

# The iService®



## Forms User Guide

Release 5.10

---

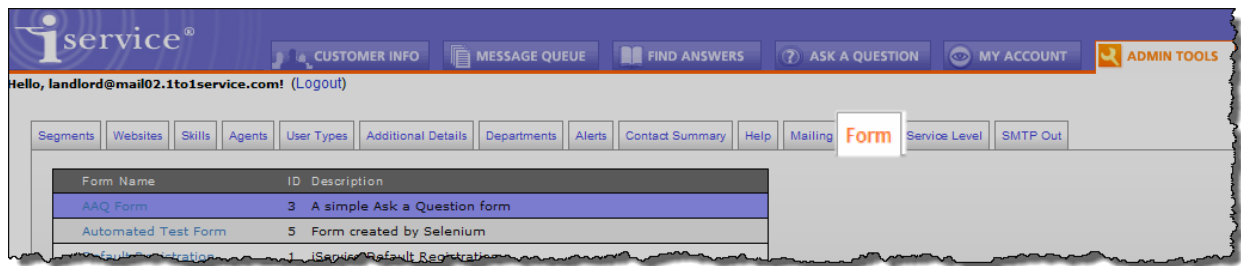
Generating custom forms and templates to interact with iService®



# Table of Contents

<b>OVERVIEW</b>	<b>3</b>
<b>THE ELEMENTS OF A FORM</b>	<b>3</b>
<b>CREATING FORMS</b>	<b>5</b>
<b>ACCESSING FORMS</b>	<b>5</b>
<b>CREATING FORMS USING THE FORM WIZARD</b>	<b>5</b>
<b>FORM VARIABLES</b>	<b>12</b>
<b>VARIABLES USED WITHIN FORM BODY</b>	<b>12</b>
EXAMPLE OF INPUT VARIABLE THAT CAPTURES A NAME	12
EXAMPLE OF INPUT VARIABLE THAT ADDS A CONTACT TO A MASS MAILING LIST AND A MASS MAILING CAMPAIGN	13
<b>ISERVICE VARIABLES USED WITHIN SUBMIT ACTIONS</b>	<b>13</b>
<b>FORM EXAMPLES</b>	<b>14</b>
<b>ISERVICE REGISTRATION (EMBEDDED)</b>	<b>14</b>
<b>CONTACT UPDATE/IMPORT FORMS</b>	<b>16</b>
FORM BODY FOR CONTACT UPDATE/IMPORT	17
FORM ACTION FOR CONTACT UPDATE/IMPORT	17
<b>USING A CONTACT IMPORT FORM TO ADD CONTACTS TO A MAILING LIST</b>	<b>17</b>
FORM BODY FOR CONTACT UPDATE/IMPORT WITH –ADDTOLIST VARIABLE	18
<b>CUSTOM MY ACCOUNT – SUBSCRIPTIONS FORM</b>	<b>18</b>
FORM BODY FOR CUSTOM MY SUBSCRIPTIONS PAGE	19
FORM ACTION FOR CUSTOM MY SUBSCRIPTIONS PAGE	20
<b>FORM ACTIONS</b>	<b>22</b>
<b>ACTIONS SUPPORTED WITHIN ISERVICE FORMS</b>	<b>22</b>
<b>ACTIONS NOT SUPPORTED WITHIN ISERVICE FORMS</b>	<b>22</b>

# Overview



iService Forms are generated from within the Admin Tools section of iService as shown above. The forms interface allows users to very quickly build customized web forms that interact directly with the iService system. These forms can be as simple as an HTML page with no actions, or as complex as a complete question submission and workflow process.

Forms are rendered by the form.aspx page within iService, but may contain any HTML desired so you can fully customize the look of your forms. You can also post to the iService form from your own website or automated process to create interactions within your iService tenant.

## The Elements of a Form

iService forms consist of two elements: the HTML contained with the page (Form Body), and iService actions that the form triggers within iService when the form is submitted (Submit Actions). iService variables can be inserted into your form HTML to interact with iService, such as capturing the phone number of a contact and saving it as a new contact property. These variables are inserted directly into your HTML similar to standard HTML commands.

When the form is loaded, the forms.aspx page will substitute iService variables with the code necessary to build the final, functional page that interacts with iService to complete its actions. The actions available include:

- Lookup a contact and create a new contact if it does not exist
- Submit a ticket (requires agent login credentials to be submitted with form)
- Submit an “ask a question” interaction
- Send an agent email
- Create a public or private note
- Add a contact to a mass mailing list or mass mailing campaign
- Display and allow update to subscriptions to mailing lists, mailing campaigns, and find answers articles (customized My Account – Subscriptions page).

## Intended Users of the Forms Interface

The purpose of the forms interface is to empower business analysts to build various forms that support workflow and customer service needs. The user that creates the form does not need to know anything

about the underlying web services or .NET code involved. They should have a good understanding of the concepts within iService, and will often be your iService administrator. They build the functionality of the form by simply inserting the variables and configuring the form actions using the graphical interface provided by iService.

iService forms are one of the more powerful features of iService and we encourage you to designate a business analyst to this function. Forms can be created much faster than writing your own interface to the iService web services. And future releases of iService will continue to expand the power of this feature.

Once the form body is created, your web or graphic designer can create HTML for forms without having to know anything about the iService system. This separation of “setup” and “design” makes it easy to build professional looking forms very quickly.

# Creating Forms

## Accessing Forms

iService forms are managed from the Admin Tools > Forms page. Each form is assigned a form ID number that is used by the form.aspx page for rendering. You can find the form ID on the form list as shown below.

Form Name	ID	Description
AAQ Form	3	A simple Ask a Question form
Automated Test Form	5	Form created by Selenium

You can access your form using the base URL of your tenant along with the reference to the form number. For example, the following URL is a reference to a form on the 1to1service tenant of iService, where the form ID is 4. The form ID is displayed in the list of forms as shown above.

<https://1to1service.iservicecrm.com/Form.aspx?formID=4>

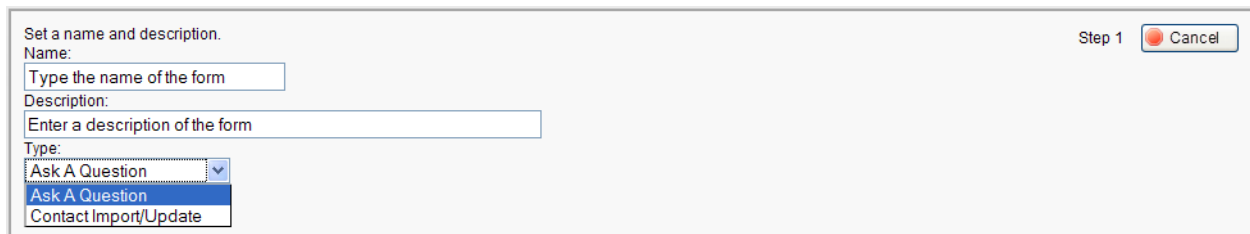
## Creating Forms Using the Form Wizard

The simplest way to create a form is to use the Forms Wizard, but you can also create forms manually or edit existing forms directly from the Forms tab. For most forms used to capture customer questions or import contacts, however, the form wizard is your best option.

There are four steps within the form wizard:

**Step 1.** Name the form, select the type of form, and click next.

The forms wizard can create two types of forms: ask a question and contact import.



**Step 2.** (Ask a Question Form Only) Choose the target topic for the question.



**Step 3.** Choose the Contact Properties to include on the form (name, company, phone, etc.). Email address is always required and is selected by default. If you want the user to confirm their input, check the “Confirm” box. To make the property required, check the “Required” box.

Choose the contact properties: Step 3 of 4

Property (Check to include)	Confirm?	Required?
<input checked="" type="checkbox"/> Email Address	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/> Password	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> First Name	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Middle Initial	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Last Name	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Phone	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Address	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Customer Type	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Company	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Job Title	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Address1	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Address2	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> City	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> State	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Postal Code	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Country	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Available	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Source	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Acct Manager	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Customer Level	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Account Number	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Product Preference	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Supervisor	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Supervisor Email Address	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Lead Source	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Opportunity Stage	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Opportunity Description	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Opportunity Name	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Product	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Sales Person	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Target Close Date	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Opportunity Amount	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Organization	<input type="checkbox"/>	<input type="checkbox"/>

**Step 4.** Choose the Interaction Properties to include on the form (priority, system affected, etc.). The subject and body of the interaction are always required and are selected by default. If you want the user to confirm their input, check the “Confirm” box. To make the property required, check the “Required” box.

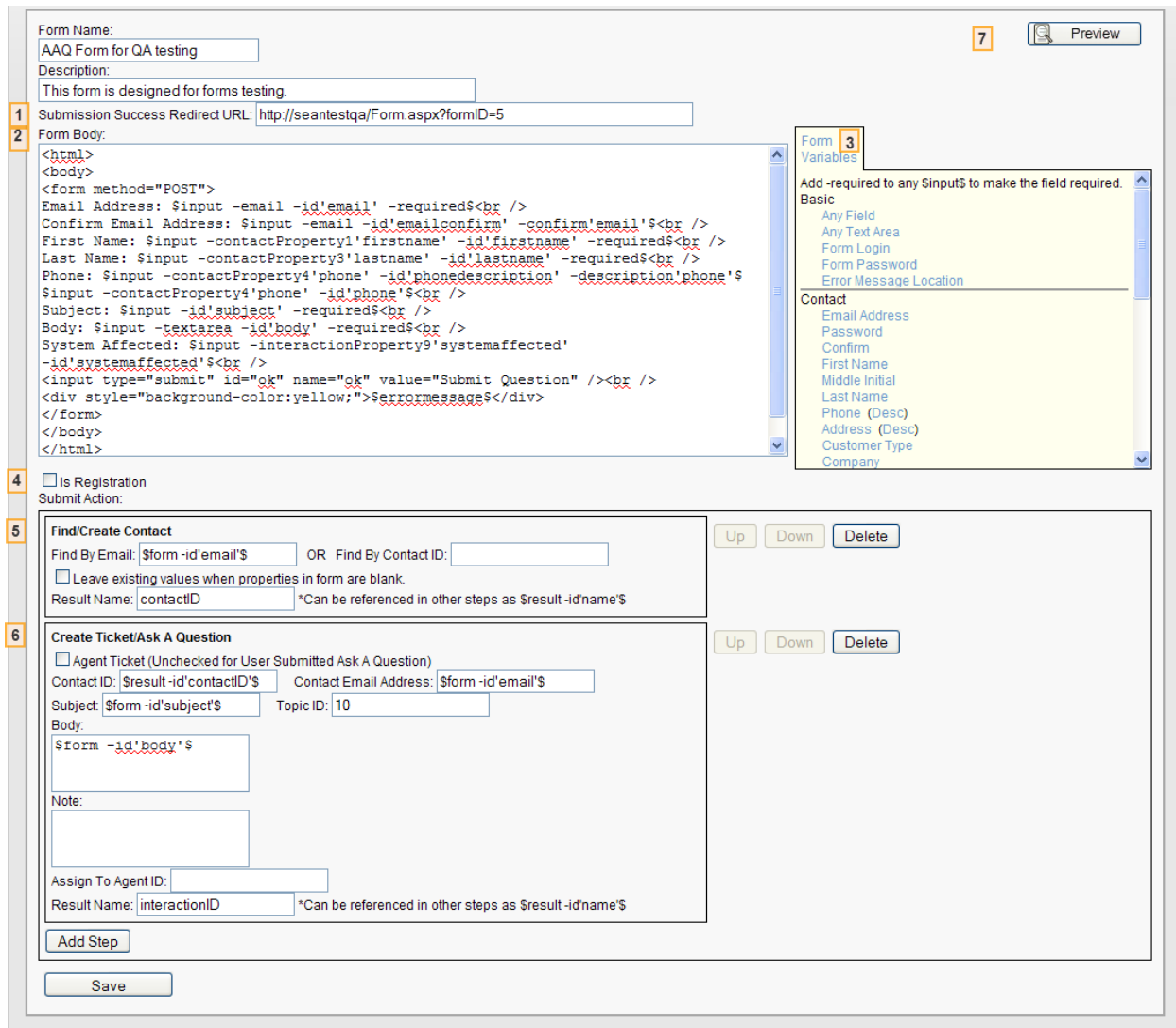
Choose the interaction properties:

Step 4 of 4

Property (Check to include)	Confirm?	Required?
<input checked="" type="checkbox"/> Subject	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> Body	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/> Billable	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Billable Minutes	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Billable MinutesS2	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> BillableS2	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> ChangeStatus1	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> ChangeStatus2	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> QuestionProperty	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Search Terms	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Search Terms Multi-line	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Search Terms Multi-value	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Svc Level 20 Minute	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> System Affected	<input type="checkbox"/>	<input type="checkbox"/>

After the form wizard completes, iService displays the completed form ready for further editing and HTML customization. The actions are automatically populated and generally will not require any changes.

The form generated by the example above is shown below with an explanation for each part of the form.



The screenshot displays the configuration interface for an iService form. It includes several numbered callouts (1-7) pointing to specific fields:

- 1** Submission Success Redirect URL: `http://seantestqa/Form.aspx?formID=5`
- 2** Form Body: A large text area containing HTML code for form fields like Email Address, Confirm Email Address, First Name, Last Name, Phone, Subject, and Body.
- 3** Form Variables: A dropdown menu showing categories like Basic, Contact, and Interaction with various field types.
- 4** Is Registration: A checkbox.
- 5** Find/Create Contact: A section with search criteria and result name fields.
- 6** Create Ticket/Ask A Question: A section with fields for Contact ID, Contact Email Address, Subject, Topic ID, Body, Note, and Assign To Agent ID.

Other visible elements include a 'Preview' button, a 'Save' button, and an 'Add Step' button.

1 – Submission Success Redirect URL - If the form includes a Submit button, the “Submission Success Redirect URL” listed here will be displayed after the form is successfully submitted. This can be any URL, including other iService forms. You must include the full http or https path with the URL. If no URL is specified, the form will reload itself after it is submitted. Therefore, if the form is accessed directly by users we highly recommend including a redirect page.

2 – Form Body - The body of the form is displayed in this text area, and is editable. For larger or more complex forms, most users will edit the form using an HTML editor such as Visual Studio or Microsoft Visual Web Developer.

3 – Form Variables - A variable picker is provided to simplify adding new elements to your form. There are three types of variables: Basic, Contact, and Interaction.

Basic Variables – these are a combination of short cuts for building forms (Any Field, Text Area) and special variables for things like password and error messages.

- Any Field – inserts a field into the form that is not an iService property.
- Any Text Area – inserts a text area into the form that is not an iService property.
- Form Login – if the form requires login before it is submitted (e.g., create Ticket action), this variable is used to capture the contact’s login. This could be either a customer or agent login. This variable is most commonly used when the action creates a Ticket, which requires agent login.
- Form Password – for forms with a Form Login (see above), this is the corresponding password.
- Error Message Location – inserts the variable to display error messages when the form is submitted with an error (e.g., required field not entered).

Contact Variables – these are variables that update iService contact properties.

- Email Address – This is a required field for every form that creates a Ticket or Ask a Question interaction.
- Password – If a new contact is created by the form, this variable allows the contact to specify their password. This is commonly used in the registration pages.
- Confirm – inserts the syntax used to make one of the fields in the form a required field.
- Every contact property defined (standard properties such as First Name plus all custom properties) will also be available from this section of the variable picker.

Interaction Variables – these are variables that update iService interaction properties.

- Subject – This is a required field for every form that creates a Ticket or Ask a Question interaction, and is the subject line of the interaction.
- Body – This is a required field for every form that creates a Ticket or Ask a Question interaction, and is the message body of the interaction.
- Note – This is an optional field for every form that creates a Ticket or Ask a Question interaction, and is the Agent Notes or Private Comments section of the interaction.
- Every interaction property defined will also be available from this section of the variable picker.

4 – IsRegistration - “Is Registration” indicates whether the form will be used as a registration page embedded within the iService application.

***The action section of a form defines the actions taken within iService when the form is submitted. In this example there are two actions: Find/Create Contact and Create Ticket/Ask a Question. These are the most common actions when forms are used to capture customer input. The actions from this example are described below.***

5 – Find/Create Contact - The first submit action, Find/Create Contact, determines whether an account already exists for the customer. If an account does not exist, it creates a new iService contact record. The lookup process can be based on either email/login or iService ContactID. This is described below.

- A. Find By Email – To lookup the contact within iService using email address, populate the ‘Find By Email’ input box. In this example, email is the ID given to the field within the form that captures email address.
- B. Find By Contact ID – To lookup the contact based upon the ContactID from the iService database, this input box would be populated.
- C. “Leave existing values when properties in form are blank” is recommended when the form has optional contact properties. If this box is not checked, blank entries on the form will overwrite existing values that the contact might have within iService for that property. For instance, if an existing contact completes the form but leaves Company blank, the form submission would remove any value the contact already has for company.
- D. Result Name - The ‘Result Name’ field is used to save the created contact ID as a result that can be reference in other actions. In this example, the term ContactID is used. However, other text could be used and reference in actions below.

6 – Create Ticket/Ask a Question - The second action creates the actual Ticket or Ask a Question interaction within iService. The required fields for a new interaction are defined within the action (shown below as \*) and must be properly defined for the form to be successfully submitted. The required and optional fields are described below.

- A. Agent Ticket – If this box is checked, the new interaction will be created as an Agent Ticket instead of an Ask a Question interaction. Agent tickets do not send automated responses acknowledging the new interaction. Only iService agents with access to the Customer Info tab have the ability to create tickets. Therefore, the form body must include the Form Login and Form Password variables that allow and agent to enter their login and password to generate tickets.
- B. Contact ID\* – The interaction must be associated with a contact, which is specified in the first action. The \$result variable is used to look up the result saved by step 5 above.
- C. Contact Email Address\* – Every incoming question must be associated with an email address. The email address is specified in the form body and given an ID, which in this example is named ‘email’.
- D. Subject\* - In this example, the subject line of the message is derived from the form input using the field with an ID of “subject”.
- E. Topic\* - Every interaction must be associated with a Topic. The ID of the target topic is entered here.
- F. Body\* - Every interaction must have a body. In this example, the body is derived from the form input using the field with an ID of ‘body’.
- G. Note –Agent Notes can be added to the new interaction, which will be viewable by agents from within iService. This is an optional field, but in this example is derived from the form input using the field within an ID of ‘note’.
- H. Assign to Agent ID – Tickets can optionally be assigned to a specific agent. The default is for the ticket to be unassigned, but you can enter the ID of the agent here for direct assignment.

- I. Result Name – If a value is entered here, the InteractionID created by the form can be used within additional submit actions. This is optional.

7 – Preview – Click the “Preview” button to load the form in a new window.

## Form Variables

Much of the power within iService forms is based on a set of variables that interact directly with the iService system. These variables are automatically converted into the necessary HTML code for the form. There are two types of variables: those used within the form body and those used within submit actions. Similar to other variables within iService, form variables are enclosed within a \$. For a full list of available variables, use the Form Variables picker.

### Variables Used within Form Body

\$input - the input variable is used to create an HTML input item on a form. This is used to capture a user's input, such as their first name and e-mail address and is the most common variable. The input variable has required and optional parameters.

#### *Example of Input variable that captures a name*

For example, the input variable for first name might look like this:

```
$input -contactProperty1'firstname' -id'firstname' -group'IT' -required$
```

- -contactProperty1'firstname' -- the first parameter in this example is a contact property. The - is used to indicate a new parameter, and the term contactProperty1 is used to indicate the input is a contact property. 'firstname' is an arbitrary name used to make the variable more readable. The reference to 1 in this example is the ID of the property within iService.
- -id'firstname' -- this second parameter is used to identify the input within the form actions section, and as a reference within the forms cascading stylesheet (CSS) for web designers. It is a unique identifier for the input variable and can use any name that you'd like. It should be descriptive of the input (e.g., "firstname").
- -group - the "group" parameter is used on forms that have multiple submit actions. It allows you to designate the action to which the input is related. When you have a form that creates multiple interactions (e.g., two tickets) and do not use the group parameter, the properties will be added to all of the interactions.
- -required - the "required" parameter is optional, and indicates that the user must complete this field before submitting the form. If the form is submitted without a value for this input item, an error message will be displayed.

Requiring the user to confirm their input.

The "confirm" parameter indicates that the user must enter their input twice on the form to confirm they have typed it correctly. This would be common for critical values such as email address. To use the confirm parameter, you create a second Input field and specify the input that is being confirmed. An example for email address is shown below.

```
Email Address: $input -email -id'email' -required$<br />
```

```
Confirm Email Address: $input -email -id'emailconfirm' -confirm'email'$<br />
```

### *Example of Input variable that adds a contact to a mass mailing list and a mass mailing campaign*

Below is an example of adding a contact to a mailing list.

```
$Input -id'List12' -addtolist12'List12'
```

- -id'List12' -- this parameter is used within the form actions section, and as a reference within the forms cascading stylesheet (CSS) for web designer.
- -addtolist12 – the addtolist parameter is used to add the contact to the specified mass mailing list. To add the contact to a mass mailing campaign, use –addtocampaign. The number (e.g., 12 in this example) is the list or campaign ID found on the Mailing tab.

The addtolist and addtocampaign variables generate a checkbox for the user to click. If you don't want the user to be required to click the checkbox, you can hide the input and use javascript to automatically check the box.

## **iService Variables Used within Submit Actions**

**\$form** – The form variable is used within an action to specify the element of the form body that is being reference.

**\$result** – The result variable is used to label the result that is created within an action. This label can then be referenced in the next action that is taken. For instance, when a new contact is created the resulting contactID created by iService can be labeled ContactID or NewContact. Then, the Create Ticket/Ask a Question action can use this to generate a new interaction for that contact.

## Form Examples

The forms interface allows you to create and manage forms of two types: embedded iService forms, and standalone forms. Embedded forms, such as registering as a customer contact within iService, are designed to be used as part of an existing iService page. That is, they only function when loaded within one of the standard iService interface pages (e.g., findanswers.aspx). These embedded forms allow you to modify the standard iService user interface. They are not intended to be viewed from the forms.aspx page.

Most forms are created as standalone forms that are designed to be used on their own. These forms should include a full HTML page and are viewed from the forms.aspx page (e.g., <https://1to1service.iservicecrm.com/Form.aspx?formID=4>).

The functionality of a form is determined by the content of the form body and the submit actions that are used. Forms can include an unlimited number of actions, such as creating multiple tickets, creating multiple contacts, and sending an agent email, all from a single form submission.

### iService Registration (Embedded)

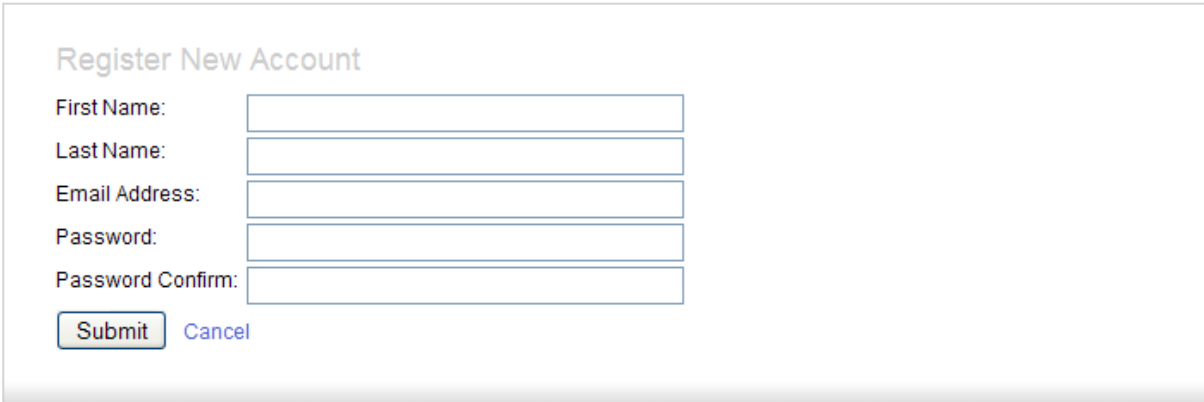
Registered iService users have access to a My Account tab where they can view the history of their interactions with the system, and manage their subscriptions to mailing lists and knowledge base articles. Registration forms are presented to users when they click the “Register new account” link within iService. Since the registration process is integrated directly into the iService website, they do not contain HTML begin and end tags and are unique to the way forms are used.

Each iService Website (Admin Tools > Websites) can have its own registration page, and all forms labeled as registration within the Forms tab are available for selection in the Websites configuration page. When an anonymous user clicks on the Register new account link shown below, the registration page associated with that iService website is presented.

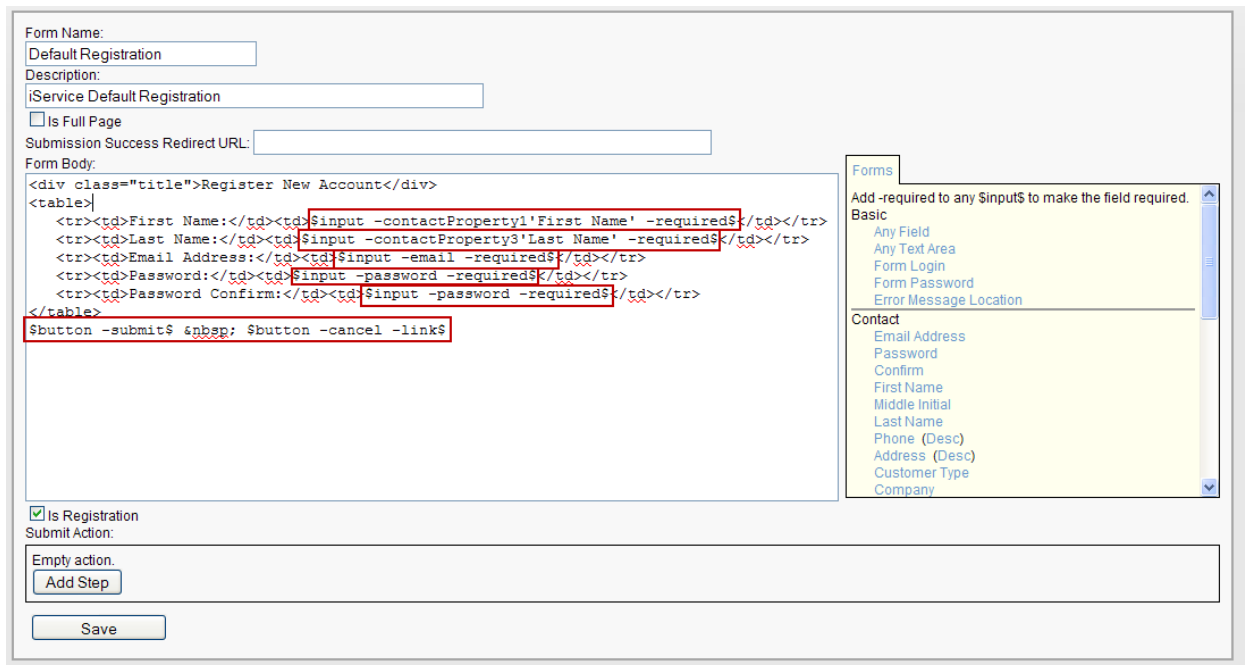


The screenshot shows a login and registration form. It includes two input fields: "Login Email Address:" and "Password:". To the right of the "Password:" field is a "LOGIN" button with a computer icon. Further right are two links: "forgot password »" and "Register new account »".

Registration forms are designed to perform two functions: create a new contact in iService, and save contact property values for the new contact. The default registration form captures the user's email address, first Name, last name, and desired password. This can be modified within the Admin Tools > Forms page to include any contact property available to customers. The standard registration page is shown below.



To revise the registration page, you can either edit the default registration page included with iService, or create a new registration page altogether. The registration page does not include Submit Action steps because it is embedded within the iService system itself. The web services used to create the contact and populate its contact properties are managed automatically by iService.



Form Name:

Description:

Is Full Page

Submission Success Redirect URL:

Form Body:

```

<div class="title">Register New Account</div>
<table>
  <tr><td>First Name:</td><td>$input -contactProperty1'First Name' -required$/td></tr>
  <tr><td>Last Name:</td><td>$input -contactProperty3'Last Name' -required$/td></tr>
  <tr><td>Email Address:</td><td>$input -email -required$/td></tr>
  <tr><td>Password:</td><td>$input -password -required$/td></tr>
  <tr><td>Password Confirm:</td><td>$input -password -required$/td></tr>
</table>
$button -submit$ &nbsp;&nbsp;&nbsp; $button -cancel -link$

```

Is Registration

Submit Action:

Forms

Add -required to any \$input\$ to make the field required.

Basic

- Any Field
- Any Text Area
- Form Login
- Form Password
- Error Message Location

Contact

- Email Address
- Password
- Confirm
- First Name
- Middle Initial
- Last Name
- Phone (Desc)
- Address (Desc)
- Customer Type
- Company

The form variables within the standard registration page are outlined above in red. All properties are required, as indicated by the ‘-required’ parameter within each variable. The Submit and Cancel buttons are outlined in red at the bottom of the form. Including the -link parameter within the button command causes the form to display that button as a link, rather than a standard button.

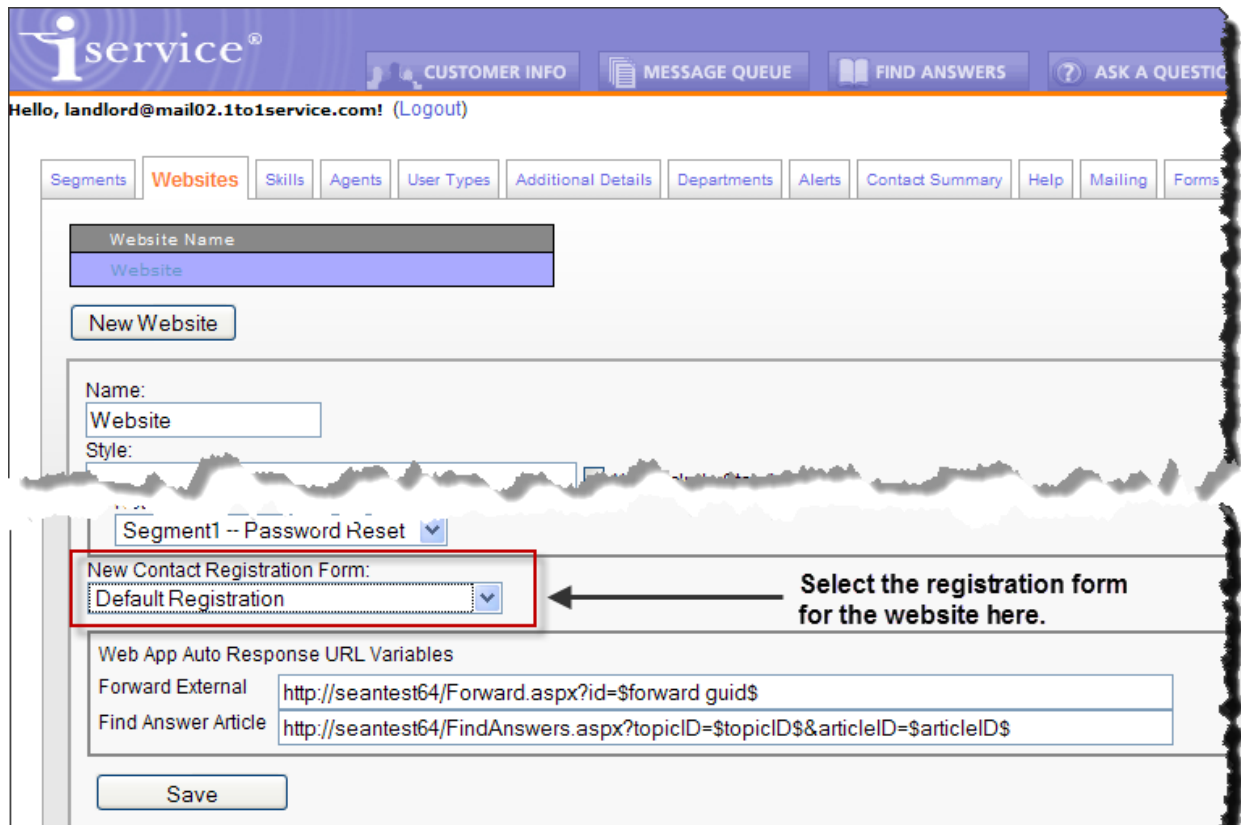
Registration forms are identified based upon the Is Registration checkbox below the form body. If this box is checked, the form will be available for selection in the Admin Tools > Websites page. When the Is Registration box is checked, the Preview link on the form is removed since the page is not meant to be displayed as a standard web page. If a user attempts to load a form that is labeled as registration from

the form.aspx page, they will receive an “Invalid Form ID Specified” error. If the box is unchecked, iService will assume the page is meant to be displayed as a standard web page.

Registration forms use a special iService variable named “button”. This variable generates the “submit” and “cancel” buttons that are unique to the registration process. The –link parameter is optional and converts the button to a hyperlink. The form of this variable is as follows:

```
$button -submit$      $button -cancel -link$
```

To test registration forms, you can create a test iService website from Admin Tools>Websites, and select the new registration form.



For additional information regarding creating and managing iService websites, see the iService Setup Guide.

## Contact Update/Import Forms

The iService forms interface includes a wizard for creating contact update/import forms. These forms can be used to manually add or update contact information, or with the iService batch forms update utility to import contacts from a.CSV file.

The Wizard allows you to select any contact property defined for your tenant, and automatically creates the form body and action required for the form. The form body generated will include an input variable

for each contact property, and a single action (Find/Create Contact) for generating or updating a contact based upon the information entered into the form.

If the form is going to be used manually, you should add a “Submission Success Redirect URL” that is displayed when the form submits successfully.

### Form Body for Contact Update/Import

An example of a completed form body for a contact update/import form is shown below.

```
<html>
<body>
<form method="POST">
```

```
Email Address: $input -email -id'email' -required$<br />
First Name: $input -contactProperty1'firstname' -id'firstname'$<br />
Last Name: $input -contactProperty3'lastname' -id'lastname'$<br />
Company: $input -contactProperty7'company' -id'company'$<br />
Address1: $input -contactProperty15'address1' -id'address1'$<br />
Address2: $input -contactProperty16'address2' -id'address2'$<br />
City: $input -contactProperty17'city' -id'city'$<br />
State: $input -contactProperty18'state' -id'state'$<br />
Postal Code: $input -contactProperty19'postalcode' -id'postalcode'$<br />
Customer Type: $input -contactProperty6'Customer Type' -id'customertype'$<br />
```

```
<input type="submit" id="ok" name="ok" value="Create/Update Contact" /><br />
<div style="background-color:yellow;">$errorMessage$</div>
</form>
</body>
</html>
```

### Form Action for Contact Update/Import

An example of the “Submit Action” used with this form is shown below.

Submit Action:

**Find/Create Contact**

Find By Email:  OR Find By Contact ID:

Property Group:  \*Specified with \$input ... -group'xx'\$

Leave existing values when properties in form are blank.

Result Name:  \*Can be referenced in other steps as \$result -id'name'\$

## Using a Contact Import Form to Add Contacts to a Mailing List

You can modify a contact import form to add each contact to a mailing list. This requires you to add the “- addtolist” or “addtocampaign” variable in the form body. Then, to your input file that includes the value “true” for each contact and needs to be added to the list or campaign.

An example of a completed “-addtolist” variable in the form body is shown below. In the example below, each contact is added to the mailing list whose ID is 1. The name of the list that is shown below

(CustomerList) can be anything and is used to make the form easier to understand. We suggest using the name of the list as specified in the Mailing List page. The full set of lists and campaigns available are shown in the Form Variable picker on the forms page.

### Form Body for Contact Update/Import with *-addtolist* variable

The full form body for this example is shown below, with the changes highlighted. There are note changes required to the form action.

```
<html>
<body>
<form method="POST">
```

```
Email Address: $input -email -id'email' -required$<br />
First Name: $input -contactProperty1'firstname' -id'firstname'$<br />
Last Name: $input -contactProperty3'lastname' -id'lastname'$<br />
Company: $input -contactProperty7'company' -id'company'$<br />
Address1: $input -contactProperty15'address1' -id'address1'$<br />
Address2: $input -contactProperty16'address2' -id'address2'$<br />
City: $input -contactProperty17'city' -id'city'$<br />
State: $input -contactProperty18'state' -id'state'$<br />
Postal Code: $input -contactProperty19'postalcode' -id'postalcode'$<br />
Customer Type: $input -contactProperty6'Customer Type' -id'customertype'$<br />
Click to Join: $input -id'List' -addtolist1'CustomerList'$<br />
```

```
<input type="submit" id="ok" name="ok" value="Create/Update Contact" /><br />

<div style="background-color:yellow;">$errorMessage$</div>
</form>
</body>
</html>
```

## Custom My Account – Subscriptions Form

If you have integrated iService mailing lists into your website, you might want to provide a customized e-mail preferences page where your customers can manage their mailing list subscriptions. This form uses the following variables in the form body to generate a custom subscription management page.

`$if -myaccountlists$` - This variable checks to see if any lists are available for display for the customer. If there are lists available, the variable `$repeat -myaccountlists$` is used to generate a table that displays all of the lists and their subscription status. If there are no lists, it will move to the `$else $` variable.

`$repeat -myaccountlists$` - The repeat variable tells iService to create a record for each list available and to display the fields specified between “repeat” and “endrepeat”.

`$myaccountlist -name$` - displays the list name.

`$myaccountlist -description$` - displays the list description.

`$myaccountlist -subscribe -id'mailinglist'$` - displays a check box for the list.

**\$endrepeat\$** - indicates the end of the fields to be displayed for each list.

**\$else\$** - The else variable is used when there are no lists available. If there are no lists available, the form will display the text that follows after **\$else\$**

**\$endif\$** - The endif variable marks the end of the account list table.

The process is identical for campaigns and find answers subscriptions, using myaccountcampaigns and myaccountarticle instead of myaccountlist.

You can use the **\$link myaccount\$** variable inside of a mass mailing to direct recipients of a mass mailing message directly to this custom subscriptions page, and they will not be required to login to view their settings.

### *Form Body for Custom My Subscriptions Page*

An example of a completed form body for a custom my subscriptions form is shown below. The iService variables are shown in **bold**.

```
<html>
<head>
</head>
<body>
<form method="POST" enctype="multipart/form-data">
<h2>Update your subscriptions.</h2>

$if -myaccountlists$
Available Mailing Lists:<br />
<table>
<tr><th>Mailing List Name</th><th>Description</th><th>Subscribed</th></tr>
$repeat -myaccountlists$
<tr><td>$myaccountlist -name$</td>
<td>$myaccountlist -description$</td>
<td>$myaccountlist -subscribe -id'mailinglist'$</td></tr>
$endrepeat$
</table>
$else$
There are no mailing lists.<br />
$endif$

<br />
$if -myaccountcampaigns$
Available campaigns:<br />
<table>
<tr><th>Campaign Name</th><th>Description</th><th>Subscribed</th></tr>
$repeat -myaccountcampaigns$
<tr><td>$myaccountcampaign -name$</td>
```

```

<td>$myaccountcampaign -description$</td>
<td>$myaccountcampaign -subscribe -id'campaign'$</td></tr>
$endrepeat$
</table>
$else$
There are no campaigns.<br />
$endif$

<br />
$if -myaccountarticles$
You are subscribed to these articles:<br />
<table>
<tr><th>Article Subject</th><th>Topic</th><th>Subscribed</th></tr>
$repeat -myaccountarticles$
<tr><td> bold that </td>
<td>$myaccountarticle -subject$</td>
<td>$myaccountarticle -subscribe -id'article'$</td></tr>
$endrepeat$
</table>
$endif$
<br />
<input type="submit" id="ok" name="ok" value="Save Subscriptions" /><br />
<div style="background-color:yellow;">$errorMessage$</div>

$if -submitsuccess$
<div><h3>Your subscription settings have been saved.</h3></div>
$endif$

</form>
</body>
</html>

```

*Form Action for Custom My Subscriptions Page*

The only action required on a subscription page is `_Form My Account Subscriptions`.

Submit Action:

<p><b>My Account Subscriptions</b>          Subscriptions will be set for logged in user or user targeted with \$link myaccount\$.</p>	<input type="button" value="Up"/> <input type="button" value="Down"/> <input type="button" value="Delete"/>
<input type="button" value="Add Step"/>	



## Form Actions

The forms interface uses the same action object as iService **Filters** and **Alerts**. However, some action steps are dependent upon input, such as a regular expression match within an iService filter. Therefore, not all actions are appropriate in the context of a form submission.

### Actions Supported within iService Forms

- Form-Find/Create Contact – This action is used to capture contact information from the user's input and update iService. It can be used independently to capture and update contact information, or in conjunction with the Create Ticket/Ask a Question action. For example, before you create a Ticket you must identify or create the contact for which you are creating the ticket.
- Form-Create Ticket/Ask a Question – This action creates a new interaction and requires the Find/Create Contact action to be run first. Forms that create tickets require an agent login, but only agents are authorized to use the Customer Info-Contact-Create Ticket functionality.
- Form-Create Note – This action can create either a Public or Private note for the selected contact.
- Form-Create Agent Email – This action sends an Agent Email to the selected contact. All of the options available from the Customer Info – Contact – Agent Email tab are available (send secure, expect reply, etc.), plus the message can be added to an existing thread by use of the Parent Interaction ID parameter.

### Actions Not Supported within iService Forms

The following actions are not currently supported from the iService Forms interface. Adding these actions to a form will have no effect on the form submission and will simply be ignored. Only those actions with names preceded by Form are available for use.

- Change Topic
- Filter
- Resolve Interaction
- Select Agents
- Select Auto Responses
- Set Interaction Property
- Change Interaction Parent
- Forward External