Installation and Deployment

Help Documentation

Installation and Deployment

Browser Requirements

SmarterTrack is fully supported by the browsers below.

- Google Chrome 10 and higher
- FireFox 3.6 and higher
- Safari 3 and higher
- Opera 10 and higher
- Internet Explorer 9 and higher

Mobile

With regards to mobile browsers, the SmarterTrack interface works well on most major browsers available for tablets and other larger-format mobile devices (e.g., Mobile Safari, Dolphin). For information on the SmarterTrack mobile interface for smartphones, see the mobile support outlined on the jQuery mobile framework website.

SmarterTrack System Requirements

SmarterTrack was designed to operate efficiently in shared, VPS or dedicated environments without any loss in functionality or performance based on the following minimum requirements:

- Windows 7, Windows 8** or Windows Server 2008 or higher, including Windows Server 2012**
- Microsoft .NET 4.0 running in full trust
- SmarterTrack Web server included with product*
- Microsoft SQL Server 2005 or higher, Microsoft SQL Server Express Edition or MySQL 5.1 or higher Note: At this time, MySQL 5.5.14 is the only version not compatible with SmarterTrack.

However, below are the SmarterTools recommended requirements :

- Windows Server 2008 R2 or higher
- 64-bit version of Windows Server is strongly recommended
- Microsoft .NET 4.0
- Microsoft IIS 7.0*
- Microsoft SQL Server 2008 R2 or higher

Note: Each installation and environment is unique. Extra load caused by excessive tickets, chats, agent accounts, and/or other factors may require more disk space, memory, database allocation, etc. than suggested in the online help. No warranty or guarantee is expressed or implied as to the efficacy or viability of these recommendations for a particular environment or application.

*SmarterTrack includes a basic Web server, so the product is fully functional upon installation—even without the existence of IIS or other Web servers. However, SmarterTools recommends installing Microsoft IIS 7.0 in place of the SmarterTrack Web server for increased performance and security in medium to high-volume environments. For more information, see Running SmarterTrack as an IIS Site .

***Only the most current versions of SmarterTrack (8.4 or higher) were tested on Windows 8 and Windows Server 2012. Previous versions may install on Windows 8 and/or Windows Server 2012 and run. However, some work arounds may be necessary to get them to run well. See the SmarterTools knowledge base for more information.

Installation

SmarterTrack is available as a traditional software installation or as a hosted service. Businesses that choose the traditional deployment of SmarterTrack can install the software onto a local or co-located server in dedicated, VPS, and shared environments using the standard install download. Note: Administrators planning to deploy SmarterTrack with failover functionality may have special considerations to make regarding installation. For more information, see SmarterTrack Failover Functionality .

Installation on a Dedicated Server

Please refer to the KB article How To - Install SmarterTrack on a Dedicated Server for step-by-step installation instructions for this environment.

Installation to a Shared Server

Please refer to the KB article How To - Install SmarterTrack to a Shared Server for step-by-step installation instructions for this environment.

Installation to SmarterTools Servers

Customers using SmarterTrack.com will have access to the program shortly after activating a free trial or purchasing a paid subscription. Because SmarterTrack is already installed on servers owned and maintained by SmarterTools, hosted service customers do not need to worry about installation. However, they will need to follow the instructions provided in their welcome emails to activate and set up their hosted service.

Running as an IIS Site

By default, SmarterTrack installs a basic Web server that allows companies to start using the application immediately after installation. However, SmarterTools recommends moving to a more robust and secure Web server, such as Microsoft IIS. For step-by-step instructions on configuring SmarterTrack to run with Microsoft IIS 7.5, please refer to the KB article How To - Set Up SmarterTrack as an IIS Site .

Note: This help topic assumes familiarity with Microsoft IIS and how it works. SmarterTools recommends using the basic Web server included with the SmarterTrack if you are unfamiliar with or uncomfortable using Microsoft IIS.

Setting Up the Database

SmarterTrack needs a database to store data. SmarterTools customers using an installed version of the software can run SmarterTrack with any of the database servers listed below. Note: SmarterTrack will only prompt you to automatically create a database if it does not already exist and you are connecting to a Microsoft SQL Server. In order to set up SmarterTrack with MySQL, the database will need created manually before starting the Setup Wizard.

Microsoft SQL Server (Express, Web, Standard, or Enterprise)

Microsoft SQL Server is the recommended database to use for SmarterTrack. Express Edition is available at no cost and will support the vast majority of SmarterTrack customers. For those needing advanced services or large amounts of database storage, Web, Standard or Enterprise Editions are recommended.

Microsoft SQL Server 2005 and later are supported, but for best performance, it is recommended to use Microsoft SQL Server 2008 R2 or later. If possible, use of 64-bit is also recommended.

For more information or to download SQL Server, please see:

- SQL Server Product Information
- SQL Server Express Download

For best results, when downloading and installing SQL Server, choose the Database With Tools download and install Management Studio Express with the database.

MySQL

MySQL can be used as a back-end database for SmarterTrack. MySQL Community Server can be

freely downloaded and used for most customers. For those needing advanced service and support, MySQL offers a paid Enterprise Edition.

MySQL versions 5.1 and later are supported, but it is recommended to use 5.5 or later if possible. In addition, use of 64-bit is recommended. Note: At this time, MySQL 5.5.14 is the only version not compatible with SmarterTrack.

For more information or to download MySQL Community Server, please see:

- Product Information
- MySQL Community Server Download
- MySQL Workbench

SmarterTrack will only support the InnoDB database engine and will attempt to use InnoDB for all upgrades and new installations. For information on converting a database to InnoDB, see the KB article How To Convert from MyISAM to InnoDB

Additional Information

Please refer to the KB articles How To Set Up the SmarterTrack Database on a Dedicated Server or How To - Set Up the Database for Shared Hosting for step-by-step instructions and more information regarding the database backend.

Activating SmarterTrack

In order for SmarterTrack to function for more than one agent, the product must be activated using a valid license key. In addition, if SmarterTrack is moved to another server or assigned to a different database, the product many need to be reactivated.

To access the product activation wizard, click the settings icon . Then expand the System Settings and Activation folders in the navigation pane and click Licensing . The edition, version, and license level information for the version of SmarterTrack currently being used will load in the content pane.

To activate or reactivate a valid license key, click Activate Key in the content pane toolbar. For step-by-step activation instructions, please refer to the KB article How To Activate SmarterTrack . Note: Activation of a license key requires the server to contact SmarterTools over port 443 (HTTPS). Please ensure that any firewall or internet security software you have installed allows an outgoing TCP port 443 request.

Upgrading SmarterTrack

Because the SmarterTrack download contains all of the installation files needed for any licensing level or edition, upgrading editions or levels is relatively easy. With a valid license key, companies can

easily upgrade to the Professional or Enterprise editions or increase the number of agents available in SmarterTrack. For more information, see Licensing . For step-by-step instructions, please refer to the KB article How To - Upgrade SmarterTrack Levels and Editions .

The steps for upgrading to SmarterTrack from an older version of the application vary depending on which installation package is used--the automatic installation package or the manual installation package. For more information, please refer to the KB article How To - Upgrade SmarterTrack .

SmarterTrack Failover Funtionality

Who Should Use This

This document is intended for use by system administrators that want to ensure maximum uptime of the SmarterTrack help desk. It provides architecture recommendations and considerations for deploying SmarterTrack in a failover environment. Note: Failover requires activation of SmarterTrack Enterprise with Failover Functionality. For licensing information for this product, contact the SmarterTools Sales Department.

Failover Overview

SmarterTrack Enterprise with Failover Functionality allows organizations to decrease the likelihood of service interruptions and virtually eliminate downtime by installing SmarterTrack on a passive failover server that is always connected to the database in standby mode. For businesses that use their help desk as a mission-critical part of their operations, failover functionality ensures the help desk remains online if there is a primary server failure.

Understanding How Failover Functionality Works

The main components of failover functionality are a primary server that acts as the default SmarterTrack server and manages the licensing of the server cluster and a secondary server that remains connected to the database in standby mode until the primary server experiences problems with network access or system hardware.

If the primary server fails, SmarterTrack will automatically enable the secondary server to active status. When this occurs, the secondary server takes over responsibility for processing background threads and supporting Web interface functionality. This server will remain in active status until another failure occurs or the system administrator manually changes the server status to passive.

SmarterTrack Database Server

System administrators have two options for setting up the SmarterTrack database in a failover deployment:

- The SmarterTrack database can be installed and configured on a single database server. Choosing this option means that there is a single point of failure; if the database server fails, SmarterTrack will go offline. In most cases, this setup is sufficient.
- 2For additional stability, the SmarterTrack database can be installed and configured on a clustered database server. Choosing this option means that multiple servers in the database cluster need to fail before SmarterTrack will go offline. If a clustered database is desired, the implementation, licensing and deployment of that cluster should be performed under the direction of a database administrator with related experience.

Regardless of the option chosen, it is important that all SmarterTrack servers can connect to the database(s) using the same connection string.

SmarterTrack Data Storage

It is important that every server in the failover cluster has access to the same set of physical files. For this reason, SmarterTools requires storing the SmarterTrack data files on a shared drive using iSCSI or a similar solution. Implementation is discussed in the steps to set up the primary and secondary failover servers (below).

For more information, refer to the knowledge base article How To – Store SmarterTrack Files on an Alternate Drive or SAN .

Setup of the Primary Server

Follow these steps to configure the primary server in the failover cluster:

- Download the SmarterTrack installation file from the SmarterTools website.
- Run the InstallShield Wizard.

- If you are installing SmarterTrack on a dedicated server, please refer to the knowledge base article How To – Install SmarterTrack on a Dedicated Server for step-by-step installation instructions for this environment.
- If you are installing SmarterTrack on a shared server, please refer to the knowledge base article How To – Install SmarterTrack to a Shared Server for step-by-step installation instructions for this environment.
- Set up the SmarterTrack database. For more information, see the SmarterTrack Database Server section of this document.
- Configure the server to store SmarterTrack data files on a shared drive. For more information, see the SmarterTrack Data Storage section of this document.
- Set up IIS. For more information, refer to the knowledge base article How To – Set Up SmarterTrack as an IIS Site

• Log in to the SmarterTrack management interface as the primary administrator (if you are not logged in automatically). By default, the username is “admin” and the password is “admin.”
• Run the Database Connection Wizard.
• Run the Company Setup Wizard to begin setting up your company's agents, groups, departments, and brands.

Setup of the Secondary Server

Follow these steps to configure the secondary (passive) server in the failover cluster:

• Install SmarterTrack on the secondary server.
• Set up IIS on the secondary server.
• Configure the secondary server to store SmarterTrack data files on the same shared drive as the primary server. For more information, see the SmarterTrack Data Storage section of this document.
• Log in to the SmarterTrack management interface as the primary administrator (if you are not logged in automatically). By default, the username is “admin” and the password is “admin.”
• Click the settings icon .
• Expand the System Settings and Setup folders and click Clustering in the navigation pane. Any servers on which this license of SmarterTrack has been activated will load in the content pane. To identify which server is the secondary server, look at the Status column. Any secondary servers will be listed as passive or disabled status.
• To enable the secondary server to act as a backup in the event the primary server fails, select the secondary server and click Enable in the content pane toolbar. The status of the secondary server will change to passive.

Note: Access to the portal is disabled on passive or disabled servers because issues can arise from two systems functioning simultaneously. To access the clustering settings on a passive or disabled server, go to http://[yourSmarterTrackURL]/login.aspx and log in as the primary administrator. Only the settings area of the management interface will be available.

Manually Activating a Failover Server

In the unlikely event that the automated failover process fails, system administrators can manually designate a server as the active server in the failover cluster. There are two methods to manually activate a failover server:

• Log in to the SmarterTrack management interface and click the settings icon . Expand the System Settings and Setup folders and click Clustering in the navigation pane. Select the desired server and click Set Active in the content pane toolbar to make the server the active server.

- Go to http://[yourSmarterTrackUrl]/ActivateNode.aspx on the server you wish to make active. Click the Activate Node button to make this server the active server.

These changes will be implemented within five seconds. Note: For security purposes, it is recommended that you restrict access in IIS to the “activatenode.aspx” page so that it cannot be accessed from outside networks .

Modifying Failover Servers

System administrators can add, delete or modify the status of a failover server at any time. Log in to the SmarterTrack management interface and click the settings icon . Expand the System Settings and Setup folders and click Clustering in the navigation pane. Then make any necessary modifications.

SmarterTrack External Providers

Who Should Use This Document

This document is intended for internal developers looking to extend or customize the functionality of SmarterTrack’s login system or ticket and live chat submission process. SmarterTrack users should note that the use of external providers is an advanced feature that requires programming knowledge.

External Providers Overview

SmarterTrack was built with custom configuration and integration in mind. In addition to being able to automate SmarterTrack via Web services, developers and/or system administrators have the ability to extend the functionality of the application through the use of external providers (third-party Web services). By integrating external providers into SmarterTrack, companies can integrate their login system to LDAP, show/hide custom fields based on the department a user chooses, populate fields, run reports based on externally provided information, and more.

In order for your external provider to work in SmarterTrack, the Web service(s) need to follow a specific interface. The external providers discussed in this document utilize the following interfaces:

- IExternalLoginProvider
- IExternalCustomFieldProvider
- IExternalTicketProvider
- IExternalChatProvider
- IExternalUserInfoProvider
- IExternalEventProvider

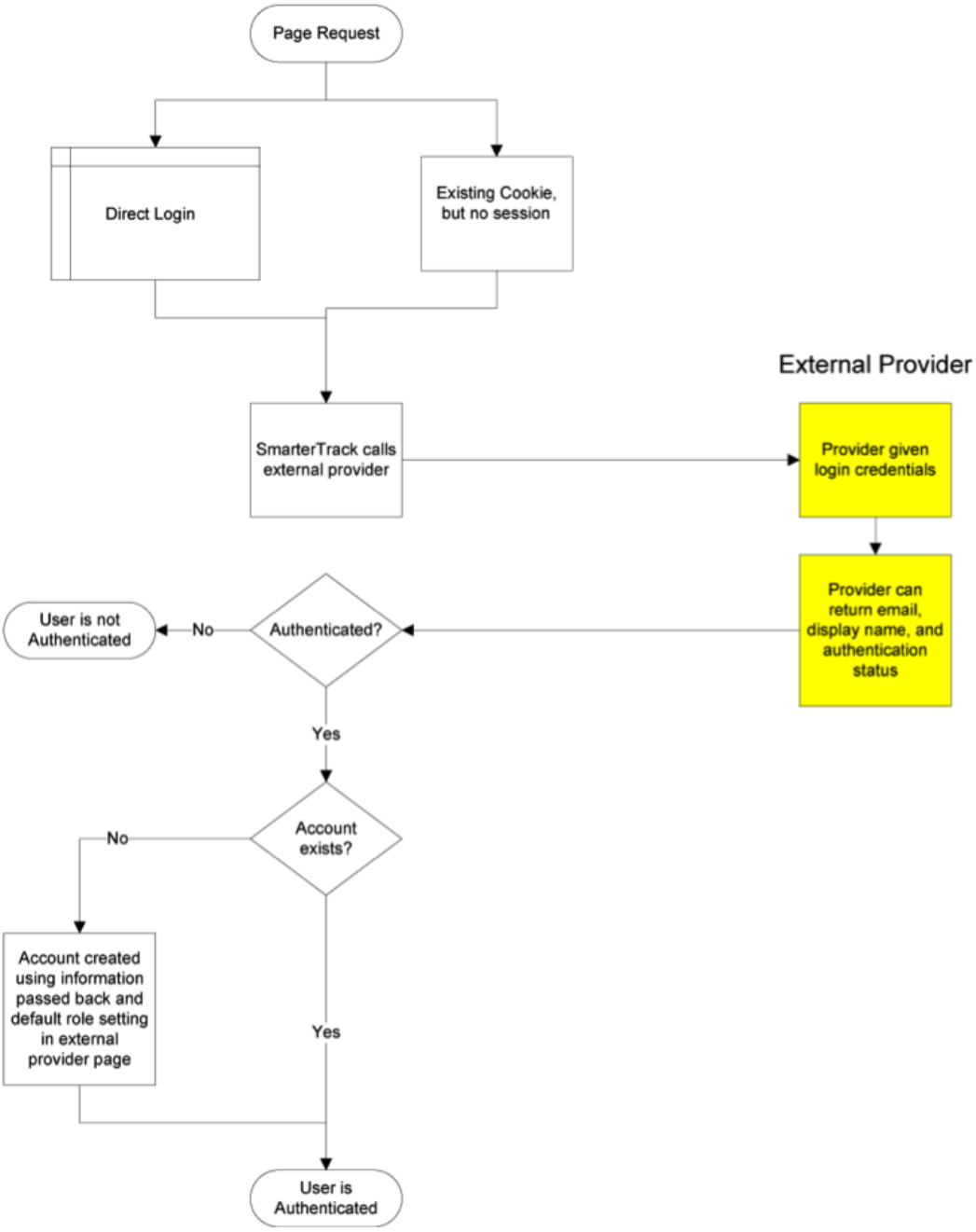
Multiple external providers can be created in the same Web service as long as the required interfaces are inherited. The interfaces needed for these external providers can all be found in the

SmarterTrack.Connector DLL. This file may be included in and distributed with your installation package.

the steps required to set them up, and an example on how to set up each Web service call.

Login Provider

The login provider allows you to customize SmarterTrack's login process. You can change the way users are authenticated in the system, create new users for unregistered logins, and even use cookies from other websites to allow for single sign-on. The login provider utilizes the IExternalLoginProvider interface. For a better understanding of how the login provider works with SmarterTrack, please refer to the flow chart below:



Authenticate Method

The authenticate method validates the username and password for access to the system. This call is made when a user attempts to log in through the portal without any existing authentication cookie.

- Parameters :

- Inputs :An object containing a generic list of string values. Each object in the list is stored in a specific format (variable=value)

- AuthPW :Used for validating if the calling software is valid
- LoginUsername :The username you are authenticating with
- LoginPassword :The password you are authenticating with

- Return Type :ExternalLoginProviderResult

- Message :A message indicating the result of the Web service call
- Success :A Boolean value indicating the result of the Web service call
- OutputVariables :A generic list of strings containing several output parameters needed by SmarterTrack. Some of the output is required in order to make SmarterTrack work correctly with this method. Each object in the list is stored in a specific format (variable=value)

- Authentication :Determines whether or not the authentication check succeeded. Possible values are OK or FAIL
- EmailAddress :The email address of the user
- DisplayName :The display name of the user

cf_* :The value of current custom fields. An example of a custom field would be cf_ticket type=phoneticket or cf_display name=Test. These custom field values will be assigned to the user who is logging in. If the provided custom fields do not exist, the value is ignored.

Here is an example of how the authenticate method is used:

```
// This function is used to authenticate user credentials when they login to the interface. // The return values are used to create and link an account from SmarterTrack into your system. // // // Inputs // authPW = Validates that calling software is valid // loginUsername // loginPassword // Outputs // Authentication = OK or FAIL // EmailAddress = Email address of user // DisplayName = Display Name of User [WebMethod] public ExternalLoginProviderResult Authenticate(ExternalLoginProviderInputs inputs) { ParsedLoginInputData iData = new ParsedLoginInputData(inputs); ExternalLoginProviderResult result; // Verify that all necessary criteria to call this function are met if (iData.WebServiceAuthorizationCode !=
```

```
WebServicePassword) return new ExternalLoginProviderResult(false,
"Permission Denied"); if (string.IsNullOrEmpty(iData.LoginUsername) ||
string.IsNullOrEmpty(iData.LoginPassword)) return new
ExternalLoginProviderResult(false, "Required Input Missing"); // This is a
sample of how to do a basic username/password check (use your own database
or list here) if (iData.LoginUsername.ToLowerInvariant() == "myusername" &&
iData.LoginPassword.ToLowerInvariant() == "mypassword") { result = new
ExternalLoginProviderResult(); result.Success = true; result.Message =
"Login Successful"; result.OutputVariables.Add("Authentication=OK");
result.OutputVariables.Add("EmailAddress=putUsersEmailAddressHere@example.com");
// optional result.OutputVariables.Add("DisplayName=putUsersFullNameHere");
// optional return result; } // Below is a sample Active Directory
Authentication implementation //string[] unDomainUsernameSplit =
loginUsername.Split(new char[] { '/', '\\' }, 2); //if
(unDomainUsernameSplit.Length == 2) //{ // if
(LDAP.AuthenticateLogin(unDomainUsernameSplit[0], unDomainUsernameSplit[1],
// loginPassword, LDAP.NetworkType.ActiveDirectory) ==
LDAP.LDAPResult.Authenticated) // { // return new
ExternalLoginProviderResult(true, "Login Successful", "Authentication=OK");
// } //} // Return a failure if there's no match return new
ExternalLoginProviderResult(true, "Login Failure", "Authentication=FAIL");
}
```

GetSignInCookieInfo Method

The GetSignInCookieInfo method validates the username and password for single sign-on from another website. This method is also called when an authentication cookie cannot be found for the currently logged in user. Your web.config file must be set up to use cookies from another site if you choose to use single sign-on. For more information, refer to the SmarterTools Knowledge Base

- Parameters
- Inputs :An object containing a generic list of string values. Each object in the list is stored in a specific format (variable=value)
- AuthPW :Used for validating if the calling software is valid
- LoginUsername :The username you are authenticating with
- Return Type :ExternalLoginProviderResult
- Message :A message indicating the result of the Web service call
- Success :A Boolean value indicating the result of the Web service call
- OutputVariables : A generic list of strings containing several output parameters needed by

SmarterTrack. Some of the output is required in order to make SmarterTrack work correctly with this method. Each object in the list is stored in a specific format (variable=value)

• Authentication :Determines whether or not the authentication check succeeded. Possible values are OK or FAIL
• EmailAddress :The email address of the user
• DisplayName :The display name of the user
• cf_* :The value of current custom fields. An example of a custom field would be cf_ticket type=phoneticket or cf_display name=Test. These custom field values will be assigned to the user who is logging in. If the provided custom fields do not exist, the value is ignored.

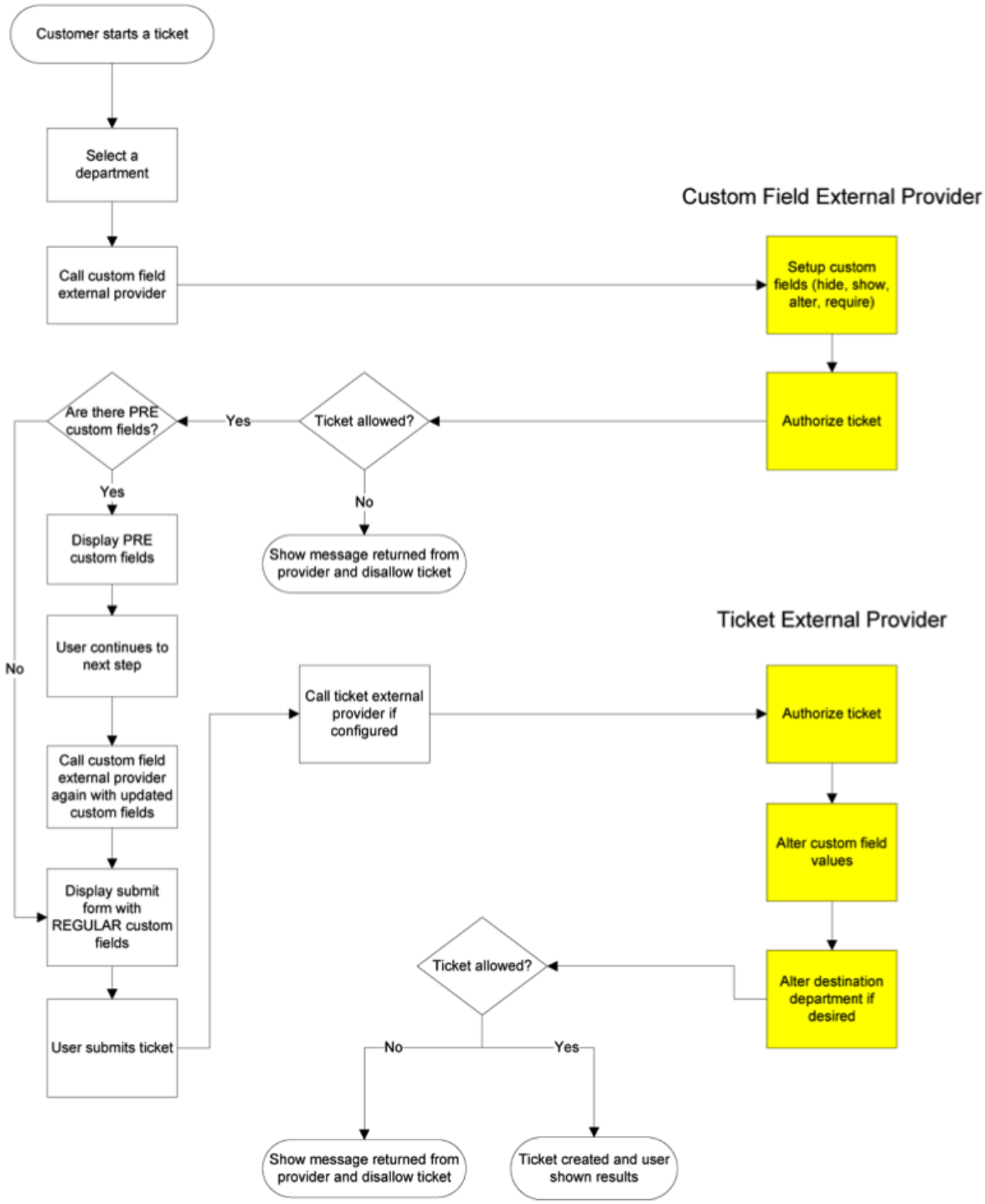
Here is an example of how the GetSignInCookieInfo method is used:

```
// Same as Authenticate function, but does not verify password because
cookie is authenticated. Therefore, your // logic will be identical to the
Authenticate function, but does not require you to check the password. //
// Inputs // authPW = Validates that calling software is valid //
loginUsername // Outputs // Authentication = OK or FAIL // EmailAddress =
Email address of user // DisplayName = Display Name of User [WebMethod]
public ExternalLoginProviderResult
GetSignInCookieInfo(ExternalLoginProviderInputs inputs) {
ParsedLoginInputData iData = new ParsedLoginInputData(inputs); // Verify
that all necessary criteria to call this function are met if
(iData.WebServiceAuthorizationCode != WebServicePassword) return new
ExternalLoginProviderResult(false, "Permission Denied"); if
(string.IsNullOrEmpty(iData.LoginUsername)) return new
ExternalLoginProviderResult(false, "Required Input Missing"); // This is a
sample of how to do a basic username/password check (use your own database
or list here) if (iData.LoginUsername.ToLowerInvariant() == "myusername") {
ExternalLoginProviderResult result = new ExternalLoginProviderResult();
result.Success = true; result.Message = "Login Successful";
result.OutputVariables.Add("Authentication=OK");
result.OutputVariables.Add("EmailAddress=putUsersEmailAddressHere@example.com");
// optional result.OutputVariables.Add("DisplayName=putUsersFullNameHere");
// optional return result; } // Below is a sample Active Directory
Authentication implementation //string[] unDomainUsernameSplit =
loginUsername.Split(new char[] { '/', '\\' }, 2); //if
(unDomainUsernameSplit.Length == 2) //{ // if
(LDAP.AuthenticateLogin(unDomainUsernameSplit[0], unDomainUsernameSplit[1],
// "", LDAP.NetworkType.ActiveDirectory) == LDAP.LDAPResult.Authenticated)
// { // return new ExternalLoginProviderResult(true, "Login Successful",
```

```
"Authentication=OK"); // } //} return new ExternalLoginProviderResult(true,
"Login Failure", "Authentication=FAIL"); }
```

CustomFieldProvider

The custom field provider allows you to customize the custom fields that display when a user submits a ticket or live chat and the values associated to each custom field. The custom field provider utilizes the IExternalCustomFieldProvider interface. For a better understanding of how the custom field provider works with SmarterTrack, please refer to the flow chart below:



GetCustomFieldOptions Method

The GetCustomFieldOptions method allows you to specify which custom fields a user may complete into two distinct steps. This is useful because it enables companies to allow their customers to complete certain fields during the first step of the live chat or ticket submission process and use the information customers provide to determine which custom fields are displayed during the second step.

People often confuse this method with the StartTicket or StartChat methods of other external providers, but these methods perform slightly different functions. See the Ticket Tracking Provider and Live Chat Tracking Provider sections of this document for more information.

• Parameters :

• Inputs :An object containing a generic list of string values. Each object in the list is stored in a specific format (variable=value)

• AuthPW :Used for validating if the calling software is valid
• LoginUsername :The customer's username
• DepartmentName :The department name that the Ticket or Live Chat is being started through
• Location :This is the step that the user is currently on in the Ticket or Live Chat submission process. Possible values for this are PRE or REGULAR
• cf_* :Value of current custom fields. An example of a custom field would be cf_ticket type=phoneticket or cf_display name=Test. These values would obviously be empty if you were on step 1 of the submission process (location=PRE)

• Return Type :ExternalCustomFieldProviderResult

• Message :A message indicating the result of the Web service call
• Success :A Boolean value indicating the result of the Web service call
• CustomFieldDataItems :A generic list of properties that will tell SmarterTrack which action(s) to take with each individual custom field

• CustomFieldName :This is the identifier for the custom field being changed. This value must be identical to the name of the custom field in SmarterTrack
• DisplayWithInformationLookup :A Boolean value that indicates whether or not this field should be displayed on the first step of the ticket or live chat submission process. If this value is set to false, this custom field will be displayed with the other custom fields on the regular submission page (location=NORMAL)
• ChangeListOptions :A Boolean value that indicates whether or not a drop down type custom field should have its list options changed. This works alongside the ListOptions property below
• ListOptions :A generic list of strings that is used to overwrite the contents of a drop down

menu type custom field. These values will only be used if ChangeListOptions is set to true

• ChangeRequired :A Boolean value that indicates whether or not the custom field should have its required status changed. This works alongside the NewRequiredValue property

• NewRequiredValue :A Boolean value that is used to determine if this custom field should be marked as required. This value is only used if ChangeRequired is set to true

• ChangeVisible :A Boolean value that indicates whether or not the custom field should have its visibility changed. The works alongside the NewVisibleValue property

• NewVisibleValue :A Boolean value that is used to determine if this custom field should be visible to the user. This value is only used if ChangeVisible is set to true

• OutputVariables :A generic list of strings containing several output parameters needed by SmarterTrack. Some of the output is required in order to make SmarterTrack work correctly with this method. Each object in the list is stored in a specific format (variable=value)

• CanStartTickets :Determines whether or a not a ticket can be started. Possible values for this are true and false. If this value if false,

• CanStartChats : Determines whether or a not a live chat can be started. Possible values for this are true and false. If this value if false, SmarterTrack will display an error message to the user

Here is an example of how the GetCustomFieldOptions method is used:

```
// Inputs // authPW = Validates that calling software is valid //
loginUsername // departmentName // location = PRE or REGULAR (pre called
after department selection made. Regular called after search page and
before compose page) // cf_* = value of current custom fields, such as
"cf_ticket type=phone ticket" // Outputs // CanStartTickets = true or false
// CanStartChats = true or false // Custom field overrides [WebMethod]
public ExternalCustomFieldProviderResult
GetCustomFieldOptions(ExternalCustomFieldProviderInputs inputs) {
ParsedCustomFieldInputData iData = new ParsedCustomFieldInputData(inputs);
// Verify that all necessary criteria to call this function are met if
(iData.WebServiceAuthorizationCode != WebServicePassword) return new
ExternalCustomFieldProviderResult(false, "Permission Denied"); bool
canStartTickets = true; bool canStartChats = true;
ExternalCustomFieldProviderResult result = new
ExternalCustomFieldProviderResult(true, "Result okay");
ExternalCustomFieldData cfdata; #region Example: Allow tickets/chats to
only certain customers /* * You could check a database and see if the user
has the ability to contact certain departments, for example. * In this
example, only the user "myusername" is allowed to start tickets and chats
for all departments. * Everyone else can only start chats (not tickets),
and can only contact the sales department */ //if (iData.LoginUsername ==
```

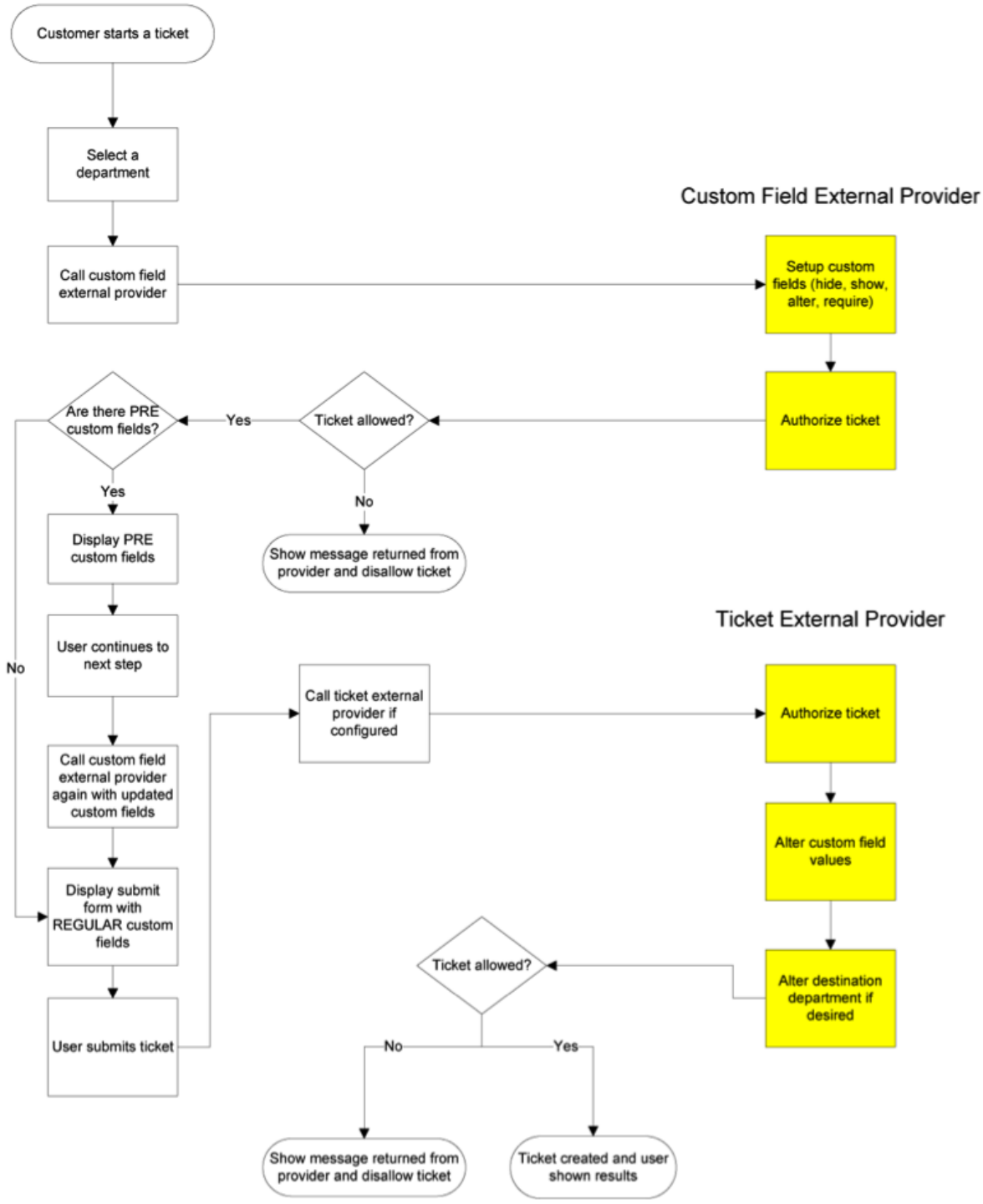
```
"myusername") //{ // canStartTickets = true; // canStartChats = true; //}
//else //{ // result.Message = "You are not currently permitted to contact
that department."; // shown in ticket pages // canStartTickets = false; //
if (iData.DepartmentName.ToUpperInvariant() == "SALES DEPARTMENT") //
canStartChats = true; // else // canStartChats = false; //} #endregion
#region Example: Set a custom list of domains for them to choose from /* *
This example shows how to fill custom field drop-down lists for the user
based on their login. In this case, * we fill a dropdown list with their
accounts listed in our database, so that they can choose which one * the
support is for. * * It is important that the custom field already exist in
SmarterTrack * */ //if (iData.DisplayLocation == DisplayLocations.Page1_Pre
&& iData.DepartmentName.ToUpperInvariant() == "SUPPORT DEPARTMENT") //{ //
// Here you would look up the options. For now, we'll assume we brought
back abc.com and def.com from our database

// cfdata = new ExternalCustomFieldData(); // cfdata.CustomFieldName =
"Account"; // cfdata.ChangeRequired = true; // cfdata.NewRequiredValue =
true; // cfdata.ChangeListOptions = true; //
cfdata.DisplayWithInformationLookup = false; // Set this to true if you
need to show/hide other custom fields based on the value they select. //
cfdata.ListOptions.Add("abc.com"); // cfdata.ListOptions.Add("def.com"); //
result.CustomFieldDataItems.Add(cfdata); //} #endregion #region Example:
Custom Issue Types by Department /* * This example shows how to fill custom
field drop-down lists for the user based on the department chosen. In this
case, * we fill a dropdown list with possible types of issues * * It is
important that the custom field already exist in SmarterTrack * */ //cfdata
= new ExternalCustomFieldData(); //cfdata.CustomFieldName = "Type of
Issue"; //cfdata.ChangeListOptions = true;
//cfdata.DisplayWithInformationLookup = false; //cfdata.ChangeRequired =
true; //cfdata.NewRequiredValue = true; //cfdata.ChangeVisible = true;
//cfdata.NewVisibleValue = true; //if
(iData.DepartmentName.ToUpperInvariant().IndexOf("SALES") != -1) //{ //
cfdata.ListOptions.Add("Change Plan"); // cfdata.ListOptions.Add("Help with
order"); // cfdata.ListOptions.Add("Questions about plans"); //
cfdata.ListOptions.Add("Questions about features"); //
cfdata.ListOptions.Add("Help finding a product"); //} //else if
(iData.DepartmentName.ToUpperInvariant().IndexOf("SUPPORT") != -1) //{ //
cfdata.ListOptions.Add("Web site issues"); //
cfdata.ListOptions.Add("Database issues"); // cfdata.ListOptions.Add("Email
issues"); // cfdata.ListOptions.Add("Dedicated server issues"); //
cfdata.ListOptions.Add("Statistics issues"); //
cfdata.ListOptions.Add("Network connectivity"); //} //else //{ // //
```

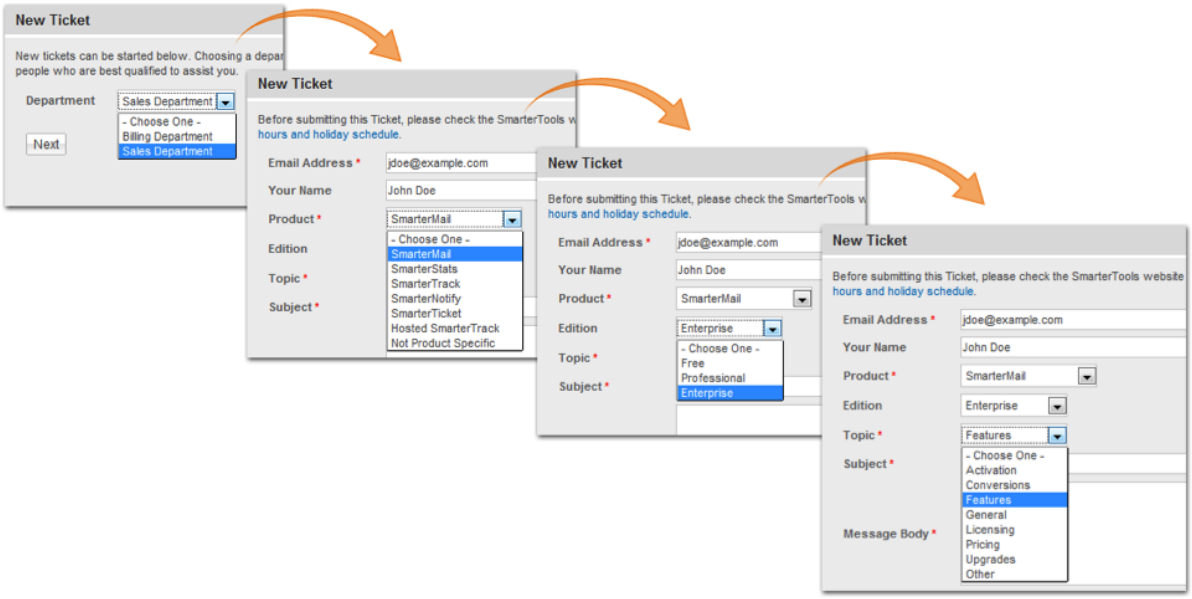
```
Removing issue types field, since there's no options //
cfdata.ChangeRequired = true; // cfdata.NewRequiredValue = false; //
cfdata.ChangeVisible = true; // cfdata.NewVisibleValue = false; //}
//result.CustomFieldDataItems.Add(cfdata); #endregion result.Success =
true; result.OutputVariables.Add("CanStartTickets=" + canStartTickets);
result.OutputVariables.Add("CanStartChats=" + canStartChats); return
result; }
```

Ticket Tracking Provider

The ticket tracking provider is used to determine if a ticket can be started based on the custom field values that have been entered and based on the needs for your company. The ticket tracking provider utilizes the IExternalTicketProvider interface. For a better understanding of how the ticket tracking provider works with SmarterTrack, please refer to the flow chart below:



In the example below, the custom field provider is used to determine the options available on each step in the ticket submission process and the ticket tracking provider uses the options selected to determine whether the ticket can be started.



StartTicket Method

The StartTicket method is used to tell SmarterTrack if the ticket can be created. This method is called just before a ticket is submitted (regardless of how the ticket was created). This method may also change the custom fields for the ticket and can be used to override the customer's department selection.

- Parameters :

- Inputs :An object containing a generic list of string values. Each object in the list is stored in a specific format (variable=value)

- AuthPW :Used for validating if the calling software is valid
- LoginUsername :The customer's username. This is used for tickets submitted through the Web interface
- EmailFrom :The customer's email address. This is used for tickets submitted through email.
- DepartmentName :The department's name that the customer is creating a Ticket for.
- TicketNumber :The ticket number for the ticket about to be created.
- cf_* :Value of this Ticket's current custom fields. An example of a custom field would be cf_ticket type=phoneticket or cf_display name=Test. At this point in the ticket submission process these values should be completed.

- Return Type :

- Message :A message indicating the result of the Web service call
- Success :A Boolean value indicating the result of the Web service call
- OutputVariables :A generic list of strings containing several output parameters needed by

SmarterTrack. Some of the output is required in order to make SmarterTrack work correctly with this method. Each object in the list is stored in a specific format (variable=value)

• TicketCreation :Determines whether or a not this ticket can be created. Possible values for this are true and false. If this value if false, SmarterTrack will display an error message to the user and will NOT create the Ticket
• departmentName :If this value is set, the ticket will be created for this department instead of what the user submitted originally
• groupName :If this value is set, the ticket will be created for this group and the department this group is associated with instead of what the user submitted originally.
• departmentID :If this value is set, the ticket will be created for this department instead of what the user submitted originally
• groupID : If this value is set, the ticket will be created for this group and the department this group is associated with instead of what the user submitted originally.
• cf_* : Value of this ticket's current custom fields. If these values are set, they will replace any existing values for the custom field

Here is an example of how the StartTicket method is used:

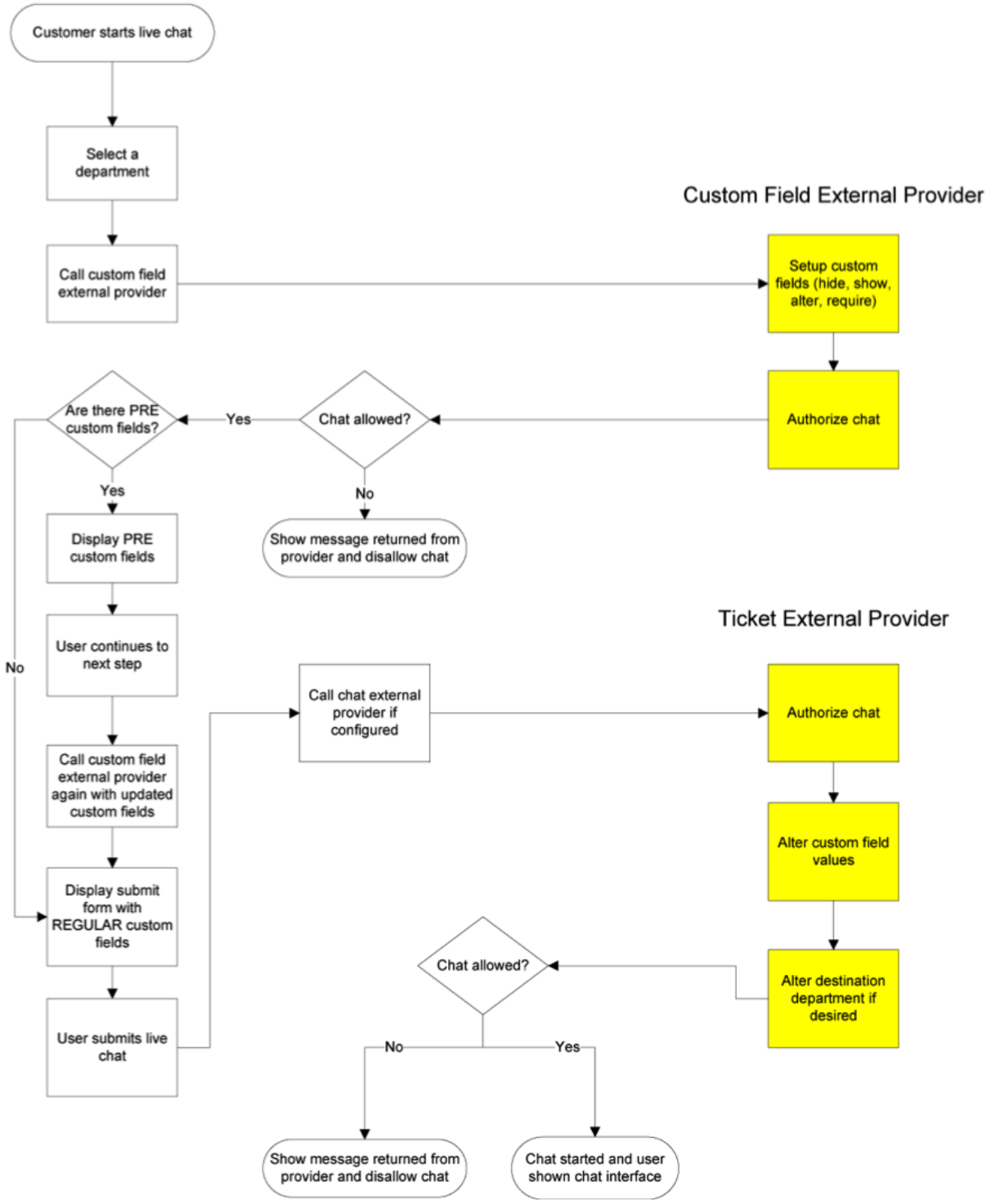
```
// This function is called right after the person clicks on the Submit Ticket button. This gives your provider one last // chance to abort the process or store any custom fields. // // Inputs // authPW = Validates that calling software is valid // loginUsername (for tickets submitted through Web interface) // emailFrom (for tickets submitted through email) // departmentName // subject // ticketNumber = the ticket number about to be created // cf_* = values of custom fields // Outputs // TicketCreation = TRUE or FALSE // cf_* = output values of custom fields [WebMethod] public ExternalTicketProviderResult StartTicket(ExternalTicketProviderInputs inputs) { ParsedTicketInputData iData = new ParsedTicketInputData(inputs); ExternalTicketProviderResult result = new ExternalTicketProviderResult(); // Verify that all necessary criteria to call this function are met if (iData.WebServiceAuthorizationCode != WebServicePassword) return new ExternalTicketProviderResult(false, "Permission Denied"); // This would be the place to check custom fields and the person's account to see if they can // submit tickets. If they can, and you use a pay-per-ticket model, deduct the ticket from // the person's account here. // For example, to return an error that the person is out of support tickets, use the following code // return new ExternalTicketProviderResult(true, "There are no more tickets in your account.", "TicketCreation=FALSE"); #region Example: Set the customer's custom fields /* * In this example, we set some hidden custom fields based on the logged in user's information. These *
```

```
custom fields MUST exist already in SmarterTrack (without the cf_ prefix) *
*/ //if (iData.LoginUsername.ToLowerInvariant() == "myusername") //{ //
result.OutputVariables.Add("cf_Billing ID=4126"); //
result.OutputVariables.Add("cf_Plan Type=ASP.NET Semi-dedicated"); //}
#endregion #region Example: Reroute tickets /* * In this example, we
reroute any tickets with the subject containing Urgent to the Critical
Support department. * We also reroute any tickets with a custom field value
of "Server Down" to the Server Operations Department * (assuming we had
that custom field set up in our system). * You could also modify this to
reroute customers with certain attributes in your database (priority
customers). * */ //if (iData.Subject.ToUpperInvariant().Contains("URGENT"))
//{ // result.OutputVariables.Add("departmentName=Critical Support"); //}
//string value; //if (iData.CustomFields.TryGetValue("Issue Type", out
value) && value.ToUpperInvariant() == "SERVER DOWN") //{ //
result.OutputVariables.Add("departmentName=Server Operations Department");
//} #endregion

result.Success = true; result.Message = "Ticket Creation Successful";
result.OutputVariables.Add("TicketCreation=TRUE"); return result; }
```

Live Chat Tracking Provider

The live chat tracking provider is used to determine if a live chat can be started based on the custom field values that have been entered and based on the needs for your company. The live chat tracking provider utilizes the IExternalChatProvider interface. For a better understanding of how the live chat tracking provider works with SmarterTrack, please refer to the flow chart below:



StartChat Method

The StartChat method is used to tell SmarterTrack if the live chat can be started. This method is called just before a live chat is submitted to the queue. This method may also change the custom fields for the live chat and can be used to override the customer's department selection.

- Parameters :

• Inputs :An object containing a generic list of string values. Each object in the list is stored in a specific format (variable=value)

• AuthPW :Used for validating if the calling software is valid
• LoginUsername :The customer's username. This is used for live chats submitted through the Web interface
• DepartmentName : The department's name that the customer is starting a live chat for
• cf_* :The value of the live chat's custom fields. An example of a custom field would be cf_chat type=support or cf_display name=Test. At this point in the live chat submission process these values should be completed.

• Return Type : ExternalCustomFieldProviderResult

• Message :A message indicating the result of the Web service call
• Success :A Boolean value indicating the result of the Web service call
• OutputVariables :A generic list of strings containing several output parameters needed by SmarterTrack. Some of the output is required in order to make SmarterTrack work correctly with this method. Each object in the list is stored in a specific format (variable=value)

• ChatCreation : Determines whether or a not this live chat can be created. Possible values for this are true and false. If this value if false, SmarterTrack will display an error message to the user and will NOT start the live chat
• departmentName :If this value is set, the live chat will be created for this department instead of what the user submitted originally
• cf_* :Value of this live chat's current custom fields. If these values are set, they will replace any existing values for the custom field

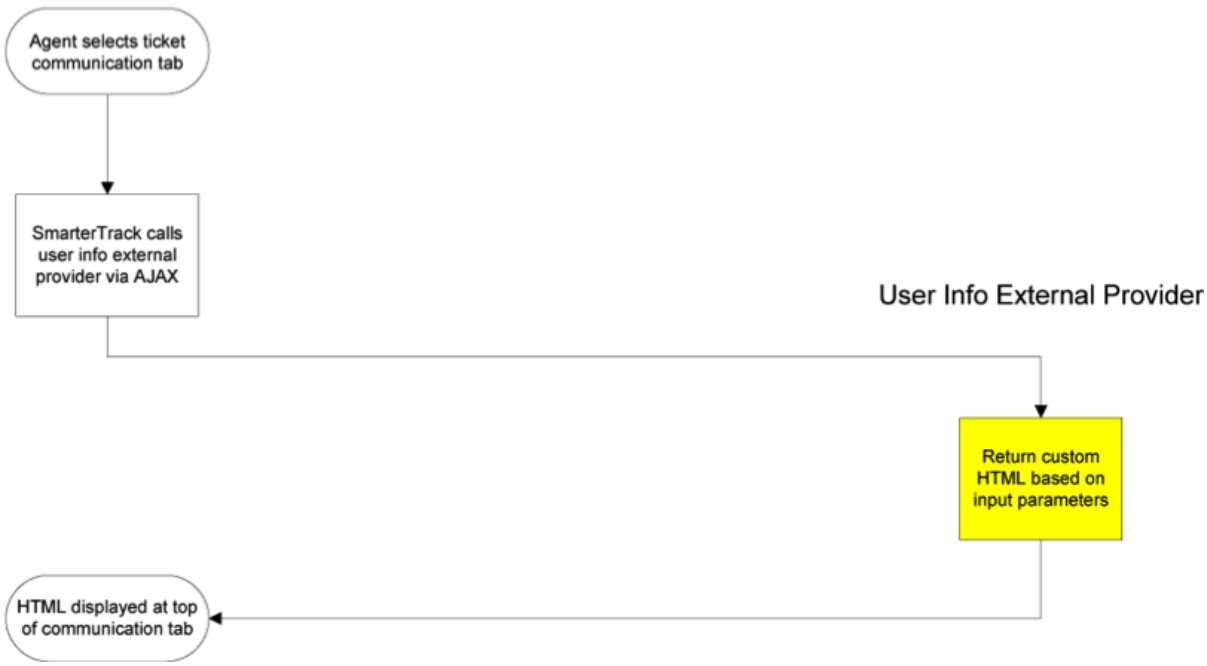
Here is an example of how the StartChat method is used:

```
// This function models the StartTicket function almost exactly. It is provided so you can make the processes run differently // for each. // // Inputs // authPW = Validates that calling software is valid // loginUsername (for chats submitted through Web interface) // emailFrom (for chats submitted through email) // departmentName // cf_* = values of custom fields // Outputs // ChatCreation = TRUE or FALSE // cf_* = output values of custom fields [WebMethod] public ExternalChatProviderResult StartChat(ExternalChatProviderInputs inputs) { ParsedChatInputData iData = new ParsedChatInputData(inputs); ExternalChatProviderResult result = new ExternalChatProviderResult(); // Verify that all necessary criteria to call this function are met if (iData.WebServiceAuthorizationCode != WebServicePassword) return new ExternalChatProviderResult(false,
```

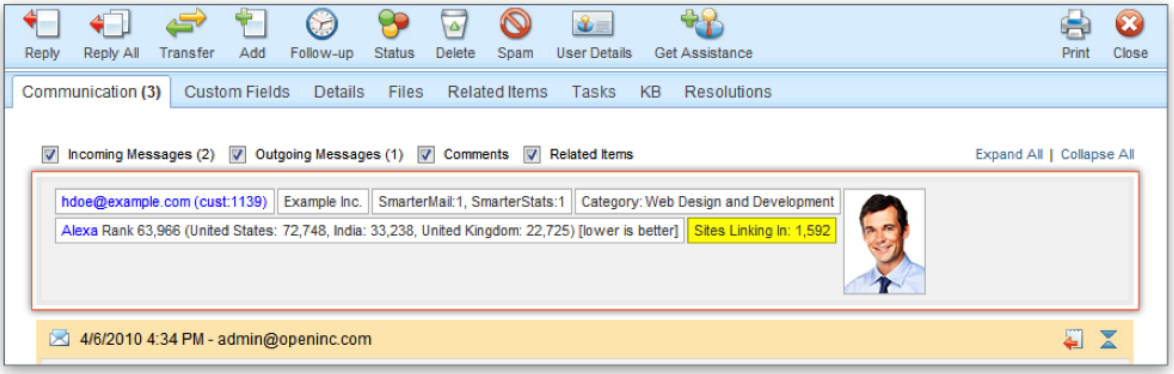
```
"Permission Denied"); // This would be the place to check custom fields and the person's account to see if they can // submit chats. If they can, and you use a pay-per-chat model, deduct the chat from // the person's account here. // For example, to return an error that the person is out of support chats, use the following code // return new ExternalChatProviderResult(true, "There are no more chats in your account.", "ChatCreation=FALSE"); // You can also set output custom fields, which will be filled in with the chat. #region Example: Set the customer's custom fields /* * In this example, we set some hidden custom fields based on the logged in user's information. These * custom fields MUST exist already in SmarterTrack (without the cf_ prefix) * */ //if (iData.LoginUsername.ToLowerInvariant() == "myusername") //{ // result.OutputVariables.Add("cf_Billing ID=4126"); // result.OutputVariables.Add("cf_Plan Type=ASP.NET Semi-dedicated"); //} #endregion result.Success = true; result.Message = "Chat Creation Successful"; result.OutputVariables.Add("ChatCreation=TRUE"); return result; }
```

User Information Provider

The user information provider is used to retrieve custom user data and display it in the communication tab of a ticket. This provider is useful for businesses that use a different database to track customer account information or for businesses that want to display all available information on a particular user. The user information provider utilizes the IExternalUserInfoProvider interface. For a better understanding of how the user information provider works with SmarterTrack, please refer to the flow chart below:



In the example below, the user information provider is querying an external database to retrieve custom user information such as the user's email address, company, number of licenses, and Alexa ranking.



GetUserDetails Method

The GetUserDetails method is used to tell SmarterTrack which additional information should be displayed when a ticket is viewed. This method is called on every ticket view in the management interface. This method works by returning HTML that is displayed at the top of the communication tab in each ticket

- Parameters :

- Inputs :An object containing a generic list of string values. Each object in the list is stored in a specific format (variable=value)

- AuthPW :Used for validating if the calling software is valid
- Username :The customer's username. This is used for ticket's submitted through the Web interface
- Email :The customer's email address.
- BrandName :The brand's name that this ticket is currently assigned to
- DepartmentName :The department's name that this ticket is currently assigned to
- Group :The group's name that this ticket is currently assigned to
- TicketNumber :The identifier for the ticket being viewed
- InteractionType :For the time being this value is always set to "ticket". This provider may be used for other purposes in later versions.
- Message :A message indicating the result of the Web service call
- Success :A Boolean value indicating the result of the Web service call
- OutputVariables :A generic list of strings containing several output parameters needed by SmarterTrack. Some of the output is required in order to make SmarterTrack work correctly with this method. Each object in the list is stored in a specific format (variable=value)

• UserInfoHTML :The HTML that is injected at the top of the communication tab for each ticket. If you do not set this value or if you leave it blank the user information area will not display.

• OutputVariables :An array of other information pertaining to the user. This functionality isn't currently supported.

Here is an example of how the GetUserDetails method is used:

```
// This function is called when a Ticket is viewed in the management interface. // If values are returned for OtherUserInfo a user // information area will appear just above the conversation. In the future, this function may // also be used to retrieve information // for Live Chats and in the users settings page. // // Inputs // authPW = Validates that calling software is valid // username = The username for the account you are retrieving information for // email = The email address for the user you are retrieving information for // brandName = The brand that the Ticket was created in // departmentName = The department that the Ticket was created in // groupName = The group that the Ticket was created in // ticketNumber = the identifier for the Ticket you are retrieving information for // interactionType = Where this call is being made from // Outputs // UserInfoHtml = The HTML that will be displayed just above the conversation of a ticket // OutputVariables = An array of other information pertaining to the user. // This functionality isn't currently supported. [WebMethod] public ExternalUserInfoProviderResult GetUserDetails(ExternalUserInfoProviderInputs inputs) { ParsedUserInfoInputData iData = new ParsedUserInfoInputData(inputs); ExternalUserInfoProviderResult result = new ExternalUserInfoProviderResult(); // Verify that all necessary criteria to call this function are met if (iData.WebServiceAuthorizationCode != WebServicePassword) return new ExternalUserInfoProviderResult(false, "Permission Denied"); // This would be the place to check account information and add custom user information and HTML #region Example: Add a user's account number /* * if (iData.Username.ToLowerInvariant() == "myusername") * { * result.UserInfoHtml = "Account Number: 000-000000-0000 Partner: Yes"; * } */ #endregion result.Success = true; result.Message = "User Account Information Retrieved Successfully"; // result.OutputVariables is unused at this time return result; }
```

Events Provider

The events provider is used to perform custom actions when an event action occurs. SmarterTrack's event system can be configured to fire this provider with details of the event and its conditions. This

provider is useful for businesses that wish to notify another system or database of any action taken in SmarterTrack. The event provider utilizes the IExternalEventProvider interface. For a better understanding of how the event provider works with SmarterTrack, please refer to the flow chart below:

NotifyEventFired Method

The NotifyEventFired method is used to notify another system or database when an event is fired in SmarterTrack. This method is called on any event where the "Notify External Provider" event action is configured. This event action is only available for system level events.

- Parameters :
- Inputs :An object containing a generic list of string values. Each object in the list is stored in a specific format (variable=value)
- AuthPW :Used for validating if the calling software is valid
- EventType :The numeric ID for this event. See section labeled "SmarterTrack Event Type Identifications" for a complete list of numeric IDs.
- Args_[Key] : The argument / condition value that was defined in the event
- Message : A message indicating the result of the Web service call
- Success : A Boolean value indicating the result of the Web service call

Here is an example of how the NotifyEventFired method is used:

```
// This is an example NotifyEventFired function that would be executed when SmarterTrack fires // the Notify External Provider event action. // // Inputs // authPW = Validates that calling software is valid // eventType = Numeric identification for the event that was fired. // args_[key] = The value of an argument passed in by the event system where [key] is the // // name of the argument or condition [WebMethod] public ExternalEventProviderResult NotifyEventFired(ExternalEventProviderInputs inputs) { ParsedEventInputData iData = new ParsedEventInputData(inputs); ExternalEventProviderResult result = new ExternalEventProviderResult(); // Perform custom notification action here... //if (iData.EventType == 501000) //iData.Arguments["title"] ... // Verify that all necessary criteria to call this function are met if (iData.WebServiceAuthorizationCode != WebServicePassword) return new ExternalEventProviderResult(false, "Permission Denied"); result.Success = true; result.Message = "Event Notification Successful"; return result; }
```

SmarterTrack Event Type Identifications

Every event type in SmarterTrack has a numerical identification associated to it that is used when configure the events external provider. When the NotifyEventFired is called, SmarterTrack uses the event identification to specify the action that triggers the event notification.

Event Type	ID
KB Article Created	501000
KB Article Modified	501001
KB Article Flagged For Review	501002
KB Article Stale	501003
KB Article Marked As Reviewed	501004
KB Article Deleted	501005
Ticket Created	502000
Ticket Transferred	502001
Ticket Status Changed	502002
Ticket Priority Changed	502003
Ticket Deleted	502004
Ticket Comment Added	502005
Ticket Message Sent	502006
Ticket Idle	502008
Ticket Count Department	502009
Ticket Count Group	502010
Ticket Count Agent	502011
Ticket Follow Up Scheduled	502012
Ticket Followed Up	502013
Ticket Time Log Created	502014
WhosOn Chat Forced	503000
WhosOn Chat Accepted	503001
WhosOn Chat Invited	503002
WhosOn Chat Invite Rejected	503003
WhosOn Chat Invite Ignored	503004
WhosOn Visitor Removed	503005
WhosOn Visitor Purged	503006
WhosOn Online Activity	503007
Chat Channel Joined	505000
Chat Channel Left	505001
Chat Channel Invited	505002
Survey Answered	506000
Survey Offered	506001
Task Created	507000
Task Modified	507001
Task Deleted	507002
Task Started	507003
Task Due	507004
Task Comment Added	507005
Call Log Created	508000

Call Log Modified	508001
Call Log Deleted	508002
Call Log Attached To Ticket	508003
Call Log Detached From Ticket	508004
Call Log Time Log Created	508005
POP Connection Failed	509000
POP Login Failed	509001
POP Download Failed	509002
POP Import Failed	509003
SMTP Connection Failed	510000
SMTP Login Failed	510001
SMTP Delivery Failed	510002
Live Chat Started	511000
Live Chat Ended	511001
Live Chat Transferred	511002
Live Chat Deleted	511003
Live Chat Attached To Ticket	511004
Live Chat Detached From Ticket	511005
Live Chat Incoming Message	511006
Live Chat Outgoing Message	511007
Live Chat Idle	511008
Live Chat Department Count	511009
Live Chat Group Count	511010
Live Chat Agent Count	511011
Live Chat Comment Added	511012
Live Chat Time Log Created	511013
Agent Ticket Status Changed	512000
Agent Live Chat Status Changed	512001

Connecting SmarterTrack to an External Provider

Although there are several different external providers available for SmarterTrack, they are all configured using the same basic process. Follow these steps to connect SmarterTrack to your third-party Web service:

- Log in to the SmarterTrack management as a system administrator.
- Click the settings icon .
- Expand the System Settings folder.
- Click External Providers in the navigation pane. The external provider settings will load in the content pane.
- Click the Options tab and select the appropriate checkbox to enable the desired external provider.
- Select the appropriate external provider tab: Login, Custom Fields, Ticket, Live Chat, User Information or Events
- In the Web Service URL field, type the URL to the Web service.

- In the Web Service Password field, type the password used to authenticate the Web service.
- Click Save .

Any other settings are optional.

Automation with Web Services

SmarterTrack was built with custom configuration and integration in mind. In addition to being able to customize the look and feel of SmarterTrack, developers and/or system administrators have the ability to code to the SmarterTrack application using Web services. These Web services allow developers and/or system administrators to automate a variety of different actions, giving them the ability to add tickets to SmarterTrack on-the-fly, add notes, and more.

The Automation with Web Services documentation may include services that have not been released to the public yet or are not available in the version you are using.

Note: Web services are intended for use by high-volume and automated businesses environments and hosting companies as they develop procedures to manage their SmarterTrack system and work flow. In addition, this document assumes a basic understanding of Web service technologies and ASP.NET programming.

Integrations

WHMCS SmarterTrack Provisioning Module

Package Description

The WHMCS SmarterTrack module is an open source module developed in PHP that replaces the default ticket and support system within WHMCS with a SmarterTrack help desk. The module can be used with an installed and licensed version of SmarterTrack or with a hosted help desk at SmarterTrack.com. Users' usernames and passwords can be synced between the two platforms and users are able to view, reply to, and close tickets from within the client interface in WHMCS. There is also the ability to initiate live chats from the WHMCS interface as well as a link to knowledgebase articles within SmarterTrack.

Package Goals

The primary goal for the SmarterTrack module is to eliminate the multi-platform maintenance for customer support agents in a business. Agents can maintain support issues inside of SmarterTrack without having to worry about WHMCS at all.

The SmarterTrack module provides the following:

• The ability to override the knowledge base links within WHMCS to redirect to SmarterTrack's knowledge base articles
• The ability to provide live chat support links within WHMCS that will pop up a SmarterTrack live chat within the WHMCS interface
• The ability to override "contact us" email to start a ticket within SmarterTrack rather than an email to a chosen email address. The ticket will also have a different comment added to it that states it was submitted via Sales Acquisition. You also get to specify which department these tickets are automatically submitted to
• Full email ticket support, including:

• Include / exclude departments based on settings within SmarterTrack
• Full custom field support
• A comment added to every ticket submitted to show it was created through WHMCS
• Customizable auto responders with the ability to direct ticket links to WHMCS rather than SmarterTrack
• View all tickets on the support tickets page and all open tickets on client area page
• Restrict users from closing tickets based on settings set in SmarterTrack
• Help phone number and / or email address to be displayed on error pages
• Full email / username and password support. This means that whenever a client changes their email or password in WHMCS, the change will also reflect within SmarterTrack
• A page in the management interface of WHMCS for syncing SmarterTrack and WHMCS users. (SmarterTrack uses the client's email address for the username)
• Full open source module with language string overrides, a custom style sheet and custom client template pages that can all be modified to serve a company's needs

Prerequisites

• Exising installation of WHMCS (version 5.0 and above)
• Existing SmarterTrack installation (version 8.4 or above) or hosted help desk at SmarterTrack.com

Installation and Configuration

Installing the SmarterTrack module is no different than installing any modules within WHMCS. Below are the steps necessary to get a SmarterTrack installation added to WHMCS.

• Extract the SmarterTrack module
• Place the contents in your WHMCS directory under "..modules/addons/smartertrack"
• Navigate to http(s)://your_WHMCS_hostname.com/admin/configaddonmods.php

• Click "Activate" next to the SmarterTrack HelpDesk Module
• Fill out the following properties:

• SmarterTrack Help Desk URL - the base URL for your SmarterTrack instance. For installed versions of SmarterTrack, use whatever URL you set up for accessing your help desk from a Web browser. For hosted customers, use the base URL that contains the SmarterTrack.com domain. (E.g., http://example.smartertrack.com)
• Administrator Name - The administrator username set up in SmarterTrack
• Administrator Password - The password associated with the administrator username
• Override Default KB - Enabling this option will override the default WHMCS knowledg base with the KB system within SmarterTrack. NOTE: When users click on a KB article link, they will be redirected to the SmarterTrack Web portal as the articles are not imported into WHMCS.

• Help Phone Number - If you provide telephone support this optional field will be displayed on any error pages, should errors occur
• Override Pre-Sales 'Contact Us' - Optional override of the 'Contact Us' link. This allows any emails/tickets generated when a user clicks the link to be generated within SmarterTrack versus WHMCS.
• Default Department Name for 'Contact Us' - The department name within SmarterTrack that will be used for 'Contact Us' submissions
• MD5 Hash Enabled - If MD5 has is enabled in WHMCS, then you will need to check this. If MD5 is not enabled, then do not check this option. The biggest reason for this setting is to allow your customers to also lg in to the SmarterTrack portal if they so desire. If MD5 Hash is enabled, then customer passwords will NOT be the same for both platform logins. This is because MD5 is an irreversible hash, so there is no two-way sync. If you need to locate the MD5 has setting in WHMCS, follow these steps:

• Click the Setup tab, the click the first option: General Settings
• In General Settings, click the Security tab
• Towards the bottom of the settings list is the setting named 'Disable MD5 Client's Password.' Note: If this is checked, MD5 is NOT enabled.
• Access Control - These are the roles that will be able to view the Management Interface side of the SmarterTrack WHMCS integration. In addition, this is the area where you will see logging and have the ability to sync users between SmarterTrack and WHMCS.
• Click Save to save your settings. Now, the SmarterTrack integration module is set up and configured.

General Tips

• You will want to disable users from changing their passwords within SmarterTrack. This setting can be found at ‘Settings >> System Settings >> Portal >> Portal Settings’ and on the Options tab, uncheck “Enable users to change passwords”

• You will want to configure your live chat script. The module will work with both image and plain text. There is also an image created just for the this module for live chats. It will take including it in the SmarterTrack images folder and configured in SmarterTrack under “Live Chat Links”

• You will want to configure autoresponders for each department that you use. The ticket link will link to the SmarterTrack Portal by default, so you will want to create your own link for your auto-responders. Your link's target will be the following: *yourdomain*/viewticket.php?tid=#TICKETNUMBER#

Custom Fields for WHMCS

The following custom fields were created specifically to work with SmarterTrack and WHMCS and can be modified and populated with data from WHMCS:

• Email (single line text) - The email address of the WHMCS user

• Display Name (Single Line Text) - The first and last name of the WHMCS user

• Domains (Single Line Text) - In WHMCS, a drop down list that is populated with a user's domains. The selected domain will be sent to SmarterTrack as a single line of text

• Product/Domain (Single Line Text) - In WHMCS, a drop down list that is populated with a user's products and associated domain. The selection will be sent to SmarterTrack as a single line of text

Note: The custom filed display names in SmarterTrack must match these names character for character or else they will not populate correctly.