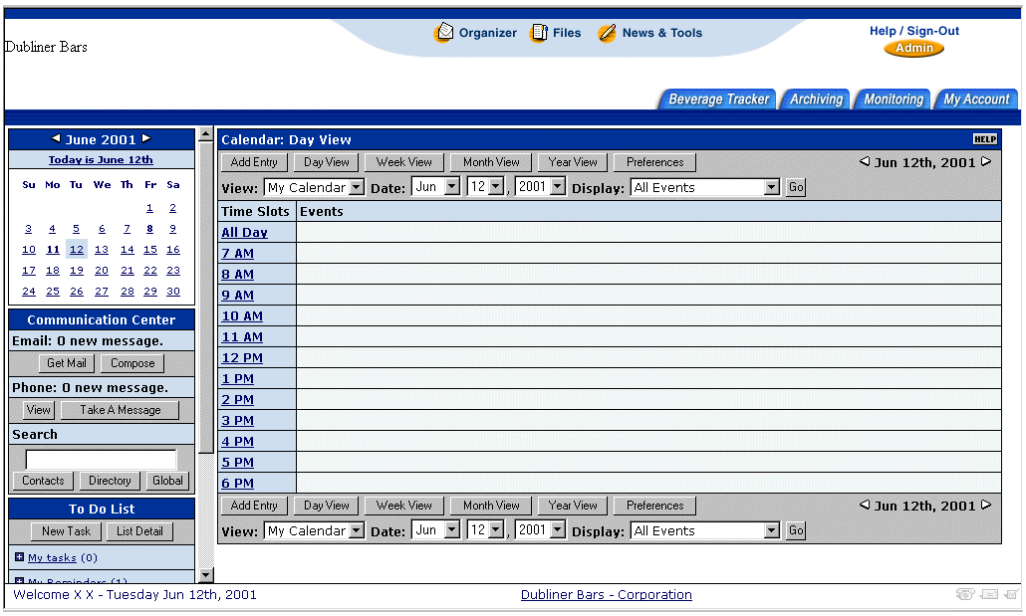
Service Provider Integration with Opendesk

Version 2.7
Rev. 1, June 2001
6/11/01 6:05 PM

Table of Contents

1. Introduction

The Opendesk platform was designed for simple and efficient integration with third-party applications. The purpose of this document is to describe the possible integration scenarios between Opendesk and the Service Provider. It will also outline the steps necessary to integrate Opendesk's Intranet software with the Service Provider's technology platform. The document is divided into two main sections - a general overview of integration scenarios, and a technical discussion of integration requirements (software, hardware).



2. Integration Scenarios

Depending on the Service Provider's existing technology resources, there are various integration scenarios that can be implemented by Opendesk. This section provides a general overview of possible scenarios. In all scenarios, the Service Provider controls branding (banner, theme selection, third-party content, etc) of the Opendesk product.

2.1 Opendesk embedded within Service Provider application

In some cases, the Service Provider has its own portal and/or its own user login interface. In this scenario, the Opendesk application suite can be accessed via a link or button in the Service Provider's interface. The Service Provider API receives the required login information (at least the user name and password) and then passes it on to the Opendesk API. After a successful login, the user has access to some or all of the Opendesk modules (depending on any application provisioning done by the Service Provider).

Benefits: minimal integration of the Opendesk product is required. Service Provider can sell Opendesk modules separately.

2.2 Service Provider application embedded within Opendesk

If the Service Provider does not have its own portal, Opendesk can serve as the default login page. Opendesk provides an API for the Service Provider to create, edit and delete intranets and users. Third-party applications are accessed via tabs or links embedded in the Opendesk suite. Opendesk provides a Backend Console that manages the tabs, and monitors usage of each application.

Benefits: since there is only one central API for login, this reduces risk of login failure. Centralization of all third-party apps in the Opendesk interface.

2.3 Third-party content embedded within Opendesk

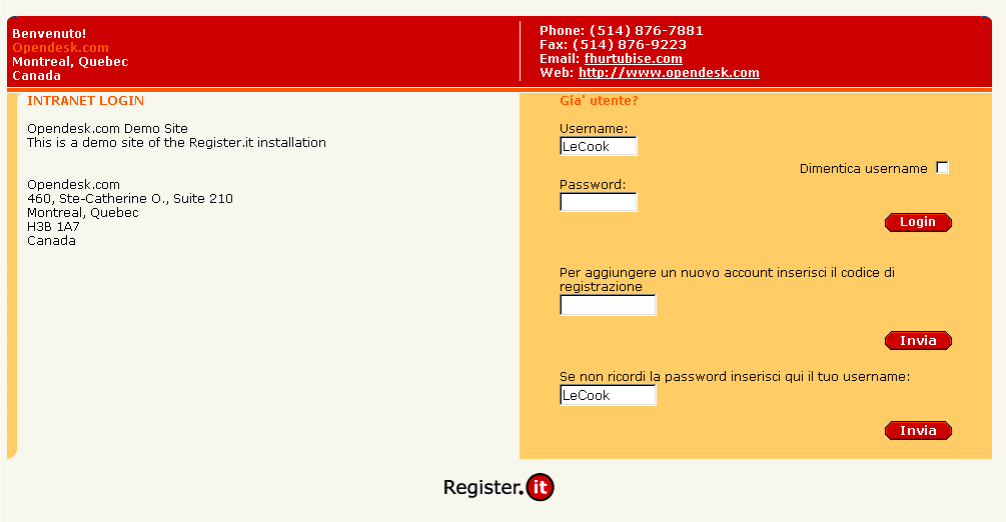
If the Service Provider is a content provider, the content can be delivered right from the Opendesk product, with the Backend Console. Opendesk provides the opportunity to distribute news, group discussions, company polls, banners, or third-party content to all intranets in the Service Provider's domain. In this scenario, Opendesk provides the user login interface. The Service Provider decides how the content will be presented on the interface.

Benefits: all content and applications are centralized in the Opendesk interface. Minimal resources required to manage the implementation.

2.4 Opendesk as stand-alone ASP

In some cases, Opendesk serves as an ASP enabler. Opendesk provides the client with all tools necessary to become an Application Service Provider. For example, an ISP can quickly become an ASP and offer value-adding web-based applications to generate more revenue.

Benefits: adds another revenue-generating component to Service Provider's business model.



3. Customization of Opendesk

Every aspect of the Opendesk product can be customized to suit the needs of the Service Provider. This section describes various options, and includes technical requirements.

3.1 Branding

If the Service Provider is using Opendesk as its default login interface, the entire product may be customized to be consistent with the Service Provider's branding. The Service Provider provides Opendesk with its choice of corporate logo, colors, product name, and the names of any third-party applications. The end-user may also be provided with a choice of user interface themes (as provided by the Service Provider and/or Opendesk). For more details, consult the "User Interface Customization" section in the Appendix.

3.2 Multi-language support

All Opendesk applications have multi-language support. Service Providers can purchase add-on language modules in the language of their choice, or provide their own translations based on template files provided by Opendesk. Once translated, end users can choose to view their user interface in one of the available languages.

For more details, consult the "User Interface Customization" section in the Appendix.

3.3 Application Layout

Because of its modularity, Opendesk is easily customizable in its layout. The location of each application (email, calendar, etc.) can be adjusted according to the requirements of the Service Provider. Furthermore, certain applications can be removed if the Service Provider already has that functionality from some other source. For example, Opendesk has been integrated with in-house and third-party email providers.

3.4 Application Functionality

Usually, the Service Provider will request additional functionality from existing Opendesk applications, or entirely new applications. For example, Opendesk can modify the contact management application to display industry-specific fields. The scheduling application can be enhanced to support more sophisticated workflow. Because many applications and modules are already pre-built, customization can be accomplished rapidly.

3.5 Login and Authentication

Opendesk can serve as the primary authentication system or act as a 'slave' to another authentication application. In either case, Opendesk provides an API for intranet creation, user creation, user roles, login

and authentication. These APIs allow Opendesk to synchronize authentication and user account information with the service provider. Opendesk uses SSL for the authentication of usernames and passwords. SSL is the industry standard for secure web transactions.

3.6 Default content

To assist new users and enhance a service provider's branding, Opendesk allows default content to be set for various applications. This content appears automatically when an Intranet is created without any action on the part of the end user. For example, service provider's can pre-set events in the calendar, contacts in the address book, and "getting started" literature in the document archive. These defaults can be updated and modified via Opendesk's Broadcast Tools.

3.7 Usage statistics

The Opendesk Backend Console provides usage statistics for each application. Opendesk can create custom reports for service providers to provide at-a-glance views of all aspects of the ASP's operations. Opendesk can also export custom reports for importing within a service provider's existing billing, rating and customer care software. Opendesk can also provide limited information on third-party applications embedded within Opendesk.

3.8 Documentation / Help Files

Licensees of Opendesk receive a variety of supporting documents, including a Sales Training Manual and a Customer Support Training Manual. Opendesk can work with the Service Provider to produce customized documents, whether they are in the form of manuals or marketing collateral.

The Opendesk application suite comes with its own Help Files, each application being linked to a corresponding Help File. The Service Provider provides the text for other Help Files (such as those for third-party applications) and this is easily integrated with the Opendesk Help Files. All Help Files are branded under the service provider's brand. For more details, consult the "User Interface Customization" section in the Appendix.

3.9 Data Center Requirements

While the Opendesk platform is designed to be compatible with a variety of data center environments, there are some requirements that must be fulfilled by the Service Provider in order to provide Opendesk in an optimal web environment. For more details, consult the "Hardware, Software and Network Requirements" section in the Appendix.

3.10 Typical Work Plan

In order to proceed with a work plan (illustrated below), both teams will need to create a Requirements Document that describes all details of the product to be delivered. Key elements of this document include:

- **Feature Set** – which Opendesk features to include, what customization (if any) is required, and any new applications that must be developed for the Service Provider.
- **Server Setup** – All server requirements (DNS, database etc) must be detailed in this document.
- **Branding** – Detailed description of branding issues (banner, color schemes, product names).
- **Documentation** – Any additional documentation required the Service Provider must be identified.
- **Long-term integration** – Depending on the time frame of the project, there may be two phases of implementation: short-term and long-term. Any discussion of long-term objectives would be of great assistance to the Opendesk production team.

Sample Work Plan

Item/details	Responsible	Date
Technical Meeting		
Requirements document		
Project plan		
Access to server machine for installation of Opendesk		
Approval/Signoff of Project Plan, Requirements & Architecture		
Working Beta: -Integrated login/authentication -Integrated administration tools -Integrated branding/UI		
Final Delivery/Demo Available: -Defaults and configuration options finalized and set -Documentation complete -Accounts populated with user information, example content		

Appendix

I. Login and Authentication

This section describes in greater detail the process by which the Service Provider's login API can be integrated with Opendesk's API.

General Description

1. Customer logs in to the Service Provider's platform using regular login/password authentication

2. Service Provider passes authentication information to Opendesk via APIs

3. Opendesk loads as the default environment in a new window or iFrame

4. Opendesk loads only those modules/features that the user has access to (controlled by Service Provider's service provisioning)

5. Administrators (as appointed by the Service Provider) will have an additional button on their interface linking to administrative tools (add users, delete users, set user role, etc)

In summary, the Service Provider will be responsible for the authentication of users and provisioning of all applications. Opendesk will simply respond to these inputs, bypassing our own built-in tools where necessary.

Account Setup & Synchronization

The following list describes the integration strategy for unifying login and other account information.

1. Initial call to account creation API

Before creating any users, a call to the Opendesk account (business, group, company) creation API will have to be made. Usually, when we create an account, we use **wildcard DNS** to create a domain so that those accounts can have simpler logins (e.g. domain.opendesk.com). This may not apply to an integrated product.

2. Unique usernames

Opendesk supports both globally unique usernames (i.e. unique across the entire user base) and account unique usernames.

3. Account maintenance

Each time a user is created/modified/deleted on the Service Provider side, a call needs to be made to the Opendesk APIs to duplicate the information. Such information (username/password) will only be able to be modifed via the Service Provider to avoid overlap (i.e. we will disable our account tools). For new users/accounts, the Service Provider can easily add API calls (standard HTTP calls) to existing account creation scripts. For existing accounts we will require a conversion script that would take the existing users/accounts and create them on Opendesk. It is not necessary that all users/accounts on the Service Provider exist on Opendesk, only those who have access to it.

Login Authentication

1) Pass the username/password over SSL. We would then re-authenticate the user without asking his username/password since it would be passed on the URL.

2) Use a validating API. We will still need a copy of the user information (username/pw). User logs in to the Service Provider and gets assigned a session ID. The Service Provider sends Opendesk the username and the session_id. Opendesk calls the Service Provider back to validate the authentication.

If the login is unsuccessful, an error page will be provided by Opendesk indicating that the login did not work.

Application Loading and User Preferences

On successful login, Opendesk loads the user preferences and displays the applications to which the user has access. Because Opendesk is an integrated suite of applications, we recommend against allowing administrators to turn on or off every application. Instead, we recommend bundling functionality that ensures that applications are always linked to complementary applications (e.g. Calendar links to Email and Task List for reminders).

Possible packages could include:

- Mobility – PalmOS/PocketPC synchronization, WAP
- Additional Users – Purchase additional users individually or in groups of 5 or 10
- Additional Storage Capacity – Increase default storage limit to allow additional storage for documents and email attachments

II. Administration

Opendesk has built-in tools for administering user accounts. There are two levels of administrative tools: user self-administration and Service Provider administration**.**

A link to the built-in user self-administration tools could appear in the Service Provider's user interface (e.g. as a button on the left-hand column). This simply loads the built-in Opendesk tools. In the long-term, the Service Provider may choose to use its own administration tools or access Opendesk's tools only via API calls, bypassing our user interfaces.

Similarly, Opendesk provides both APIs and a user interface for service providers to create and administrate user accounts. In the short term, we recommend providing a link to our built-in tools in the Service Provider's user interface. In the long term, the Service Provider can choose to use our tools, our APIs or build new tools.

III. Opendesk APIs vs Service Provider' XML API

This section is relevant to Service Providers that use XML API's.

The Opendesk Application Programming Interface uses API classes that do the actual work related to all account and user commands and report via HTTP CODES internally to the actual API script called Opendesk.pl.

This script simply takes the requested job, runs it via the API classes and returns the code returned by the API class directly. We can modify this script to interpret XML messages, still call the API classes internally and handle our error messages internally and return them a correctly formatted XML message.

Request / Output

A request example:

```
 <atxl:Message xmlns:atxl="http://opendesk.ServiceProvider.com/API/ServiceProvider.pl"
version="1.0" type="request">
 <!-- Create business operation -->
  <atxl:Object key="agent=Opendesk, operation="CreateBusiness">
   <atxl:Attribute name="BusinessSpecs">
    <atxl:AttributeValue>
     <Domain>tyrell</Domain>
     <Name>Tyrell Corporation</Name>
     <Address>123 Copperhead Road</Address>
     <Address2>Suite 210</Address>
     <City>Lansing</City>
     <StateProvince>Michigan</StateProvince>
     <Country>USA</Country>
     <Zip>12345</Zip>
     <Phone>514-876-9227</Phone>
     <LogoURL>http://www.opendesk.com/tyrell.gif</LogoURL>
     <Theme>Blues</Theme>
    </atxl:AttributeValue>
   </atxl:Attribute>
  </atxl:Object>
 </atxl:Message>
```

The subsequent reply:

```
 <atxl:Message xmlns:atxl="http://opendesk.ServiceProvider.com/API/ServiceProvider.pl"
version="1.0" type="response">
  <atxl:Object key="agent=Opendesk,operation="CreateBusiness">
   <atxl:Result>FAILURE</atxl:Result>
   <atxl:FaultCode>XLINK_FAULT_XML_OBJECT_DUPLICATE</atxl:FaultCode>
   <atxl:FaultReason>Business named Tyrell already exists.</atxl:FaultReason>
  </atxl:Object>
 </atxl:Message>
```

Such coding manipulation to our current API is of minor difficulty: parsing. We can use and adapt any of the Perl modules available on CPAN to parse XML, and extract the info needed.

Return and Error codes

XML error codes are somewhat different from the standard httpd response codes we used in our API:

HTTP CODE 200 -> OK
HTTP CODE 201 -> AUTHENTICATION FAILED
HTTP CODE 412 -> PRECONDITION FAILED
HTTP CODE 409 -> CONFLICT
HTTP CODE 500 -> SERVER ERROR

XML is virtually a “standardless standard” - it was meant as a way to create as many tag possibilities within the more largely accepted HTML syntax. What Opendesk's API parses out is different from customer to customer.

IV. Hardware, Software and Network Requirements

This section defines the minimal requirements for Opendesk to be installed within the Service Provider’s current network setup as a demo of the Service Provider/Opendesk, integrated as one seamless system.

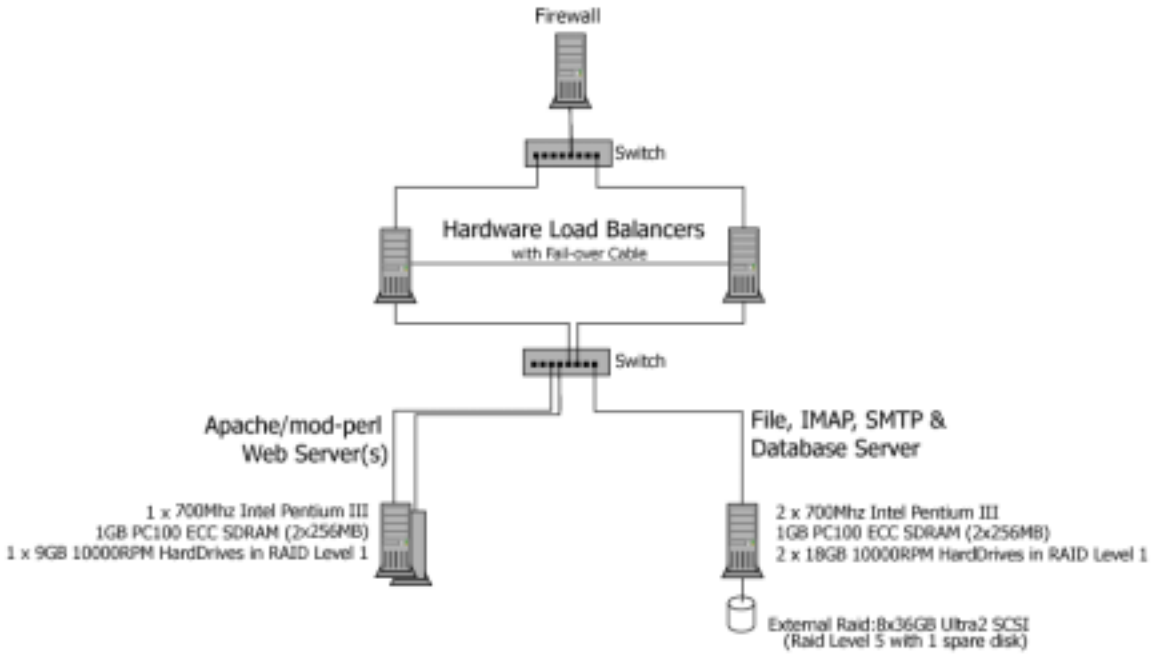
Hardware Requirements
- 2 x 700Mhz Intel Pentium III
- 1GB PC100 ECC SDRAM (2x256MB)
- 2 x 18GB 10000RPM HD in RAID Level 1

Software Requirements:
- Red Hat Linux 6.2 with Linux kernel 2.2
- OpenSSH-2.5.2p2 installed and active

Network Requirements:
- Fixed IP adresses
- Fully qualified HostName (Forward and reverse DNS)
- DNS zone control
- SSH access

Dual server setup for Full production environment

It is required for a production environment to have at least a two-server setup with load balancing. The figure below illustrates the requirements and proposes the potential addition of a RAID disk array as well as a second Web server. This allows an Opendesk implementation more flexibility in the long run, regardless of diverse user population habits.



Software Requirements:

- ❑ Red Hat Linux 6.2 with Linux kernel 2.2
- ❑ OpenSSH-2.5.2p2 installed and active

Network Requirements:

- ❑ Fixed IP adresses
- ❑ Fully qualified HostName (Forward and reverse DNS)
- ❑ DNS zone control
- ❑ SSH access

V. User Interface Customization

This section describes in greater detail the processes involved in customizing the Opendesk user interface to be consistent with the Service Provider's branding.

Since Opendesk is a private-label product, our design team can create a UI that reflects specific branding and corporate look and feel. This includes colors, layout, buttons and other design elements that do not affect functionality. The result is a totally unique-looking product.

The user interface is built on a system of "skins" or "themes" that can be customized for each customer or even by end users.

Multiple CSS support

The use of multiple cascading style sheets (CSS) defines the entire look and feel of the Opendesk product. Just about every cell and table uses some kind of style element to define colors and fonts. Cascading style sheets are an effective way to cut down on HTML file size. They eliminate the need to repeat background colors and fonts throughout the page, and allow for faster delivery of web content.

Opendesk uses a separate CSS file that is made up of custom elements, as opposed to using standard group classes, such as <P> and <H1>. Since the style sheet is not embedded in the applications, the Opendesk platform can have unlimited design customization, allowing anyone to customize a new scheme in a matter of minutes. Classes are applied in two ways, either as a tag element or a style definition.

Creating a new skin for the Opendesk platform requires three simple steps. This involves creating custom images, customizing the CSS file and changing some server configurations. A guide to that effect is available for service providers who would want multiple theme control.

Nomenclature control

Opendesk uses an extension of the Apache::Language Perl module for detecting a user's language settings and delivering content in that language. It uses the User-Agent language setting of the client's Web browser to determine the best language match.

Via Apache::Language, all text labels within Opendesk are encoded with a token system: TOKEN-VALUE-LANGUAGE. For example, the values (cow, 'vache', 'fr') and (cow, 'cow', 'en') are stored in the database. Whenever an Opendesk application needs to display the text label represented by the "cow" token, the language system automatically replaces the token with the correct version in the user's language, if such a translated version exists. If not, the default value appears.

This system not only allows multi-lingual capabilities but also gives the licensee full control through the creation of multiple 'token files' to store all language-dependent content separately from the application itself.

As a result, adding support for a new language or changing button names, panel titles and HTML links simply requires adding or updating the 'tokens' in the language module, without altering the application in any way. This principle applies to all Opendesk modules.

Help And Support Files

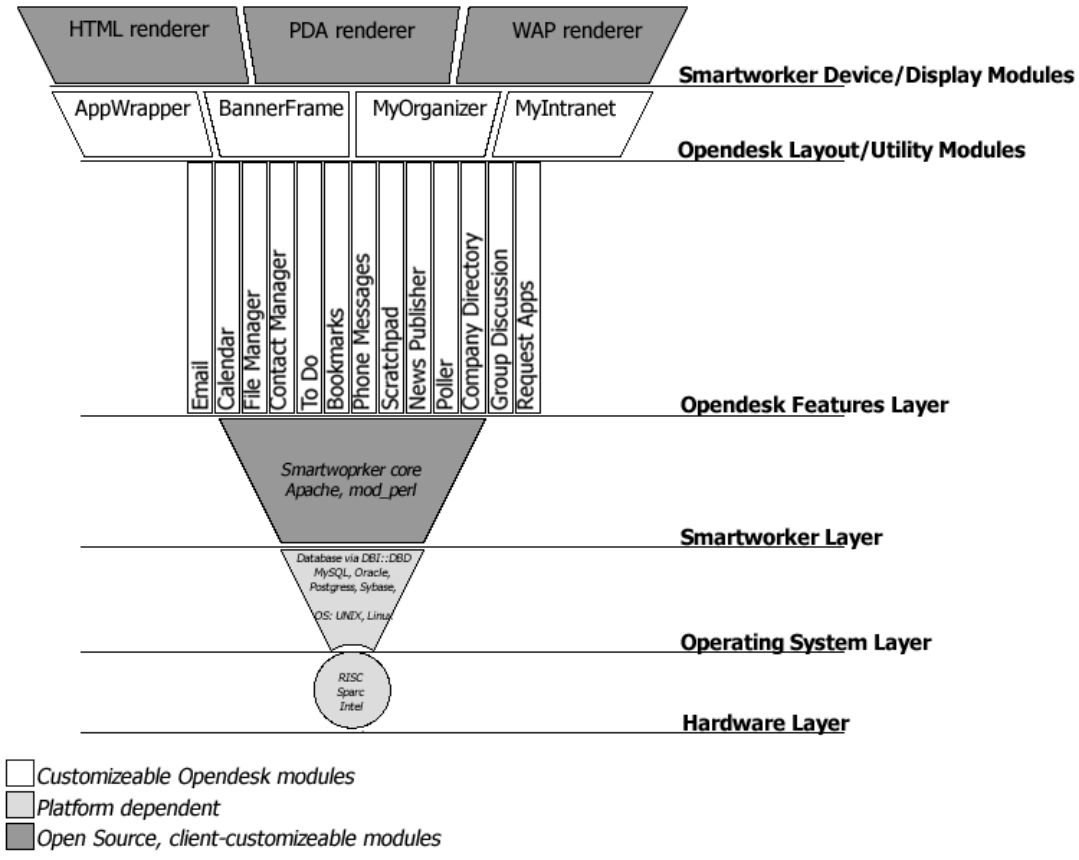
The help files structure, provided as easily edited and clearly named and structured HTML files, is organized to work with Opendesk's multi-language and multliple nomenclature support. The Service Provider will, with the help of a customization guide, have the ability to offer multi-lingual or specific use-case nomenclature to the help files for the specific modules provided.

Custom Application Layout

Through specific visual "widget" changes, some Opendesk modules dedicated to layout can be modified to group the existing feature set into various custom UI panel designs. For example, you may want to forgo the separation of the My Organizer and My Intranet frames and have a combination of news publisher and mail status in one UI panel. The many combinations of modules Opendesk offers can be rearranged according to your specific use cases and according to the Opendesk standard module dependencies.

Custom Module Assembly

Customization is not limited to design layout and nomenclature of the current feature set. For the Service Provider's particular mix of functionality, we can add, remove or slightly modify modules as well as their interdependencies to meet the needs of the target audience. The figure below illustrates the different parts of the Opendesk puzzle that can be modified.



VI. Opendesk API

Overview

The Opendesk Application Programming Interface (referenced as OD API) allows you to talk securely to the Opendesk core objects without requiring the use of the user intended forms. The OD API is used to provide a generic, programmatic way of manipulating Opendesk's objects, mostly to provide cross-system synchronization, or simply to replace user intended applications for registration. The API is also there to provide Service Provider administrators to be able to create their own administrative tools to speak with the Opendesk engine, thus providing easy technical support repair and maintenance access.

OD API Implementation

Protocol

The OD API uses the HTTP/1.1 (http://www.w3.org/Protocols/rfc2616/rfc2616.html) protocol as the mean of transport for the parameters. Both ends of the API communication must use the SSL (http://www.netscape.com/eng/ssl3/draft302.txt) protocol to ensure secure data transmission and reception.

Parameters

Requests are sent to the API using HTTP/SSL and the POST method. There are 2 MANDATORY parameters to any API requests.

The "command" parameter specifies the action to be performed (we discuss the possible actions in the next section).

The "domain" parameters specifies in which "business" domain the action is to be performed.
These 2 parameters must be present or a code 412 (Precondition Failed) will be returned.

Parameters and commands are sent to the API using your opendesk installation machine. Usually the URL is: https://youropendesk.install.com/API/Opendesk.pl

Available Commands

The Opendesk API contains default base commands that allows you to control your Opendesk objects. This is a brief overview of the command action available by default in the Opendesk API.

CreateBusiness

This function will create a new Business object.

UpdateBusiness

Update an existing Business object's profile and parameters.

DeleteBusiness

Delete an existing Business object.

EnableBusiness

Enable a business object, permitting user logins and business operations.

DisableBusiness

Disable a business object, denying user logins and business operations.

CreateUser

Create a new User object in a business domain.

UpdateUser

Update an existing user's profile.

DeleteUser

Delete an existing user in a business domain.

EnableUser

Enable a user object, permitting him to login to his business.

DisableUser

Disable a user object, denying him to login to his business.

SignOnUser

SignOnUser is used to perform seamless login procedures and "logs in" a user to the Opendesk engine.

These commands more detailed in the API Reference section.

Return codes

The OD API uses HTTP/1.1 specified return codes to confirm or refuse the tasks specified. Each of these code conforms to W3C standards as recent as draft 2616 of the HTTP/1.1 Protocol specifications.

Code 200 OK: Confirmation Code

The confirmation code for the API is specified by HTTP code 200 OK. When this code is returned, it means the requested action has been received and performed successfully.

Code 401: Unauthorized

The request requires user authentication. Either the authentication has not taken place, or is invalid.

Code 409: Conflict

A conflict has occurred. This can happen in multiple circumstances. As an example, trying to modify a business domain that does not exist, trying to create a user that already exist. In all cases, the body of the message will contain more information on the nature of the conflict.

Code 412: Precondition Failed

A condition before execution was not fullfilled. This code is usually returned to expose problems with parameters. In example, trying to modify a business without specifying a business name, or deleting a user without a username and a domain.

Code 500: Server Error

An error occured while executing the specified task. Either there is a problem with the Opendesk installation, or the database server is down, or something of higher severity has happened during the execution. This is a serious code, and should never be encountered after the Opendesk installation is completed.

VII. Opendesk API Reference

This section describes each of the available commands in more detail, while giving information that will help a smooth integration.

Each of the available base command actions are describes here, a complement document should accompany this manual to specify any extension that you might have requested to the base OD API. This section is very important for the integration because some action have to be performed before others for objects to be coordinated and complete.

Business domain related command actions.

These functions are related to business object manipulation.

CreateBusiness

This function will create a new business object. All parameters except the command and the domain are optional.

The business will then be available using its URL (usually businessdomain.yourdomain.com). Note that the business will not be operational until a user is created in its domain and is specified as Creator.

Parameters:

**** domain :*** Specifies the domain of the concerned business. (e.g. hbe for business hbe.opendesk.com)

**** name :*** Uniquely identifies the company name for this domain. e.g. HBE Software inc.

***address1 :** Address field for this domain. i.e. 460 St-Catherine West

***address2 :** Second address field for this domain. i.e. Suite 210

***city :** City for this business. i.e. Montreal

***state :** State or Province for this business i.e. Quebec.

***country :** Country for this domain. i.e. Canada.

***zip :** Zip/Postal code for this domain. i.e. H3B1A7

***phone :** Phone number for this domain. i.e. 514-876-9227

Return Codes

***200 :** The Business has been created successfully.

***409:** A conflict has occurred, usually means the business already exists.

***412:** A parameter is probably missing, like the business domain for example.

UpdateBusiness

This function will update an existing business object. Every parameter sent to UpdateBusiness will be updated in the object, including missing values.

The client application has to respecify every fields in an UpdateBusiness command.

Parameters

***domain :** Specifies the domain of the concerned business. (i.e. hbe for business hbe.opendesk.com)

***name :** Uniquely identifies the company name for this domain. i.e. HBE Software inc.

***address1 :** Address field for this domain. i.e. 460 St-Catherine West

***address2 :** Second address field for this domain. i.e. Suite 210

***city :** City for this business. i.e. Montreal

***state :** State or Province for this business i.e. Quebec.

***country:** Country for this domain. i.e. Canada.

***zip:** Zip/Postal code for this domain. i.e. H3B 1A7

***phone:** Phone number for this domain. i.e. 514-876-9227

Return Codes

***200 :** The Business has been updated successfully.

409: A conflict has occurred, usually means the business does not exist.

412: A parameter is probably missing, like the business domain for example.

DeleteBusiness

This function is used to delete an existing business object. The object will be wiped from the database.

Parameters

domain : Specifies the domain of the concerned business. (i.e. hbe for business hbe.opendesk.com)

Return Codes

200 : The Business has been deleted successfully.

409: A conflict has occurred, usually means the business does not exist.

412: A parameter is probably missing, like the business domain for example.

EnableBusiness

This function is used to enable a business object. The business will then be unlocked, and user logins will be allowed.

Parameters

domain : Specifies the domain of the concerned business. (i.e. hbe for business hbe.opendesk.com)

Return Codes

200 : The Business has been enabled successfully.

409: A conflict has occurred, usually means the business does not exist.

412: A parameter is probably missing, like the business domain for example.

DisableBusiness

This function is used to enable a business object. The business will then be locked, and user logins will be denied.

Parameters

domain : Specifies the domain of the concerned business. (i.e. hbe for business hbe.opendesk.com)

Return Codes

200 : The Business has been disabled successfully.

409: A conflict has occurred, usually means the business does not exist.

412: A parameter is probably missing, like the business domain for example.

User Accounts related command actions

This section describes command actions related to the management of users accounts in businesses.

CreateUser

This function is used to create a user inside a business object. All fields are optional except the domain, username and password.

Note that the password will be encrypted and will then be non-retrievable.

Parameters

domain : Specifies the domain of the concerned business. (i.e. hbe for business hbe.opendesk.com)

username : Specifies the username of the concerned user. (i.e. bbeausej)

password : Clear text version of the password for the concerned user.

firstname : First name of the concerned user.

lastname : Last name of the concerned user.

Email : Specifies the email of the concerned user.

is_admin : Specifies if the concerned user is to be considered as a Master Admin in his business.

is_creator : Specifies if the concerned user is to be considered as the Creator of his business.

Return Codes

200 : The User has been created successfully.

409: A conflict has occurred, usually means the user already exists in this business.

412: A parameter is probably missing, like the business domain for example.

UpdateUser

This function is used to update a user inside a business object. All fields are optional except the domain, username and password.

Note that the password will be encrypted and will then be non-retrievable. Every field will be updated, the client application should always specifiy all fields.

Parameters

domain : Specifies the domain of the concerned business. (i.e. hbe for business hbe.opendesk.com)

username : Specifies the username of the concerned user. (i.e. bbeausej)

password : Clear text version of the password for the concerned user.

firstname : First name of the concerned user.

lastname : Last name of the concerned user.

Email : Specifies the email of the concerned user.

is_admin : Specifies if the concerned user is to be considered as a Master Admin in his business.

is_creator : Specifies if the concerned user is to be considered as the Creator of his business.

Return Codes

200 : The User has been updated successfully.

409: A conflict has occurred, usually means the user does not exist in this business.

412: A parameter is probably missing, like the business domain for example.

DeleteUser

This function is used to delete a user inside a business object.

Parameters

domain : Specifies the domain of the concerned business. (i.e. hbe for business hbe.opendesk.com)

username : Specifies the username of the concerned user. (i.e. bbeausej)

Return Codes

200 : The User has been deleted successfully.

409: A conflict has occurred, usually means the user does not exist in this business.

412: A parameter is probably missing, like the business domain for example.

EnableUser

This function is used to enable a user inside a business object, allowing him from logging into the system.

Parameters

domain : Specifies the domain of the concerned business. (i.e. hbe for business hbe.opendesk.com)

username : Specifies the username of the concerned user. (i.e. bbeausej)

Return Codes

200 : The User has been enabled successfully.

409: A conflict has occurred, usually means the user does not exist in this business.

412: A parameter is probably missing, like the business domain for example.

DisableUser

This function is used to disable a user inside a business object, denying him from logging into the system.

Parameters

domain : Specifies the domain of the concerned business. (i.e. hbe for business hbe.opendesk.com)

username : Specifies the username of the concerned user. (i.e. bbeausej)

Return Codes

200 : The User has been disabled successfully.

409: A conflict has occurred, usually means the user does not exist in this business.

412: A parameter is probably missing, like the business domain for example.

VIII. API Extensibility

Session handling and API Sessions.

The Opendesk API is used as a base for custom API. Different client applications have different ways of doing authentication, synchronization or session creation.

The Opendesk API by default does not define these functions, as they are developed depending on the client application. There are many methods to validate session across the API, create sessions, validate users on the fly and track API transactions with an API session ID between the client and the API machine.

It is up to the client application to supply a specification on how sessions are handled. Opendesk will then provide API extensions to suit the need.

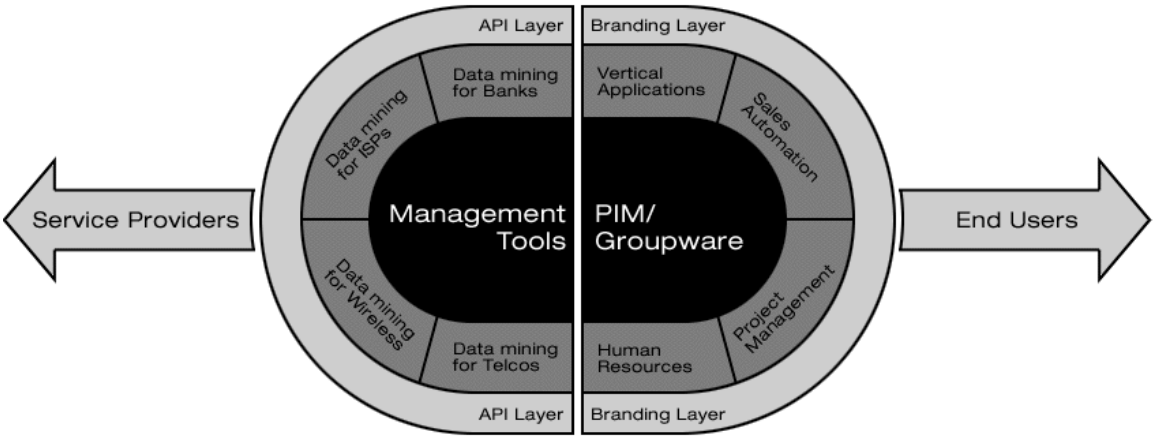
API Extensions

The API can and will be extended to the application level in the future. This will permit a client application to validate users, and access their data and represent it in a specific format that it can interpret. (i.e. XML/SOAP) The API will also be used to provide simple common functions and could then be used to serve as a utility server. (i.e. Provide unidentified password encryption, PGP key management etc.)

IX. Future Application Integration

The Opendesk team offers custom application services. In addition to regular product updates, new features that can be easily developed using Opendesk's object-oriented development system.

Through the use of shared database tables, external APIs or custom XML interfacing, we can assign our specification and usability team to design a new module specific to your needs. Opendesk application developers will then create modules and dynamic functionality targeted directly to your specific use-case offering:



Some examples:

- Provisioning systems
- Unified registration and login for existing ASP services
- Crawlers & Data-mining tools
- Sales automation
- Project Management tools for specific niches and/or project types
- Personalized news, weather or quotes content feeds
- Search engines, banner engines & servers
- FTP file browsing support
- POP or IMAP gateway listeners
- LDAP user databases through Opendesk API
- Audio-Video Conferencing gateways
- Etc