the staff account that most recently updated the flag.	integer	G
flag	The importance of the flag. • 1—Low • 2—Medium • 3—High	integer	CGU
txt	The flag text.	string	CGU
upd	The time the flag text was updated.	time	G

Table 28: Answer Pairs (Continued)

Name	Use	Type	Visibility
created	The time the answer was created.	time	G
custom_field	A custom field associated with the answer. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	CGU
description	The description of the answer.	string	CGU
ee_flag	Determines whether external events run when an answer is created, updated, or deleted. • 0—External events do not run • 1—External events run	integer	CU
eust_solved_count	The customers' short-term solved count for the answer.	integer	CGU
expires	The date the answer expires and is set to review answer status.	time	CGU
keywords	The keywords of the answer.	string	CGU
lang_id	The ID number of the answer's language.	integer	CGU
last_access	The date and time the answer was last accessed.	time	G
last_edited_by	The ID number of the staff member who last edited the answer.	integer	G
last_notify	The date and time a notification was last sent for the answer.	time	G
links	Link data for answers that is used for related answers. Uses the following nested pairs.	pair	CGU
link_item	A link between answers. Uses the following nested pairs.	pair	CG
access_time	The time a link was created.	time	CU

Table 28: Answer Pairs (Continued)

Name	Use	Type	Visibility
action	The action to take on the link. This field must be set to 1 to create a link, 2 to update a link, and 3 to delete a link.	integer	CDU
from_a_id	The first linked answer viewed.	integer	CG
static_strength	The static strength of the link.	integer	CG
strength	The relative relatedness of the linked answers.	integer	CG
to_a_id	The second answer viewed.	integer	CG
m_id	The meta-answer the answer is associated with.	integer	CDG
next_notify	The date a notification will be sent for the answer.	time	CGU
notes	The notes field of the answer.	string	CGU
notif_type	The type of notification.	string	CGU
publish_on	The date the answer will be published on.	time	CGU
rule_ctx	Escalation and rule state information associated with the answer. Uses the following pairs.	pair	G
escldate	The date and time the answer was escalated.	time	G
escllevel	The level that the answer has been escalated to through the rules engine.	integer	G
state	The rule state the answer is currently in.	integer	G
solution	The solution of the answer.	string	CGU
solved_count	The relevancy ranking of this answer.	integer	CGU
source_upd	The source of the answer. Uses the following nested pairs.	pair	CGU

Table 28: Answer Pairs (Continued)

Name	Use	Type	Visibility
lvl_id1	The level-one source of the answer. Refer to Appendix B, "Source Codes," on page 177.	integer	CGU
lvl_id2	The level-two source of the answer. Refer to Appendix B, "Source Codes," on page 177.	integer	CGU
static_solved	The fixed relevancy ranking of this answer (100 is fix at top, 50 is fix at middle, 0 is fix at bottom).	integer	CGU
status	The status of the answer. Uses the following nested pairs.	pair	CGU
id	The ID number of the status of the answer.	integer	CGU
type	The status type the answer is assigned to.	integer	CGU
sub_tbl	Used in the ans_get function to identify the table to retrieve notes from. Uses the following nested pair.	pair	G
tbl_id	Specifies the table to get notes from. This pair should always contain a value of 164 (the ID of the *notes* table).	integer	G
summary	The title of the answer.	string	CGU
type	The type of answer. • 1—HTML • 2—URL • 3—File Attachment	integer	CGU
url	The answer URL, if the answer type is URL.	string	CGU
wf_flag	Determines whether business rules run when the answer is created, updated, or deleted. • 0—Business rules do not run • 1—Business rules run	integer	CU

Contact API

The pairs described in the following table are available to use in contact functions.

Table 29: Contact Pairs

Name	Use	Type	Visibility
acquired	The time the first opportunity associated with the contact was closed.	time	G
addr	The address of the contact. Uses the following nested pairs.	pair	CGU
city	The name of the city in the contact's address information.	string	CGU
country_id	The ID number of the country in the contact's address information.	integer	CGU
postal_code	The postal or zip code in the contact's address.	string	CGU
prov_id	The ID number of the province or state in the contact's address information.	integer	CGU
street	The contact's street address.	string	CGU
alt_name	The alternative name of the contact. Uses the following nested pairs.	pair	CGU
first	The alternative first name of the contact.	string	CGU
last	The alternative last name of the contact.	string	CGU
banner	The flag information associated with the contact.	pair	CGU
acct_id	The ID number of the staff account that most recently updated the flag.	integer	G
flag	The importance of the flag. • 1—Low • 2—Medium • 3—High	integer	CGU
txt	The flag text.	string	CGU

Table 29: Contact Pairs (Continued)

Name	Use	Type	Visibility
upd	The time the flag text was updated.	time	G
cat	Defines the default category for the contact's searching. Uses nested pairs lvl_id1 through lvl_id6.	pair	CGU
lvl_id<1-6>	The pair data of the default category for the contact's searching.	integer	CGU
c_id	The ID number of the contact. In the mailing_send_to_contact function, this is the ID number of the contact the mailing will be sent to.	integer	CDGU
contact_list_ids	The contact lists that the contact is associated with. Uses the following nested pair.	pair	CGU
int_item<#>	The ID number of the contact list that the contact is associated with. The first contact list pair should be int_item1, the second should be int_item2, and so on.	integer	CGU
ctype_id	The ID number of the contact type.	integer	CGU
custom_field	A custom field associated with the contact. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	CGU
disabled	The disabled status of the contact. • 0—enabled • 1—disabled	integer	CGU
ee_flag	Determines whether external events run when a contact is created, updated, or deleted. • 0—External events do not run • 1—External events run	integer	CDU

Table 29: Contact Pairs (Continued)

Name	Use	Type	Visibility
email	The email address and security information associated with the contact. Uses the following nested pairs.	pair	CGU
addr	The email address associated with the contact.	string	CGU
cert	The S/MIME account certificate associated with the contact.	string	CGU
email_alt1	The first alternate email address and security information associated with the contact. Uses the following nested pairs.	pair	CGU
addr	The first alternate email address associated with the contact.	string	CGU
cert	The S/MIME account certificate associated with the contact.	string	CGU
email_alt2	The second alternate email address and security information associated with the contact. Uses the following nested pairs.	pair	CGU
addr	The second alternate email address associated with the contact.	string	CGU
cert	The S/MIME account certificate associated with the contact.	string	CGU
email_invalid	The email invalid status of the contact's primary email address.	integer	CGU
flow_id	Used in the mailing_send_to_contact function to define the campaign flow ID.	integer	CU
lines_per_page	The default number of lines per page shown for a contact.	integer	CGU
login	The contact login name.	string	CGU
ma_alt_org_name	The alternate Marketing organization name associated with the contact.	string	CGU

Table 29: Contact Pairs (Continued)

Name	Use	Type	Visibility
ma_mail_type	The Marketing mail type associated with the contact.	integer	CGU
ma_opt_in	The Marketing opt-in flag associated with the contact.	integer	CGU
ma_org_name	The Marketing organization name associated with the contact.	string	CGU
mailing_id	Used in the mailing_send_to_contact function to indicate the ID number of the mailing or survey to send.	integer	CU
name	The name of the contact. Uses the following nested pairs.	pair	CGU
first	The first name of the contact.	string	CGU
last	The last name of the contact.	string	CGU
note	Used in contact_create and contact_update to add note entries to contact records.	pair	CGU
note_item<#>	A note entry. Uses the following pairs. The first note entry should be named note_item1, the second should be note_item2, and so on.	pair	CGU
action	The action for the note. This field must be set to 1 to create a note, 2 to update a note, and 3 to delete a note.	integer	CDGU
channel	The ID number of the channel the note was created from. Refer to Table 15 on page 83.	integer	CGU
created	The time the note was created.	time	CGU
seq	The sequence of the note.	integer	CGU
text	The text of the note.	string	CGU
updated	The time the note was updated.	time	CGU
updated_by	The ID number of the staff account that the note is associated with.	integer	CG

Table 29: Contact Pairs (Continued)

Name	Use	Type	Visibility
org_id	The ID number of the organization associated with the contact.	integer	CGU
password	The contact's password.	string	CGU
ph_asst	The phone number of the contact's assistant.	string	CGU
ph_asst_raw	The contact's assistant phone number, without formatting (spaces or punctuation).	string	CGU
ph_fax	The contact's fax number.	string	CGU
ph_home	The contact's home phone number.	string	CGU
ph_home_raw	The contact's home phone number, without formatting (spaces or punctuation).	string	CGU
ph_mobile	The contact's mobile phone number.	string	CGU
ph_mobile_raw	The contact's mobile phone number, without formatting (spaces or punctuation).	string	CGU
ph_office	The contact's office phone number.	string	CGU
ph_office_raw	The contact's office phone number, without formatting (spaces or punctuation).	string	CGU
prod	The default product for the contact's searching. Uses the nested pairs lvl_id1 through lvl_id6.	pair	CGU
lvl_id<1-6>	The pair data of the default product for the contact's searching.	integer	CGU
prodcat_notif	The product and category notifications that the contact has subscribed to. Uses the following nested pairs.	pair	CGU
prodcat_notif_item	A notification. Uses the following nested pairs.	pair	CGU

Table 29: Contact Pairs (Continued)

Name	Use	Type	Visibility
hm	The hierarchy of the product or category. Uses the following nested pairs.	pair	CGU
lvl_id<#>	The hierarchy of the product or category.	integer	CGU
start_time	The time the subscription was created.	time	CGU
rule_state	The rule state the contact is currently in.	integer	G
sales_acct_id	The account ID of the sales rep assigned to the contact.	integer	CGU
scheduled	Used in the mailing_send_to_contact function to specify the time the mailing or survey should be sent.	time	CU
search_text	The default search text for the contact's searching.	string	CGU
search_type	The code of the default search type for the contact's searching.	integer	CGU
slai	The SLA instance associated with the contact. Refer to "Creating and deleting SLA instances" on page 85.	pair	CGU
sn_c_id	The Salesnet contact ID.	integer	CGU
source_upd	The creation source of the contact.	pair	CGU
lvl_id1	The level-one source of the contact. Refer to Appendix B, "Source Codes," on page 177.	integer	CGU
lvl_id2	The level-two source of the contact. Refer to Appendix B, "Source Codes," on page 177.	integer	CGU
state	The state of the contact. Uses the following nested pairs.	pair	CGU
css	The Service state of the contact.	integer	CGU
ma	The Marketing state of the contact.	integer	CGU

Table 29: Contact Pairs (Continued)

Name	Use	Type	Visibility
sa	The Sales state of the contact.	integer	CGU
sub_tbl	Used in the contact_get function to identify the table to retrieve notes from. Uses the following nested pair.	pair	G
tbl_id	Specifies the table to get notes from. This pair should always contain a value of 164 (the ID of the *notes* table).	integer	G
survey_opt_in	The Feedback opt-in flag associated with the contact.	integer	CGU
title	The contact's title.	string	CGU
trigger	Used in the mailing_send_to_contact function to indicate the incident or opportunity associated with the mailing or survey. This information is used for reporting purposes. Uses the following nested pairs	pair	CU
id	The ID number of the incident or opportunity that caused the mailing to be sent.	integer	CU
tbl	The ID number of the database table the record is associated with. • 1—Incidents • 87—Opportunities	integer	CU
updated	The time the contact was last updated.	time	G
updated_by	The staff member the contact was last updated by.	integer	CU
wf_flag	Determines whether business rules run when the contact is created, updated, or deleted. • 0—Business rules do not run • 1—Business rules run	integer	CU

Custom field API

The pairs described in the following table are available to use in custom field functions. Refer to "Setting custom fields" on page 80.

Table 30: Custom Field Pairs

Name	Use	Type	Visibility
custom_field	A custom field pair array using the following nested pairs. Refer to "Setting custom fields" on page 80.	pair	CGU
cf_item	A custom field item pair array using the following nested pairs.	pair	CGU
cf_id	The code number of a custom field. Refer to "Using cf_id pairs" on page 80, and "Finding code numbers" on page 89.	integer	CGU
data_type	The data_type of the custom field. Refer to "Using data_type pairs" on page 80.	integer	G
val_int	The integer value of the custom field. Refer to "Using value pairs" on page 81.	integer	CGU
val_str	The string value of the custom field. Refer to "Using value pairs" on page 81.	string	CGU
val_time	The time value of the custom field. Refer to "Using value pairs" on page 81.	time	CGU

Flow API

The pairs described in the following table are available to use in the flow_execute function.

Table 31: Flow Pairs

Name	Use	Type	Visibility
c_id	The ID number of the contact associated with the flow.	integer	N/A
shortcut	The Shortcut ID field that is entered in the window for an Entry Point action in a flow.	string	N/A

Table 31: Flow Pairs (Continued)

Name	Use	Type	Visibility
flow_id	The ID number of the flow to be executed.	integer	N/A

Hierarchical menu API

The pairs described in the following table are available to use in hierarchical menu functions.

Table 32: Hierarchical Menu Pairs

Name	Use	Type	Visibility
desc	The hierarchical menu item's description information. Uses the following nested pairs.	pair	CGU
lbl_item<#>	The descriptions and languages of the hierarchical menu item. Uses the following nested pairs. The first label item should be lbl_item1, the second should be lbl_item2, and so on.	pair	CGU
label	The description text.	string	CGU
lang_id	The ID number of the language the label is written in.	integer	CGU
id	The ID number of the hierarchical menu item.	integer	DGU
label	The name(s) of the hierarchical menu items. Uses the following nested pairs.	pair	CGU
lbl_item	The names and languages of the hierarchical menu item. Uses the following nested pairs. The first label item should be lbl_item1, the second lbl_item2, and so on.	pair	CGU
label	The name text.	string	CGU
lang_id	The ID number of the language the name is written in.	integer	CGU

Table 32: Hierarchical Menu Pairs (Continued)

Name	Use	Type	Visibility
new_lvl	In the css_category_move, css_disposition_ move, and css_product_move functions, the new level of the hierarchical menu item being moved.	integer	N/A
new_seq	In the css_category_move, css_disposition_ move, and css_product_move functions, the new sequence of the hierarchical menu being moved.	integer	N/A
np_lvl_id	In the css_category_move, css_disposition_ move, and css_product_move functions, the level IDs of new parent. Uses nested pairs lvl_id1 through lvl_id6.	pair	N/A
lvl_id<1-6>	The pair data specifying the level IDs of the new parent menu.	integer	N/A
old_lvl	In the css_category_move, css_disposition_ move, and css_product_move functions, the old level of the hierarchical menu item being moved.	integer	N/A
old_parent	In the css_category_move, css_disposition_ move, and css_product_move functions, the ID of the old parent of the hierarchical menu item being moved.	integer	N/A
old_seq	In the css_category_move, css_disposition_ move, and css_product_move functions, the old sequence of the hierarchical menu item being moved.	integer	N/A
parent	The hierarchy of the parent menu item.	pair	CGU
lvl_id<1-6>	The pair data specifying the level IDs of the parent menu.	integer	CGU
seq	The position of the hierarchical menu item within the list of hierarchical menu items.	integer	CGU

Table 32: Hierarchical Menu Pairs (Continued)

Name	Use	Type	Visibility
vis	The visibility of the hierarchical menu item. Uses the following nested pairs.	pair	CGU
vis_item	The visibility settings for the hierarchical menu item. Uses the following nested pairs.	pair	CGU
admin	The visibility of the hierarchical menu item on the administration interface.	integer	CGU
enduser	The visibility of the hierarchical menu item on the end-user interface.	integer	CGU
intf_id	The ID number of the interface that the hierarchical menu item is associated with.	integer	CGU

Incident API

The pairs described in the following table are available to use in incident functions.

Table 33: Incident Pairs

Name	Use	Type	Visibility
assigned	The staff member the incident is assigned to. Uses the following nested pairs.	pair	CGU
acct_id	The ID number of the staff member the incident is assigned to.	integer	CGU
group_id	The ID number of the staff group the incident is assigned to.	integer	CGU
banner	The flag information associated with the incident.	pair	CGU
acct_id	The ID number of the staff account that most recently updated the flag.	integer	G

Table 33: Incident Pairs (Continued)

Name	Use	Type	Visibility
flag	The importance of the flag. • 1—Low • 2—Medium • 3—High	integer	CGU
txt	The flag text.	string	CGU
upd	The time the flag text was updated.	time	G
call_id	The ID number of the call the incident was created from.	integer	CGU
cat	Defines the category the incident is associated with. Uses nested pairs lvl_id1 through lvl_id6.	pair	CGU
lvl_id<1-6>	The pair data of the category the incident is associated with.	integer	CGU
closed	The time the incident was closed.	time	G
contact	The contact(s) associated with the incident. Uses the following nested pairs.	pair	CGU
ic_item<#>	A contact associated with the incident. Uses the following nested pairs. The first contact should be ic_item1, the second should be ic_item2, and so on.	pair	CGU
c_id	The ID number of the contact associated with the incident.	integer	CGU
prmry	Determines whether the contact is the primary contact for the incident. • 0—Not the primary contact • 1—Primary contact	integer	CGU
created	The time the incident was created.	time	G
created_by	The ID number of the incident creator.	integer	CG

Table 33: Incident Pairs (Continued)

Name	Use	Type	Visibility
custom_field	A custom field associated with the incident. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	CGU
disp	The disposition assigned to the incident. Uses nested pairs lvl_id1 through lvl_id6.	pair	CGU
lvl_id<1-6>	The pair data specifying the disposition of the incident.	integer	CGU
ee_flag	Determines whether external events run when an incident is created, updated, or deleted. • 0—External events do not run • 1—External events run	integer	C
ei_cust	The emotive index of the contact associated with the incident.	integer	G
ei_staff	The emotive index of the staff assigned to the incident.	integer	G
i_id	The ID number of the incident.	integer	CDGU
initial_soln	The date and time the incident was responded to ending with a status change to a type other than unresolved.	time	G
interface_id	The ID number of the interface associated with the incident.	integer	CGU
lang_id	The ID number of the language associated with the incident.	integer	CGU
last_resp	The date and time the incident was last responded to.	time	G
last_survey_score	The score of the most recent survey completed for the incident.	integer	G

Table 33: Incident Pairs (Continued)

Name	Use	Type	Visibility
mailbox_id	The ID number of the mailbox the incident was created from.	integer	G
mailing_id	The ID number of the marketing mailing.	integer	G
org_id	The ID number of the organization associated with the incident.	integer	CGU
prod	The product hierarchy the incident is associated with. Uses the nested pairs lvl_id1 through lvl_id6.	pair	CGU
lvl_id<1-6>	The pair data of the product hierarchy the incident is associated with.	integer	CGU
queue_id	The ID number of the queue the incident is assigned to.	integer	CGU
ref_no	The reference number of the incident.	string	CG
rel_due	The relative due date to be met to meet the SLA. If SLAs have not been implemented, this would apply to the default response requirements.	time	G
resp_sav	An uncommitted (not sent) response thread.	string	CGU
response	Indicates what type of response to send. If this pair is not included, a response is not sent.	pair	CU
type	The ID number of the response type to send. • 1—Incident closed message • 2—Incident receipt message • 3—Incident response message	integer	CU
rnl_queue_id	The ID number of the RightNow Live queue the incident is assigned to.	integer	G

Table 33: Incident Pairs (Continued)

Name	Use	Type	Visibility
rr_id	The ID number of the response requirements associated with the incident.	integer	G
rule_ctx	Escalation and rule state information associated with the incident. Uses the following pairs.	pair	G
escldate	The date and time the incident was escalated.	time	G
esclevel	The level that the incident has been escalated to through the rules engine.	integer	G
state	The rule state the incident is currently in.	integer	G
severity_id	The ID number of the severity level assigned to the incident.	integer.	CGU
sla_resp_delta	The number of minutes it took to respond to the incident past the SLA's response requirement.	integer	G
sla_rsln_delta	The number of minutes it took to resolve the incident past the SLA's resolution requirement.	integer	G
slai_id	The ID number of the SLA instance the incident is assigned to.	integer	CGU
source_upd	The creation source of the incident. Uses the following nested pairs.	pair	CU
lvl_id1	The level-one source of the incident. Refer to Appendix B, "Source Codes," on page 177.	integer	CU
lvl_id2	The level-two source of the incident. Refer to Appendix B, "Source Codes," on page 177.	integer	CU
status	The status of the incident. Uses the following nested pairs.	pair	CGU
id	The ID number of the status.	integer	CGU

Table 33: Incident Pairs (Continued)

Name	Use	Type	Visibility
type	The ID number of the status type.	integer	CGU
subject	The title of the incident.	string	CGU
sub_tbl	Used in the incident_get function to identify the table to retrieve threads from. Uses the following nested pair.	pair	G
tbl_id	Specifies the table to get threads from. This pair should always contain a value of 18 (the ID of the *threads* table).	integer	G
thread	The incident threads (response, note, customer). Uses the following nested pairs. Refer to "Adding thread entries" on page 82.	pair	CGU
thread_entry<#>	A entry within the incident thread. Uses the following nested pairs. The first thread entry should be named thread_entry1, the second should be thread_entry2, and so on.	pair	CGU
acct_id	The ID number of the staff account associated with the thread.	integer	CG
c_id	The ID number of the contact associated with the thread.	integer	CGU
channel	The ID number of the channel associated with the thread. Refer to Table 15 on page 83.	integer	CGU
ei	The emotive index rating of the thread.	integer	G
entered	The time the thread was created.	time	G
entry_type	The ID number of the incident thread type. Refer to Table 14 on page 83.	integer	CGU
note	The text contained in the thread entry.	string	CGU
seq	The sequence of the thread entry.	integer	CGU

Table 33: Incident Pairs (Continued)

Name	Use	Type	Visibility
time_billed	The time billed for the incident. Uses the following nested pairs.	pair	CGU
tb_item<#>	A time billed entry. Uses the following nested pairs. The first tb_item pair should be named tb_item1, the second should be tb_item2, and so on.	pair	CGU
acct_id	The ID number of the staff member billing the time.	integer	CGU
bill_date	The time the time-billed item was created.	time	CGU
minutes	The number of minutes billed.	integer	CGU
notes	Notes associated with the time-billed entry.	string	CGU
bt_id	The ID number of the time-billed activity associated with the time-billed entry.	integer	CGU
updated	The time the incident was last updated.	time	G
updated_by	The ID number of the staff member updating the incident.	integer	CU
updated_by_c_id	The ID number of the contact updating the incident.	integer	U
use_smime	Indicates whether S/MIME is used for the incident. • 0—S/MIME not used • 1—S/MIME used	integer	G
wf_flag	Determines whether business rules run when the incident is created, updated, or deleted. • 0—Business rules do not run • 1—Business rules run	integer	CU

Meta-Answer API

The pairs described in the following table are available to use in meta-answer functions.

Table 34: Meta-Answer Pairs

Name	Use	Type	Visibility
categories	The categories associated with the meta-answer.	pair	CU
hier_item	A category. Uses the following nested pair.	pair	CU
hm	The hierarchy of the category. Uses the following nested pairs.	pair	CU
lvl_id<1-6>	The pair data of the category hierarchy the meta-answer is associated with.	integer	CU
m_id	The ID number of the meta-answer.	integer	CDU
notes	The notes field of the meta-answer.	string	CU
orig_ref_no	The original reference number of an incident that has been converted to an answer.	string	CU
products	The products associated with the meta-answer.	pair	CU
hier_item	A product. Uses the following nested pair.	pair	CU
hm	The hierarchy of the product. Uses the following nested pairs.	pair	CU
lvl_id<1-6>	The pair data of the product hierarchy the meta-answer is associated with.	integer	CU
source_upd	The source of the meta-answer. Uses the following nested pairs.	integer	CU
lvl_id1	The level-one source of the meta_answer. Refer to Appendix B, “Source Codes,” on page 177.	integer	CU
lvl_id2	The level-two source of the meta-answer. Refer to Appendix B, “Source Codes,” on page 177.	integer	CU

Opportunity API

The pairs described in the following table are available to use in opportunity functions.

Table 35: Opportunity Pairs

Name	Use	Type	Visibility
assigned	The sales representative assigned to the opportunity. Uses the following nested pairs.	pair	CGU
chain	The management hierarchy of the staff account assigned to the opportunity. Uses the nested pairs lvl_id1 through lvl_id12.	pair	CGU
lvl_id<1-12>	The pair data defining the management hierarchy of the sales representative assigned to the opportunity.	integer	CGU
id	The ID number of the sales representative assigned to the opportunity.	integer	CGU
banner	The flag information associated with the opportunity.	pair	CGU
acct_id	The ID number of the staff account that most recently updated the flag.	integer	G
flag	The importance of the flag. • 1—Low • 2—Medium • 3—High	integer	CGU
txt	The flag text.	string	CGU
upd	The time the flag text was updated.	time	G
call_id	The ID number of the call the opportunity was created from.	integer	CGU
closed	The date and time the opportunity was closed.	time	CGU
closed_value	The closed-value information for the opportunity. Uses the following nested pairs.	pair	CGU

Table 35: Opportunity Pairs (Continued)

Name	Use	Type	Visibility
curr_id	The ID number of the closed-value currency type.	integer	CGU
rate_id	The ID number of the closed-value exchange rate.	integer	CGU
val	The closed value of the opportunity.	string	CGU
competitor	Competitors associated with the opportunity. Uses the following nested pairs.	pair	CGU
comp_item	A competitor associated with the opportunity. Uses the following nested pairs.	pair	CGU
competitor_id	The ID number of the competitor.	integer	CGU
prmry	Defines which competitor is the primary competitor for the opportunity. One competitor must be specified as the primary. A value of 1 identifies the primary competitor.	integer	CGU
contact	Contacts associated with the opportunity. Uses the following nested pairs.	pair	CGU
oc_item<#>	A contact associated with the opportunity. Uses the following nested pairs. The first contact should be named oc_item1, the second should be oc_item2, and so on.	pair	CGU
c_id	The ID number of the contact.	integer	CGU
cr_id	The ID number of the contact role of the contact.	integer	CGU
oc_primary	Defines which contact is the primary contact for the opportunity. One contact must be specified as the primary contact. A value of 1 identifies the primary contact.	integer	CGU
cos	The cost of sale of the opportunity. Uses the following nested pairs.	pair	CGU

Table 35: Opportunity Pairs (Continued)

Name	Use	Type	Visibility
curr_id	The ID number of the cost-of-sale currency type.	integer	CGU
rate_id	The ID number of the cost-of-sale exchange rate.	integer	CGU
val	The cost of sale of the opportunity.	string	CGU
created_by	The ID number of the staff member who created the opportunity.	integer	CG
custom_field	A custom field associated with the opportunity. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	CGU
ee_flag	Determines whether external events run when an opportunity is created, updated, or deleted. • 0—External events do not run • 1—External events run	integer	CDU
flow_id	The ID number of the flow the opportunity is associated with.	integer	CGU
forecast_close	The date the opportunity is forecasted to close.	time	CGU
initial_contact	The date the sales representative initially made contact with the organization.	time	CGU
interface_id	The ID number of the interface the opportunity is associated with.	integer	CGU
last_survey_score	The last survey score for the opportunity.	integer	G
lead_rej_desc	The comments entered when the lead was rejected.	string	CGU
lead_rej_dttm	The time the lead was rejected.	time	CGU
lead_rej_id	The ID of the lead rejection reason.	integer	CGU

Table 35: Opportunity Pairs (Continued)

Name	Use	Type	Visibility
lost	The time that the opportunity was lost.	time	CGU
mgr_commit	The committed status of the manager-forecasted value. • 0—Not committed • 1—Committed	integer	CGU
mgr_value	The manager-forecasted value of the opportunity. Uses the following nested pairs.	pair	CGU
curr_id	The ID number of the currency of the manager-forecasted value.	integer	CGU
rate_id	The ID number of the exchange rate of the manager-forecasted value.	integer	CGU
val	The manager-forecasted value of the opportunity.	string	CGU
name	The name of the opportunity.	string	CGU
note	Used in opp_create and opp_update to add note entries to opportunities.	pair	CU
note_item<#>	A note entry. Uses the following pairs. The first note entry should be named note_item1, the second should be note_item2, and so on.	pair	CGU
action	The action for the note. The action for the note. This field must be set to 1 to create a note, 2 to update a note, and 3 to delete a note.	integer	CGU
channel	The ID number of the channel the note was created from. Refer to Table 15 on page 83.	integer	CDGU
created	The time the note was created.	time	CGU
created_by	The ID number of the staff account that the note is associated with.	integer	CGU
seq	The sequence of the note.	integer	CGU

Table 35: Opportunity Pairs (Continued)

Name	Use	Type	Visibility
text	The text of the note.	string	CGU
updated	The time the note was updated.	time	CGU
updated_by	The ID number of the staff account that the note is associated with.	integer	CG
op_id	The ID number of the opportunity.	integer	CDGU
org_id	The ID number of the organization associated with the opportunity.	integer	CGU
qt	The quotes associated with the opportunity. Refer to "Quote API" on page 164 for a list of the nested pairs used with this pair.	pair	GU
recall	The recall date associated with an opportunity.	time	CGU
rep_commit	The committed status of the sales representative-forecasted value. • 0—Not committed • 1—Committed	integer	CGU
rep_value	The sales-representative-forecasted value of the opportunity. Uses the following nested pairs.	pair	CGU
curr_id	The ID number of the currency of the sales-representative-forecasted value.	integer	CGU
rate_id	The ID number of the exchange rate of the sales-representative-forecasted value.	integer	CGU
val	The sales-representative-forecasted value of the opportunity.	string	CGU
ret_value	The return value of the opportunity. Uses the following nested pairs.	pair	CGU
curr_id	The ID number of the currency of the return value.	integer	CGU

Table 35: Opportunity Pairs (Continued)

Name	Use	Type	Visibility
rate_id	The ID number of the exchange rate of the return value.	integer	CGU
val	The return value of the opportunity.	string	CGU
rule_ctx	Escalation and rule state information associated with the opportunity. Uses the following pairs.	pair	G
escldate	The date and time the opportunity was escalated.	time	G
escllevel	The level that the opportunity has been escalated to through the rules engine.	integer	G
state	The rule state the opportunity is currently in.	integer	G
source_upd	The creation source of the opportunity.	pair	CG
lvl_id1	The level-one source of the opportunity. Refer to Appendix B, "Source Codes," on page 177.	integer	CG
lvl_id2	The level-two source of the opportunity. Refer to Appendix B, "Source Codes," on page 177.	integer	CG
stage	The stage the opportunity is in. Uses the following nested pairs.	pair	CGU
stage_id	The ID number of the stage the opportunity is in.	integer	CGU
strategy_id	The ID number of the strategy the opportunity is associated with.	integer	CGU
status	The status of the opportunity. Uses the following nested pairs.	pair	CGU
id	The ID number of the status of the opportunity.	integer	CGU

Table 35: Opportunity Pairs (Continued)

Name	Use	Type	Visibility
type	The ID number of the status type of the opportunity.	integer	CGU
summary	The summary of the opportunity.	string	CGU
sub_tbl	Used in the opp_get function to identify the table to retrieve notes from. Uses the following nested pair.	pair	G
tbl_id	Specifies the table to get notes from. This pair should always contain a value of 164 (the ID of the *notes* table).	integer	G
terr	The territory associated with the opportunity. Uses the following nested pairs.	pair	CGU
chain	The territorial hierarchy associated with the opportunity. Uses the following nested pairs.	pair	CGU
id	The ID number of the territory associated with the opportunity.	integer	CGU
lvl_id<1-12>	The pair data defining the territorial hierarchy associated with the opportunity.	integer	CGU
updated	The date and time the opportunity was last updated.	time	G
updated_by	The ID number of the staff member who last updated the opportunity.	integer	CGU
wf_flag	Determines whether business rules run when the opportunity is created, updated, or deleted. • 0—Business rules do not run • 1—Business rules run	integer	CU
win_loss_desc	The win/loss description for the opportunity.	string	CGU
win_loss_id	The ID number of the win/loss reason associated with the opportunity.	integer	CGU

Organization API

The pairs described in the following table are available to use in organization functions.

Table 36: Organization Pairs

Name	Use	Type	Visibility
acquired	The time the first opportunity associated with the organization was closed.	time	G
alt_name	The alternate name for the organization.	string	CGU
banner	The flag information associated with the organization.	pair	CGU
acct_id	The ID number of the staff account that most recently updated the flag.	integer	G
flag	The importance of the flag. • 1—Low • 2—Medium • 3—High	integer	CGU
txt	The flag text.	string	CGU
upd	The time the flag text was updated.	time	G
created	The time the organization was created.	time	G
custom_field	A custom field associated with the organization. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	CGU
ee_flag	Determines whether external events run when an opportunity is created, updated, or deleted. • 0—External events do not run • 1—External events run	integer	CDU
industry_id	The ID number of the industry the organization is associated with.	integer	CGU

Table 36: Organization Pairs (Continued)

Name	Use	Type	Visibility
login	The organization login name.	string	CGU
name	The name of the organization.	string	CGU
note	Used in org_create and org_update to add note entries to organizations.	pair	CGU
note_item<#>	A note entry. Uses the following pairs. The first note entry should be named note_item1, the second should be note_item2, and so on.	pair	CGU
action	The action for the note. • 1—Create a note • 2—Update a note • 3—Delete a note	integer	CGU
channel	The ID number of the channel the note was created from. Refer to Table 15 on page 83.	integer	CG
created	The time the note was created.	time	G
created_by	The ID number of the staff account that the note is associated with.	integer	G
seq	The sequence of the note.	integer	CGU
text	The text of the note.	string	CGU
updated	The time the note was updated.	time	G
updated_by	The ID number of the staff account that the note is associated with.	integer	G
num_employees	The number of employees the organization has.	integer	CGU
oaddr	The addresses of the organization. Uses the following nested pairs.	pair	CGU

Table 36: Organization Pairs (Continued)

Name	Use	Type	Visibility
oaddr_item<#>	An organization address, including address type. Uses the following nested pairs. The first oaddr_item pair should be named oaddr_item1, the second should be oaddr_item2, and so on.	pair	CGU
addr	The organization address. Uses the following nested pairs.		
city	The city associated with the address.	string	CGU
country_id	The ID number of the country associated with the address.	integer	CGU
postal_code	The postal or zip code associated with address.	string	CGU
prov_id	The ID number of the state or province associated with the address.	integer	CGU
street	The street address.	string	CGU
oat_id	The type of address. • 1—Billing • 2—Shipping	integer	CGU
org_id	The ID number of the organization.	integer	CDGU
parent	The ID of the higher-level hierarchical menu item that the lower-level hierarchical menu item is associated with.	integer	CGU
lvl_id<1-6>	The pair data specifying the level IDs of the parent menu.	integer	CGU
password	The password of the organization.	string	CGU
rule_state	The rule state the organization is currently in.	integer	G
sales_acct_id	The ID number of the sales representative who is associated with the organization.	integer	CGU

Table 36: Organization Pairs (Continued)

Name	Use	Type	Visibility
slai	The SLA instance associated with the organization. Refer to "Creating and deleting SLA instances" on page 85.	pair	CGU
sn_org_id	The Salesnet organization ID number.	integer	CGU
source_upd	The creation source of the organization.	pair	CGU
lvl_id1	The level-one source of the organization. Refer to Appendix B, "Source Codes," on page 177.	integer	CGU
lvl_id2	The level-two source of the organization. Refer to Appendix B, "Source Codes," on page 177.	integer	CGU
state	The state of the organization. Uses the following nested pairs.	pair	CGU
css	The Service state of the organization.	integer	CGU
ma	The Marketing state of the organization.	integer	CGU
sa	The Sales state of the organization.	integer	CGU
sub_tbl	Used in the org_get function to identify the table to retrieve notes from. Uses the following nested pair.	pair	G
tbl_id	Specifies the table to get notes from. This pair should always contain a value of 164 (the ID of the *notes* table).	integer	G
tot_rev	The total revenue generated by the organization. Uses the following nested pairs.	pair	CGU
curr_id	The ID number of the currency of the total revenue.	integer	CGU
rate_id	The ID number of the exchange rate of the total revenue.	integer	CGU
val	The total revenue of the organization.	string	CGU

Table 36: Organization Pairs (Continued)

Name	Use	Type	Visibility
updated	The time the organization was last updated.	integer	G
updated_by	The staff member who last updated the organization.	integer	CG
wf_flag	Determines whether business rules run when the organization is created, updated, or deleted. • 0—Business rules do not run • 1—Business rules run	integer	CU

Purchased product API

The pairs described in the following table are available to use in the pur_prod_create function.

Table 37: Purchased Product Pairs

Name	Use	Type	Visibility
pp_item<#>	A purchased product. Uses the following nested pairs. The first purchased product pair should be named pp_item1, the second should be pp_item2, and so on.	pair	C
c_id	The ID number of the contact that purchased the product.	integer	C
campaign_id	The ID number of the campaign associated with the purchased product.	integer	C
custom_field	A custom field associated with the quote. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	C
finalized_by	The ID number of the staff member that finalized the sale.	integer	C
license_end	The time the product license ends.	integer	C

Table 37: Purchased Product Pairs (Continued)

Name	Use	Type	Visibility
license_start	The time the product license begins.	integer	C
mailing_id	The ID number of the mailing associated with the purchased product.	integer	C
notes	Notes associated with the purchased product.	string	C
oa_c_id	The Offer Advisor contact ID.	integer	C
op_id	The ID number of the opportunity the purchased product is associated with.	integer	C
org_id	The organization that purchased the product.	integer	C
price	The price of the purchased product. Uses the following nested pairs.	pair	C
curr_id	The ID number of the currency of the purchase price.	integer	C
rate_id	The ID number of the exchange rate of the purchase price.	integer	C
val	The purchase price.	string	C
purchase_date	The time the product was purchased.	time	C
quote_id	The ID number of the quote the purchased product is associated with.	integer	C
serial_number	The serial number of the purchased product.	string	C

Quote API

The pairs described in the following table are available to use in opportunity functions.

Table 38: Quote Pairs

Name	Use	Type	Visibility
qt	Quotes associated with the opportunity. Uses the following nested pairs.	pair	GU
qt_item	A quote. The first qt_item pair should be named qt_item1, the second should be qt_item2. Uses the following nested pairs.	pair	GU
action	The action for the quote item. This field must be set to 1 to create a quote item, 2 to update a quote item, and 3 to delete a a quote item.	integer	GU
adj_total	The adjusted total of the quote. Uses the following nested pairs.	pair	GU
curr_id	The ID number of the currency of the adjusted total.	integer	GU
rate_id	The ID number of the exchange rate of the adjusted total.	integer	GU
val	The adjusted total value.	string	GU
created	The time the quote was created.	time	G
created_by	The ID number of the staff member who created the quote.	integer	G
custom_field	A custom field associated with the quote. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	GU
discount	The discount applied to the quote.	integer	GU
forecast	The forecast status of the quote. • 0—Forecast check box is cleared • 1—Forecast check box is selected	integer	GU

Table 38: Quote Pairs (Continued)

Name	Use	Type	Visibility
name	The name of the quote.	string	GU
notes	The notes associated with the quote.	string	GU
offer_end	The offer end date.	time	GU
offer_start	The offer start date.	time	GU
prod	The sales products contained in the quote. Uses the following nested pairs.	pair	GU
pq_item	A sales product associated with the quote. Uses the following nested pairs.	pair	GU
adjusted_desc	The adjusted description of the product associated with the quote.	string	GU
adjusted_id	The adjusted ID of the product associated with the quote.	string	GU
adjusted_name	The adjusted name of the product associated with the quote.	string	GU
adjusted_price	The adjusted price of the product associated with the quote. Uses the following nested pairs.	pair	GU
curr_id	The ID number of the currency of the adjusted price.	integer	GU
rate_id	The ID number of the exchange rate of the adjusted price.	integer	GU
val	The adjusted price.	string	GU
adjusted _total	The adjusted total for the product associated with the quote (adjusted price multiplied by quantity). Uses the following nested pairs.	pair	GU
curr_id	The ID number of the currency of the adjusted total.	integer	GU
rate_id	The ID number of the exchange rate of the adjusted total.	integer	GU

Table 38: Quote Pairs (Continued)

Name	Use	Type	Visibility
val	The adjusted total value.	string	GU
custom_field	A custom field associated with the sales product. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	GU
discount	The adjusted discount for the product.	integer	GU
notes	The notes associated with the edited sales product.	integer	GU
original_desc	The original sales product description, before it was edited.	integer	G
original_id	The original sales product ID, before it was edited.	string	G
original_name	The original sales product name, before it was edited.	string	G
original_price	The original sales product price, before it was edited. Uses the following nested pairs.	pair	G
curr_id	The ID number of the currency of the original price.	integer	GU
rate_id	The ID number of the exchange rate of the original price.	integer	GU
val	The adjusted original price.	string	GU
product_id	The ID number of the product.	integer	DGU
qty	The quantity of the sales product.	integer	GU
seq	The sequence of the sales product in the list of sales products associated with the quote.	integer	GU
quote_id	The ID number of the quote.	integer	GU
schedule_id	The ID number of the price schedule associated with the quote.	integer	GU

Table 38: Quote Pairs (Continued)

Name	Use	Type	Visibility
sent	The date and time the quote was sent.	time	GU
sent_to	The email address the quote was sent to.	string	GU
status	The current status of the quote.	integer	GU
tmpl_file_id	The ID number of the quote template used in the quote.	integer	GU
total	The total value of the quote. Uses the following nested pairs.	pair	GU
curr_id	The currency ID associated with the total value.	integer	GU
rate_id	The exchange rate ID associated with the total value.	integer	GU
val	The total value.	string	GU
updated	The time the quote was last updated.	time	GU
updated_by	The ID number of the staff member who last updated the quote.	integer	GU

Sales product API

The pairs described in the following table are available to use in sales product functions.

Table 39: Sales Product Pairs

Name	Use	Type	Visibility
cnt	The number of times the product has been offered by Offer Advisor.	integer	CGU
custom_field	A custom field associated with the sales product. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	CGU

Table 39: Sales Product Pairs (Continued)

Name	Use	Type	Visibility
desc	The description of the sales product. Uses the following nested pairs.	pair	CGU
lbl_item<#>	The descriptions and languages of the sales product. Uses the following nested pairs. The first label item should be lbl_item1, the second should be lbl_item2, and so on.	pair	CGU
label	The description text.	string	CGU
lang_id	The ID number of the language the label is written in.	integer	CGU
disabled	Indicates if the sales product is disabled.	integer	CGU
folder_id	The ID number of the folder the product is associated with.	integer	CDG
id	The ID number of the sales product.	integer	CGU
label	The name of the sales product.	pair	CGU
lbl_item<#>	The descriptions and languages of the sales product. Uses the following nested pairs. The first label item should be lbl_item1, the second should be lbl_item2, and so on.	pair	CGU
label	The description text.	string	CGU
lang_id	The ID number of the language the label is written in.	integer	CGU
oa_exclude	Indicates whether the product is excluded from being suggested by Offer Advisor. • 0—Included • 1—Excluded	integer	CGU
product_id	The ID number of the sales product.	integer	DGU
sched	Price schedules associated with the sales product. Uses the following nested pairs.	pair	CGU

Table 39: Sales Product Pairs (Continued)

Name	Use	Type	Visibility
sch_item<#>	A price schedule associated with the sales product. Uses the following nested pairs. The first schedule item should be sch_item1, the second should be sch_item2, and so on.	pair	CGU
notes	Notes associated with the product-to-schedule relationship.	string	CGU
price	The price of the sales product in the associated schedule. Uses the following nested pairs.	pair	CGU
curr_id	The ID number of the currency of the sales product's price.	integer	CGU
rate_id	The ID number of the exchange rate of the sales product's price.	integer	CGU
val	The sales product's price.	string	CGU
schedule_end	The schedule start date.	time	CGU
schedule_id	The ID number of the schedule associated with the sales product.	integer	CGU
schedule_start	The schedule end date.	time	CGU
seq	The sequence of the sales product in the folder or folder list.	integer	CDG
updated	The time the sales product was last updated.	time	G
vis	Visibility settings for the sales product. Uses the following nested pairs.	pair	CGU
vis_item<#>	Visibility settings for an interface. Uses the following pair. The first vis_item pair should be named vis_item1, the second should be vis_item2, and so on.	pair	CGU

Table 39: Sales Product Pairs (Continued)

Name	Use	Type	Visibility
admin	Indicates whether the sales product is visible on the interface. • 0—Not visible • 1—Visible	integer	CGU
intf_id	The ID number of the interface the visibility setting applies to.	integer	CGU
yes_cnt	The number of times the product has been accepted when offered by Offer Advisor.	integer	CGU

Search API

The pairs described in the following table are available to use in the search function.

Table 40: Search Pairs

Name	Use	Type	Visibility
search_args	The search argument. Uses the following nested pairs.	pair	N/A
search_field<#>	The search fields. Uses the following nested pairs. The first search field should be named search_field1, the second should be search_field2, and so on.	pair	N/A
name	The name of the run-time selectable filter being searched on.	string	N/A
compare_val	The value of the field being searched on.	string	N/A

SLA instance API

The pairs described in the following table are available to use in contact and organization functions.

Table 41: SLA Instance Pairs

Name	Use	Type	Visibility
slai	SLA instances. Uses the following nested pairs.	pair	CDGU
slai_item<#>	An SLA instance. Uses the following pairs. The first SLA instance should be named slai_item1, the second should be slai_item2, and so on.	pair	CGU
action	The action for the SLA instance. This field must be set to 1 to create an SLA instance, 2 to update an SLA instance, and 3 to delete an SLA instance.	integer	CDGU
activedate	The activation date of the SLA instance.	time	CGU
expiredate	The expiration date of the SLA instance.	time	CGU
inc_chat	The number of chat incidents remaining in the SLA instance.	integer	CGU
inc_csr	The number of CSR incidents remaining in the SLA instance.	integer	CGU
inc_email	The number of email incidents remaining in the SLA instance.	integer	CGU
inc_total	The total number of incidents remaining in the SLA instance.	integer	CGU
inc_web	The number of web incidents remaining in the SLA instance.	integer	CGU
sla_id	The ID number of the SLA that the SLA instance is associated with.	integer	CGU
sla_set	The shared ID number of the SLA if a modified version of an SLA is used.	integer	CGU

Table 41: SLA Instance Pairs (Continued)

Name	Use	Type	Visibility
slai_id	The ID number of the SLA instance.	integer	CDGU
state	The state of the SLA. • 1—Not ready • 2—Active • 3—Used up • 4—Disabled	integer	CGU

Task API

The pairs described in the following table are available to use in task functions.

Table 42: Task Pairs

Name	Use	Type	Visibility
a_id	The ID number of the answer the task is associated with.	integer	CGU
assgn_acct_id	The ID number of the staff member assigned to the task.	integer	CGU
c_id	The ID number of the contact associated with the task.	integer	CGU
campaign_id	The ID number of the campaign the task is associated with.	integer	CGU
completed	The time the task was completed.	time	CGU
created	The time the task was created.	time	G
created_by	The ID number of the staff member the task was created by.	integer	CG
custom_field	A custom field associated with the task. For information on the nested pairs this pair uses, refer to "Custom field API" on page 140.	pair	CGU

Table 42: Task Pairs (Continued)

Name	Use	Type	Visibility
doc_id	The ID number of the document the task is associated with.	integer	CGU
due_date	The date and time the task is due.	time	CGU
i_id	The ID number of the incident the task is associated with.	integer	CGU
inherit	A bitmask defining the type of data inherited from the parent task. • 1—Staff assignment • 2—Organization association • 4—Contact association	integer	CGU
mailing_id	The ID number of the mailing the task is associated with.	integer	CGU
name	The name of the task.	string	CGU
notes	The notes associated with the task.	string	CGU
op_id	The opportunity the task is associated with.	integer	CGU
org_id	The organization the task is associated with.	integer	CGU
pct_complete	The percentage of the task that has been completed.	integer	CGU
planned_completion	The planned completion date and time for the task.	time	CGU
priority	The priority level of the task. • 1—Low • 2—Normal • 3—High	integer	CGU
rule_ctx	Escalation and rule state information associated with the task. Uses the following pairs.	pair	G
escldate	The date and time the task was escalated.	time	G

Table 42: Task Pairs (Continued)

Name	Use	Type	Visibility
escllevel	The level that the task has been escalated to through the rules engine.	integer	G
state	The rule state the task is currently in.	integer	G
source_upd	The source of the task. Uses the following nested pairs.	pair	CGU
lvl_id1	The level-one source of the task. Refer to Appendix B, "Source Codes," on page 177.	integer	CGU
lvl_id2	The level-two source of the task. Refer to Appendix B, "Source Codes," on page 177.	integer	CGU
start_date	The time the task started.	time	CGU
status	The status of the task. Uses the following nested pairs.	pair	CGU
id	The ID number of the status of the task.	integer	CGU
type	The ID number of the status type of the task.	integer	CGU
survey_id	The ID number of the survey the task is associated with.	integer	CGU
tt_id	The ID number of the task template that the task is associated with.	integer	CGU
tbl	The table the task is associated with.	integer	CGU
task_id	The ID number of the task.	integer	CDGU
updated	The time the task was last updated.	time	G
updated_by	The ID number of the last staff member to update the task.	integer	CGU

Table 42: Task Pairs (Continued)

Name	Use	Type	Visibility
wf_flag	Determines whether business rules run when the task is created, updated, or deleted. • 0—Business rules do not run • 1—Business rules run	integer	CU

Appendix B
Source Codes

This appendix lists the source codes that can be used when creating and updating answers, contacts, incidents, opportunities, and organizations. Table 43 lists each level-one source and its corresponding code value. Table 44 on page 178 lists each level-two source, organized by level-one source, and its corresponding code value. If the level-one source code is 32001 for the Administration Console, the level-two source code corresponds to the ID number of the table. Table ID codes are located in Table 45 on page 181.

Caution If you do not include the source pairs in a function, the sources will automatically be set to indicate that the record was created from the XML API (source_lvl1=32007 and source_lvl2=6001). Setting the sources to any other values may have significant adverse effects on your data, so you should use caution and carefully test your work.

Table 43: Level-One Source Codes

Source	Code
Administration Console	32001
RightNow Console	32002
Accessibility Interface	32003
End-user interface	32004
RightNow Wireless	32005
Utilities	32006
Public API	32007
Outlook Integration	32008

Table 43: Level-One Source Codes (Continued)

Source	Code
Import	32009
Campaign or survey flow	32010

Table 44: Level-Two Source Codes

Level-One Source	Level-Two Source	Code
32001—Administration Console	Refer to Table 45 on page 181.	

Table 44: Level-Two Source Codes (Continued)

Level-One Source	Level-Two Source	Code
32002—RightNow Console	Incident editor	1001
	Contact editor	1002
	Organization editor	1003
	Opportunity editor	1004
	Task Instance editor	1005
	Answer editor	1006
	Mailing editor	1007
	Survey editor	1008
	Campaign editor	1009
	Document editor	1010
	Mailing format editor	1011
	Segment editor	1012
	Contact list editor	1013
	Offer Advisor	1014
	Answer propose	1015
	Opportunity create from incident editor	1016
	RightNow Live	1017
	Analytics	1018
32003—Accessibility interface	Incident editor	2001
	Contact editor	2002
	Organization editor	2003

Table 44: Level-Two Source Codes (Continued)

Level-One Source	Level-Two Source	Code
32004—End-user interface	Ask a Question	3001
	My Stuff—Questions	3002
	My Stuff—Profile	3003
	Pass-through authentication	3004
	Answer feedback	3005
	Site feedback	3006
	Survey response	3007
32005—RightNow Wireless	Administration incident edited or assigned	4001
	Ask a Question	4002
	My Stuff—Questions	4003
	My Stuff—Profile	4004
	Pass-through authentication	4005
	Answer feedback	4006
32006—Utilities (except *kimport*)	*techmail*—Service mailbox	5001
	techmail—Marketing mailbox	5002
	agedatabase—Closed incident with Waiting status	5003
	agedatabase—Answer set to review	5004
	agedatabase—Answer published	5005
	agedatabase—Answer decayed	5006
	dbstatus—Escalated	5007
	rnmd—Mailer daemon	5008

Table 44: Level-Two Source Codes (Continued)

Level-One Source	Level-Two Source	Code
32007—Public API	XML API	6001
	SOAP API	6002
	External event	6003
	Custom tab	6004
32008—Outlook Integration (during synchronization)	Contacts added or update	7001
	Threads and notes appended	7002
	Tasks added or updated	7003
32009—Import	Contact Upload	8001
	kimport utility	8002
32010—Flow	Campaign	9001
	Survey	9002

Table 45: Table ID Codes

Code	Table
1	incidents
2	contacts
3	orgs
4	links
6	tree
7	cluster
8	visibility
9	answers
10	meta_answers

Table 45: Table ID Codes (Continued)

Code	Table
11	ans_access
12	quotes
13	products
14	categories
15	custom_fields
16	rnl_chats
17	fattach
18	threads
19	statuses
20	menu_items
21	languages
22	std_content
23	map2meta_ans
24	accounts
25	ac_dashboard_items
26	interfaces
27	prodcat_notif
28	mailboxes
29	var2intf
30	holidays
31	billable_tasks
32	profiles
33	profile2intf
34	time_billed

Table 45: Table ID Codes (Continued)

Code	Table
35	inc2contacts
36	opp_phrases
37	dispositions
38	variables
39	queues
40	contact_types
41	sla2ans_access
42	slas
43	sla_instances
44	rr_intervals
45	rr2holidays
46	response_reqs
47	org_addrs
48	provinces
49	countries
50	org_addr_types
51	documents
52	contact_lists
53	mailings
54	mailing_formats
55	proofs
56	pipeline_snapshots
57	contact2list
58	tracked_links

Table 45: Table ID Codes (Continued)

Code	Table
59	audiences
60	dca_recs
61	link_categories
62	tmp_keyword
63	profile2queue
64	labels
65	hier_menus
66	dates
67	transactions
68	session_summary
69	user_trans
70	archived_incidents
71	phrases
72	ans_phrases
73	keyword_searches
74	ans_stats
75	stats
76	inc_performance
77	ans_notif
78	rule_alerts
79	segments
80	ruleacts
81	ruleconds
82	clicktrack

Table 45: Table ID Codes (Continued)

Code	Table
83	msg_types
84	rnl_staff_login
85	ma_trans
86	tmp_ext_keyword
87	opportunities
88	sa_strategies
89	sa_stages
90	sa_tasks
91	flow_map2state
92	sa_prod2sched
93	sa_products
94	purchased_products
95	queue_stats
96	currencies
97	exchange_rates
98	rules
99	rules_archive
100	sa_price_schedules
101	prod2quotes
102	rule_states
103	rule_escalations
104	mail_lists
105	mail_list2addr
106	task_instances

Table 45: Table ID Codes (Continued)

Code	Table
107	gap_report
108	ac_permissions
109	opp_performance
110	opp_snapshots
111	configuration
112	profile2layout
113	gap_info
114	gap_tree
115	db_maint_hist
116	locks
117	bounced_msgs
118	document_tags
119	mail_addrs
120	mail_groups
121	analytics_core
122	ac_nodes
123	mailing_stats
124	ac_alerts
125	ac_schedules
126	flows
127	surveys
128	questions
129	question_choices
130	ac_styles

Table 45: Table ID Codes (Continued)

Code	Table
131	question_sessions
132	ac_color_schemes
133	question_responses
134	rule_state_xitions
135	rnl_chat_activities
136	rule_log
137	opp2contacts
138	rnl_staff_activity
139	ac_run_vals
140	rx_email
141	data_imports
142	folders
143	cluster_class
144	cluster_info
145	cluster_tree
146	rnl_staff_engage
147	data_import_tmpl
148	sa_period2acct
149	sa_territories
150	sa_sales_periods
151	sa_contact_roles
152	event_queue
153	meta_ans_vis
154	integration_errors

Table 45: Table ID Codes (Continued)

Code	Table
155	oa_contacts
156	offer_phrases
157	offer_trans
158	offers
159	target2offers
160	segment_attributes
161	oa_segments
162	dependencies
163	campaigns
164	notes
165	survey_migration
166	meta_map
167	help_links
168	ac_scripts
169	flow_web_pages
170	dictionary
171	mail_queue
172	layouts
173	account_speed_dial
174	agent_acd_modes
175	profile2acd_mode
176	topic_words
177	cti_logins
178	cti_mode_changes

Table 45: Table ID Codes (Continued)

Code	Table
179	cti_calls
180	call_activity
181	similar_search_links
182	similar_searches
183	cti_current_calls
184	proof_recipients
185	proof_trans
186	contact_sessions
187	flow_entry_points
188	rnl_chat2ma
189	deleted_recs
190	isync_recs
191	topic_words_phrases
192	exclude_addrs
193	exclude_trans
194	prod_links
195	meta_ans_prod_links
196	rnl_queue2cats
197	rnl_agent_queue
198	rnl_user_queue
199	rnl_ext_queue_history
200	voice_stats
201	aims_data
202	label_menus

Table 45: Table ID Codes (Continued)

Code	Table
203	opp2competitor
204	rule_variables
205	ans_var_depends
206	pc_phrases
207	ac_filters
208	ac_tables
209	ac_columns
210	ac_params
211	ac_param_opts
212	ac_exceptions
213	ac_audit_log
214	ac_charts
215	ac_chart_src
216	ac_chart_styles
217	spider_track
218	nav_sets
219	nav_list_items
220	cti_custom_items
221	revenue_snapshots

Index

A

B

C

D

E

F

H

I

L

M

N

O

P

Q

R

S

T

U

V

X

Certain components of the software that are provided under license from RightNow belong to third parties that have authorized RightNow to sub-license the components:
"Apache Projects" (webservers and associated utilities and sub-projects) Copyright © 2000-2007 The Apache Software Foundation; "Sendmail" (sending email header files) is © 1983 to Eric P. Allman and © 1983, 1993 to the Regents of the University of California; "Mime pph" (library for email mime, encoding/decoding) is © 1996-1998 to Douglas W. Saunder; "Base64" (data encoding) is © 1996 to Internet Software Consortium; "Brill NLP Tagger" (natural language parser) is © 1993 MIT and University of Pennsylvania; "Expat XML Parser" (XML parser) is © 1998-2000 to Thai Open Source Software Center Ltd. and Clark Cooper and © 2001-2002 to Expat Maintainers; "GIF Image Reading Routines" (image reading routines) is © 1990, 1991, 1993 to David Koblas; "Semaphore Implementation" (generic semaphore routines for live server) is © 1995 to UMASS CS Dept.; "Scandir for Solaris and Win 32" (utility routines code) is © 1997-2003 to the PHP Group; "Regular Expression Library" (pattern matching library) is © 1992-1994 to Henry Spencer; "PDF Library" is © to PDFlib GmbH; "PopChart Enterprise" is © to Corda Technologies, Inc.; "eWeb Edit Pro + XML" is © to Ektron, Inc.; "SSCE Java Source" (spell checker) is © to Wintertree Software, Inc.; "libxslt" (xml transformation tool) is © 1998-2000 to David Veillard; "log4net" (logging tool) is © 1999 to the Apache Software Foundation; "awk data encoding" (data encoding) is © 1997 to Lucent Technologies; "Open SSL" (secure data transfer) includes cryptographic software written by Eric Young (eay@cryptsoft.com); "strftime" (utility routine code) is © 1989 the Regents of the University of California; "strtoll" (utility routine code) is © 1992 the Regents of the University of California; portions of "ExDataGrid" (active X control for data display) are © 2003 to Jan Tielens; portions of "svgdom" (vector graphics routines) are © 2002 to James W. Newkirk, Michael Two, and Alexei A. Vorontosov, and © 2000-2002 to Phillip A. Craig; "php" (dynamic web html programming language) is © 1999-2000 to the PHP Group; "PHP zip class" (ZIP file utilities) is © 2005 to Rochak Chauhan; "HtDig" (web index/search library) is © 1999-2005 to The HtDig Group; "CLucene" (information retrieval library) is © 2003-2005 to the CLucene team; "Snowball" (natural language parser stemmers) is © 2001, Dr. Martin Porter and Snowball Team; "ONC-RPC" (network utility routines) is © 1997 WD Klotz, 1993 by Martin F. Gergeleit and 1988 Sun Microsystems, Inc.; "Overlib" (Javascript Utilities) is Copyright Erik Bosrup 1998-2001; "cUrl" (networking utilities) © 1996-2007, Daniel Stenberg; "libmcrypt" (data security routines) © 1998, 1999, 2001 Nikos Mavroyanopoulos; "zlib" (data compression and utilities) is © 1995-2005 Jean-loup Gailly and Mark Adler; "HTML Tidy" (HTML/XML cleaning routines) 1998-2003 World Wide Web Consortium - WC3; "GroupableTableHeaderUI" (java code) is © 2004 D.I.A.L. S.L.; "win_service" (utility routines) is © Firebird Ashes Project and Inprise Corporation; "sgml entities" (Unicode data) is © 1997-1999 W3C; "HTML & XHTML DTDs" (HTML and XHTML grammar data) is © 1997-2007 W3C; "Mozilla Root SSL Certificates" (Secure data transfer) is © Mozilla; "Timer.java" (timing routines) is © 2000 The Apache Software Foundation; "YUI Ajax Library" (javascript and xml routines) is © 2006, Yahoo! Inc.; "GIFDECOD" (graphics/ image routines) is © 1990, 1991, 1993, David Koblas; "Mersenne Twister MT19937" (random number generator) is © 1997-2002, Makoto Matsumoto and Takuji Nishimura; "WindowsHook" (.NET utility routines) is © 2002, Dino Esposito & erms-rma project; "CRC32" (.NET data encoding) is © 2003 Thoraxcentrum, Erasmus MC; "zip compression" (.NET data compression) is © 2004-2005 DevelopDotNet, Alberto Ferrazzoli; "PropertiesBag" (.NET utility routines) is © 2002 Tony Allowatt; "RichTextBox Extensions" (.NET utility routines) is © 2003-2007 Pete Vidler; "FontConfig" (font utilities) is © 2001, 2003 Keith Packard; "Xrender/render" (graphics rendering) is © 2001, 2003 Keith Packard; "FreeType" (fonts) is © 1996-2002 by David Turner, Robert Wilhelm, and Werner Lemberg; "LibJPEG" (graphics library) is © 1991-1998, Thomas G. Lane and the Independent JPEG Group; "GIFLIB" (graphics library) is © 1997 Eric S. Raymond; "LibXML" (XML Parser) is © 1998-2003 Daniel Veillard; "glib/gthread/gmodule" (utilities) is © 2001-2008 The Mono Project and Contributors; "gdiplus" (graphics utilities) is © 2001-2008 The Mono Project and Contributors; "Mono Project" (.NET Vir-