PureConnect®

2023 R3

Generated:

09-November-2023

Content last updated:

21-February-2023

See Change Log for summary of changes.

GENESYS™

CIC Web Applications

Installation and Configuration Guide

Abstract

This installation and configuration content describes configuration settings needed to install CIC Web Applications on IIS, Apache, or Nginx servers. It also includes information about load balancing and troubleshooting, and enabling HTTPS.

For the latest version of this document, see the PureConnect Documentation Library at: http://help.genesys.com/pureconnect.

Table of Contents

About the CIC Web Applications Installation and Configuration Guide

The *CIC Web Applications Installation and Configuration Guide* describes the configuration settings needed to install CIC Web Applications on IIS, Apache, or Nginx web servers. It assumes you are installing all of the CIC Web Applications at once. However, you can install individual applications by placing the appropriate application folders in the CIC Web Applications folder created following the process described in the guide.

These installation instructions are for a first-time installation of the CIC Web Applications. For information about applying upgrades, see Application updates (subsequent installs).

Download the CIC Web Applications from the PureConnect Product Information site at https://my.inin.com/products/Pages/Downloads.aspx.

Note: CIC Web Applications 2015 R2 and later contain the IPA Stand-alone Web Client, Optimizer Web Features, Interaction Connect, and CX Insights web app.

Web server requirements

You can install CIC Web Applications on any of the following web servers:

- **Apache 2.4**
  Supported with Windows Server 2012 R2 and higher
  Apache may work with Linux OS, but this combination is not tested or supported by Genesys.
  If installing on Windows, you can find 2.4.x versions at http://www.apachehaus.com/cgi-bin/download.plx or http://www.apachelounge.com/download/
- **IIS 8.5** (Verified with Windows Server 2012 R2)
  IIS 10 (Verified with Windows Server 2016 on CIC 2018R2 and higher)
  - Application Request Routing extension (https://www.microsoft.com/en-us/download/details.aspx?id=47333)
  - URL Rewrite extension (https://www.microsoft.com/en-us/download/details.aspx?id=47337)
- **Nginx**
  Supported with Windows Server 2012 R2 and higher
  Nginx may work with Linux OS, but this combination is not tested or supported by Genesys.
  No additional modules are necessary.

Transport Layer Security

The recommended setup for CIC Web Applications uses a separate web server and Customer Interaction Center server. In this setup, the web server is used as a reverse proxy to communicate with Interaction Center Web Services (ICWS) on a Customer Interaction Center server.

There are two legs of communication you can encrypt in a CIC web applications deployment.

Configuration of encryption for each leg is independent of configuration for the other leg: you can encrypt either, both, or neither leg of traffic.

- The **front-end connection** between the web browser and the web server
  To avoid credential theft, the recommended deployment is to encrypt this connection.
- The **back-end connection** between the web server and the CIC server
  If your web server and CIC server(s) are in the same secured environment, the recommended deployment is to use HTTP between the web server and CIC to reduce load on the CIC server.
  However, you can also secure the connection between the web server and the CIC server. In order to do so, use the correct HTTPS port noted in the instructions for your server.

To enable encryption, see the instructions for your server type:

- **IIS:** Enable HTTPS between the web browser and IIS and Enable HTTPS between IIS and CIC
- **Apache:** Enable HTTPS between the web browser and Apache and Enable HTTPS between Apache and CIC
- **Nginx:** Enable HTTPS between the web browser and Nginx and Enable HTTPS between Nginx and CIC

Parent directories

You host CIC Web Applications at the root of web servers. However, if there are additional parent directories, the rewrite rules will also work.

The default setup *application* is located at *http://webServer/application*, but the rewrite rules will continue to work if your URLs need to follow a pattern such as *http://webServer/ININWebApps/application*. Note that in this scenario, you may need to modify some of the configuration file entries to avoid affecting non-CIC web applications.

If a parent directory is named `api`, modify the URL rewrite rules as needed.

Prerequisites

- No reverse proxies should exist in front of the one created for CIC Web Applications. If there are multiple reverse proxies in your environment, it is crucial that the same rewrite rules for HTTP headers X-Forwarded-For and ININ-ICWS-Original-URL are also set up in each reverse proxy. At minimum, they must be set up in the reverse proxy that client requests hit first.
- If you use the same reverse proxy for CIC Web Applications and Genesys Intelligent Automation, then the Intelligent Automation rewrite rule must come **before** the CIC Web Application rule. For more information, see the PureConnect Integration with Genesys Intelligent Automation Technical Reference.
- Forward proxies must not change the "Host" header of clients' requests, or CIC Web Applications may not be fully functional.

Before beginning the installation process, verify that

- Your web server is accessible from client computers. Make firewall modifications as needed.
- Your CIC server (and any OSSMs) are accessible from your web server. Make firewall modifications as needed: the web server will communicate with ICWS over port 8018 (or port 8019 if you are securing each ICWS endpoint with certificates).
- The appropriate level of security is set up on your web server. Genesys requires that your front-end web server is at least secured via TLS.
- Your web server(s) (and any proxies) are configured to forward any HTTP header without modifying it. A primary example is **X-Forwarded-For**, which is unchanged by default in IIS, Apache, and Nginx.

Installation Overview for IIS, Apache, and Nginx

For all server types, download and unzip the CIC Web Applications files and move the files to your web server's document root.

The remaining instructions for installing CIC Web Applications vary depending on your web server software. Although the details vary, the general steps are as follows:

- Install and Configure your Web Server Software (IIS, Apache or Nginx).
- Create the application pool that will be used by the new CIC Web Application Site. (IIS only).
- Create the CIC Web Application site.
- Create the rewrite rules that will be used for this site. This section will include adding the rules, the required server variables, and the rewrite map.
- Configure the required MIME types for the site. This will ensure that all the necessary files for the applications will be loaded.
- Restart the web server and test the new site.

Note: You must use the latest version of the CIC Web Applications, even if your main CIC server is at a previous release level.

Security Considerations

Genesys makes the following recommendations to improve security in our web-based applications. We recommend you implement these methods of securing your system.

Cross-Frame Scripting or clickjacking

Cross-Frame Scripting (XFS) vulnerability can allow an attacker to load the vulnerable application inside an HTML iframe tag on a malicious page. Clickjacking is an attack that occurs when an attacker uses a transparent iframe in a window to trick a user into clicking on a Call to Action (CTA), such as a button or link, to another server in which they have an identical looking window. The attacker in a sense hijacks the clicks meant for the original server and sends them to the other server. This is an attack on both the visitor themselves and on the server.

To prevent cross-frame scripting or clickjacking, set the **X-Frame-Options** value to one of the following in your web server configuration file:

- **DENY**: Deny all attempts to frame the page.
- **SAMEORIGIN**: The page can be framed by another page only if it belongs to the same origin as the page being framed.
- **ALLOW-FROM** *origin*: Developers can specify a list of trusted origins in the *origin* attribute. Only pages on origin are permitted to load this page inside an iframe.

Note: The server parameter **X-Frame-Options** can specify the value of the **X-Frame-Options** header for ICWS responses. You can use the expected values of **DENY** or **SAMEORIGIN** or set to **NONE** to indicate that the header value should not exist at all.

See the instructions and examples for IIS, Apache, and Nginx.

Prevent directory scan

Directories that are restricted to certain users can be discovered during a directory scan. Risks associated with an attacker discovering a directory on your application server depend on the type of directory and the types of files it contains. The primary threat, other than accessing files containing sensitive information, is that an attacker can use the information to perform other types of attacks. Restrict access to important directories or files by adopting a "need to know" requirements before granting access for both the document and the server root. Turn off features such as Automatic Directory Listing that can be used by an attacker in formulating or conducting an attack.

Note: In addition to the recommended configuration changes, you should also assign the appropriate ACLs (Access Control Lists) to the directories on the web server used by your organization.

- For IIS, to prevent an attacker from executing a Directory Enumeration scan, set **directoryBrowse** to false in your web server configuration file. For instructions, see disable directory listing.
- For Nginx, set **autoindex** to off. You can apply the rule to all directories or specify the directory names you do not want listed. For instructions, see block all or some directories from directory scanning.
- For Apache, set DirectoryMatch Options to FollowSymLinks and create a .htaccess file in the related application directory. For instructions, see disable directory scanning.

File Uploads

Interaction Connect users are blocked from including the following types of executable files (exe, .sh, and .js) when creating Personal Responses or using Response Management in an email interaction. For more information, see Create Personal Responses in the Interaction Connect help.

Interaction Connect does not examine email interactions or file attachments for malicious intent. It is your responsibility to configure your email server to block attacks.

- Adopt a strict file upload policy that prevents malicious material from being uploaded via sanitization and filtering. Limit the types of files that can be uploaded.
- Restrict web users from accessing the server directory.

Enable Strict Transport Security

HTTP Strict Transport Security (HSTS) is a policy mechanism that helps to protect websites against man-in-the-middle attacks such as protocol downgrade attacks and cookie hijacking. It allows web servers to declare that web browsers should

automatically interact with web servers only through HTTPS connections, which provide Transport Layer Security (TLS/SSL), unlike the insecure HTTP used alone.

The HSTS Policy is communicated by the server to the user agent via an HTTP response header field named "Strict-Transport-Security". HSTS Policy specifies a period during which the user agent should only access the server in a secure fashion. Websites using HSTS often do not accept clear text HTTP, either by rejecting connections over HTTP or systematically redirecting users to HTTPS.

To add the HSTS header in all the responses, the following header details need to be configured in webserver. This will enable 'HSTS' to all responses that the server sends.

'Strict-Transport-Security': 'max-age=31536000; includeSubDomains; preload;'

Configuring HSTS header in IIS / NGINX / Apache

- To configure HSTS header in **IIS**, open HTTP response header and add the following custom header in web.config file.

```
<system.webServer>
  ...
      <httpProtocol>
          <customHeaders>
              <add name="Strict-Transport-Security" value="max-age=31536000; includeSubDomains; preload"/>
          </customHeaders>
      </httpProtocol>
  ...
</system.webServer>
```

- To set the HSTS header in **NGINX**, include an 'add_header line' in your server block:

```
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload" always;
```

- To add the HSTS header in **Apache**, include the Header given below to the appropriate VirtualHost section:

```
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload”
```

Install CIC Web Applications on Microsoft IIS

For a basic working installation, such as for a test environment, complete the following steps:

- Step 1: Add Required IIS Services
- Step 2: Download and Copy CIC Web Applications Files
- Step 3: Configure IIS

For a production environment, you can also follow the instructions in the following:

- Enable HTTPS between the web browser and IIS
- Enable HTTPS between IIS and CIC
- Application Configuration.

Step 1: Add Required IIS Services

For information about all the installation steps, see Install CIC Web Applications on Microsoft IIS.

To add required IIS services

1. In Server Manager, verify that the Web Server Role (IIS 7) is added with the following (minimum required) role services installed:
   Common HTTP Features
   - Static Content
   - Default Document

   Performance
   - Static Content Compression

   Security
   - Request Filtering

   Management Tools
   - IIS Management Console
2. If you have not installed the Application Request Routing and URL Rewrite extensions, install them.
   - Application Request Routing extension (https://www.microsoft.com/en-us/download/details.aspx?id=47333)
   - URL Rewrite extension (https://www.iis.net/downloads/microsoft/url-rewrite#additionalDownloads)
3. Enable server as proxy and enable response buffering:
   a. In **IIS Manager**, click your server.
   b. Double-click the **Application Request Routing Cache** module.
   c. In the Actions pane, click **Server Proxy Settings**.
   d. Check **Enable proxy**.
   e. Change the **Response buffer threshold (KB)** setting under "Buffer Setting" to **0**.
   f. Click **Apply**.
4. Verify that `index.html` and `index.htm` are present as Default Documents.

Step 2: Download and Copy CIC Web Applications Files

For information about all the installation steps, see Install CIC Web Applications on Microsoft IIS.

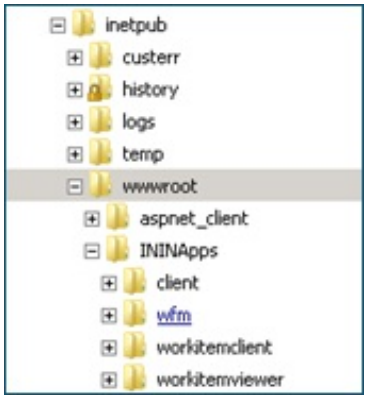
To download and copy CIC Web Applications files

1. In Windows Explorer, create a directory in the Home Directory in IIS for the CIC Web Applications. In a default IIS installation, the Home Directory is `C:\\inetpub\wwwroot`. Verify that IIS has the appropriate permissions for that newly created directory.

   > **Note:** In this example, the directory is `ININApps`.

2. Download the CIC Web Applications `zip` file from https://my.inin.com/products/Pages/Downloads.aspx. All the web applications are contained in this single `zip`.
3. Unzip the `CIC Web Applications` folder.
4. Navigate to the `web_files` folder inside the unzipped `CIC Web Applications` folder.
5. Copy all of the folders inside of `web_files`. Each folder contains one CIC web application.
6. Paste the folders copied in the previous step into the directory you created in step 1. Doing so places the appropriate directory structure and files for all CIC Web Applications on your web server.

   Following is an example directory structure for CIC Web Applications in IIS.

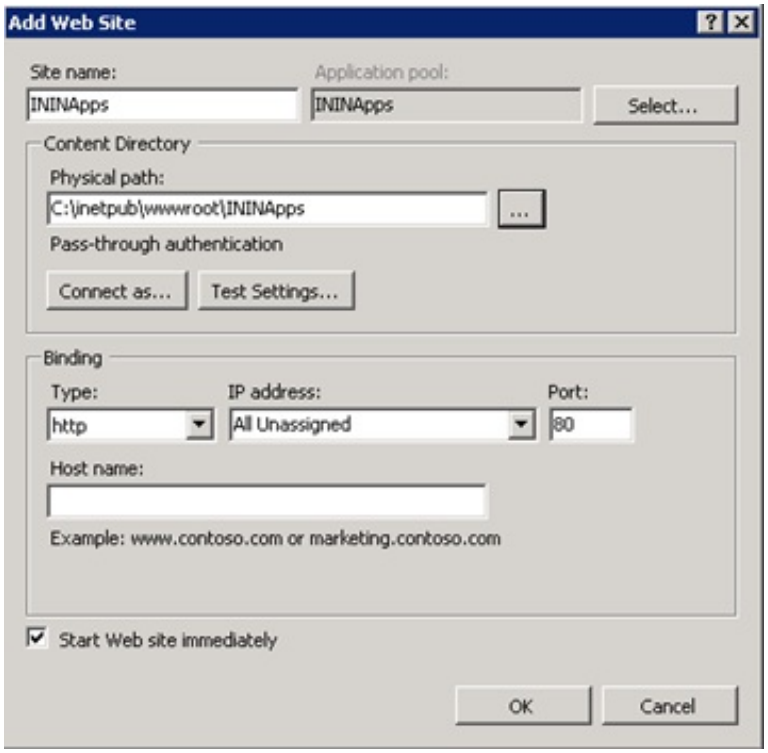


Step 3: Configure IIS

For information about all the installation steps, see Install CIC Web Applications on Microsoft IIS. See also Appendix A: IIS XML Configuration.

To configure IIS

1. Create a new Site named **ININApps** in IIS:
   a. Right-click on **Sites** and choose **Add web site**.
   b. In the dialog box, set the **Content Directory Physical path** to the CIC Web Applications folder you previously created in your server's Home Directory.



2. Remove the .NET Framework version of the application pool:
   a. In the IIS Manager side pane, click **Application Pools**.
   b. Right-click the newly created ININApps application pool.
   c. Click **Basic Settings**.
   d. Change the .NET Framework version to "No Managed Code."
   e. Click **OK**.
3. Enable static content compression on the new Site:
   a. Click the site in **IIS Manager**.
   b. Double-click the **Compression** module.
   c. Check **Enable static content compression**.
   d. Click **Apply**.
4. Update the maximum URL and query string size in Request Filtering, if enabled:
   a. Click the site in the **IIS Manager**.
   b. Double-click on the **Request Filtering** module, if enabled. If the module doesn't appear, Request Filtering is not enabled.
   c. Select the **URL** tab in the Request Filtering view.
   d. Click on **Edit Feature Settings** in the Actions pane.
      i. Update **Maximum URL Length (bytes)** to "8192".
      ii. Update **Maximum Query String (bytes)** to "8192".
      iii. Update **Maximum allowed content length (bytes)** to something greater than or equal to "20971520".
   e. Click **OK**.
5. Add allowed server variables:

   > **Note:** Steps 6 through 10 can alternatively be completed using XML configuration file. See Appendix A for XML configuration.

   a. Click the site in the **IIS Manager**.
   b. Double-click on the **URL Rewrite** module.
   c. In the Actions pane, click **View Server Variables**.
   d. Create the following three server variables by clicking **Add** in the Actions pane.
      - **WEB_APP**
      - **ICWS_HOST**
      - **HTTP_ININ-ICWS-Original-URL**
6. Create the rewrite map.
   a. Click the site in the **IIS Manager**.

b. Double-click the **URL Rewrite** module.
c. In the Actions pane on the right, click **View Rewrite Maps**.
d. Click **Add Rewrite Map**.
e. Enter **MapScheme** for the rewrite map name.
f. In the Actions pane, click **Add Mapping Entry**.
g. Type the following:

Original value:	New value:
on	https

h. Repeat steps f and g with the following information:

Original value:	New value:
off	http

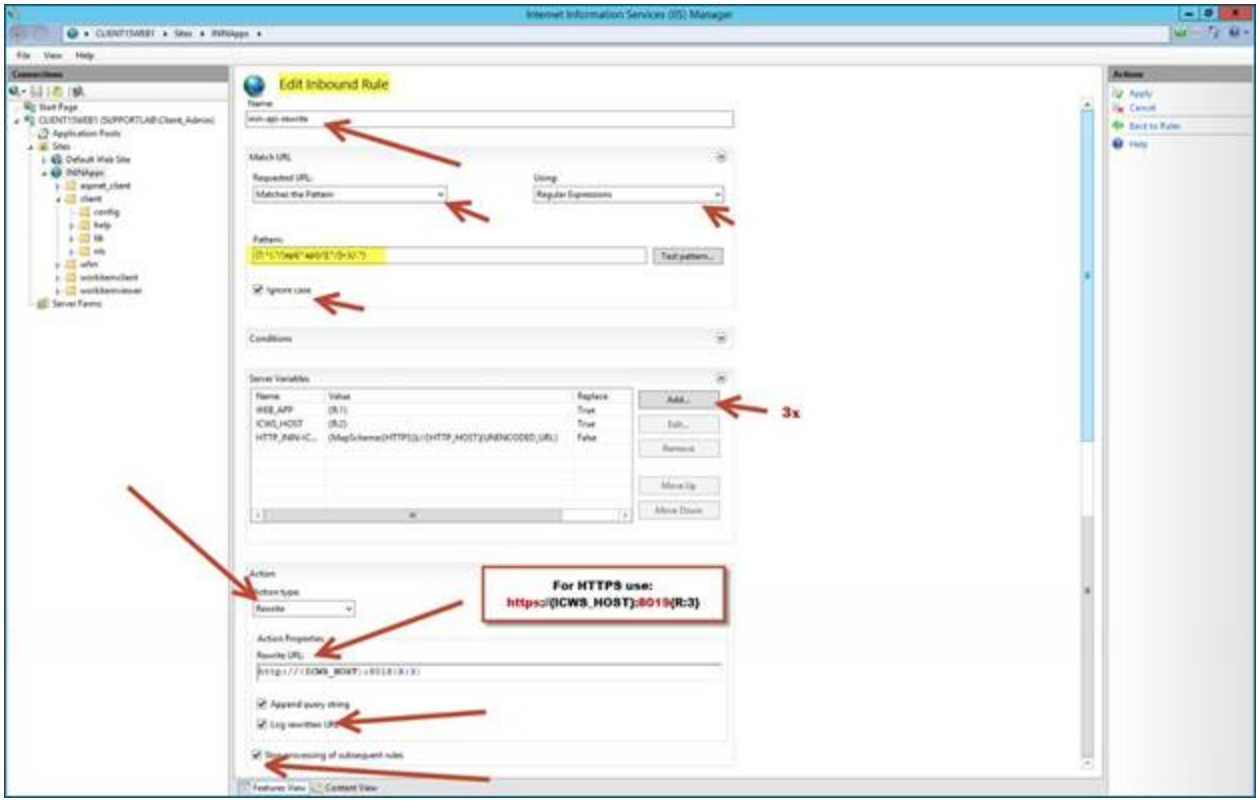
7. Create URL rewrite rules. You will create one inbound rule and two outbound rules.
   a. Click the site in the **IIS Manager**.
   b. Double-click the **URL Rewrite** module.
   c. Navigate to the Actions pane and select **Add Rule(s)**.
   d. For each rule, select **Blank rule** under the appropriate type (Inbound rule or Outbound rule).
   e. Enter the following information for each rule. Tables are provided for ease of copying values, followed by screenshots for each rule.

> **Note:** Do not add conditions for any of the rules.

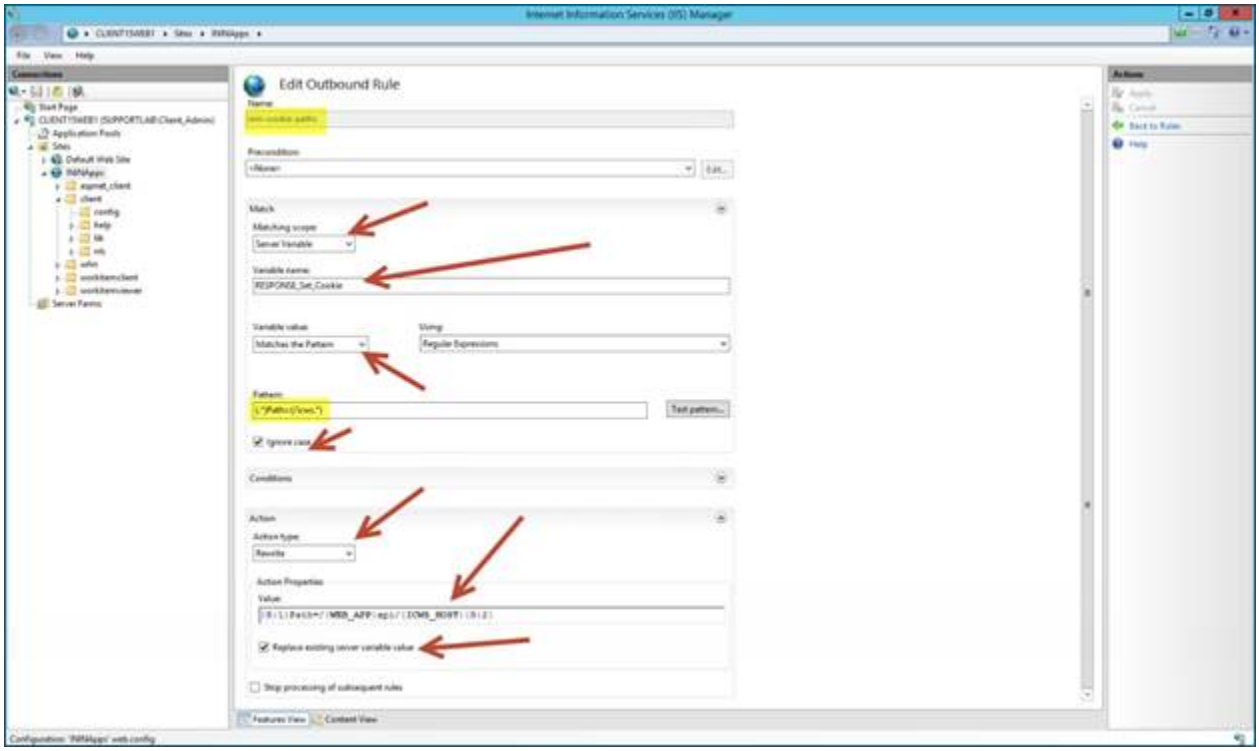
Inbound rule This rule allows the client to reach the Session Manager host that ICWS is served from.	
Name	inin-api-rewrite
Requested URL	Matches the Pattern
Using	Regular Expressions
Pattern	`(?:^(.*/)api|^api)/([^/]+)(/.*)`
Ignore case	checked
Server Variables	See "Server Variables" table below
Action type	Rewrite
Rewrite URL (see "Configure HTTPS for IIS" for HTTPS)	`http://{ICWS_HOST}:8018{R:3}`
Append query string	checked
Log rewritten URL	checked
Stop processing of subsequent rules	checked

Server Variables

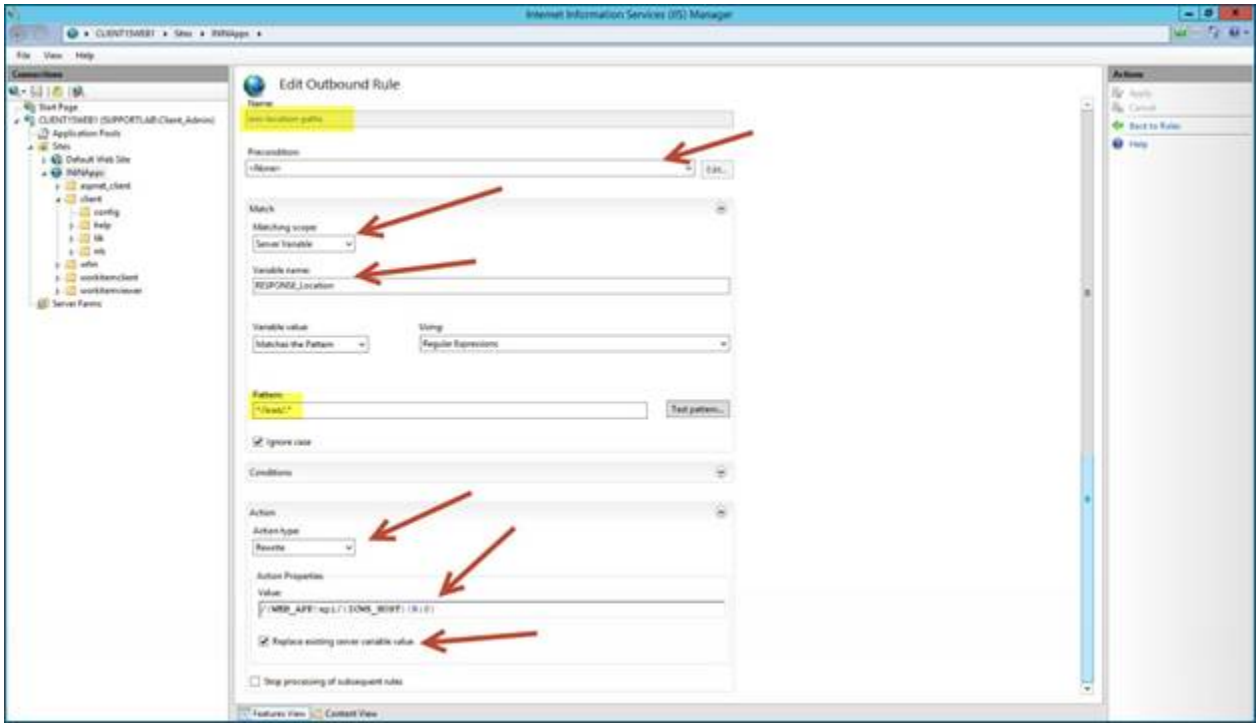
Name	Value	Replace
WEB_APP	`{R:1}`	`True`
ICWS_HOST	`{R:2}`	`True`
HTTP_ININ-ICWS-Original-URL	`{MapScheme:{HTTPS}}://{HTTP_HOST}{UNENCODED_URL}`	`False`



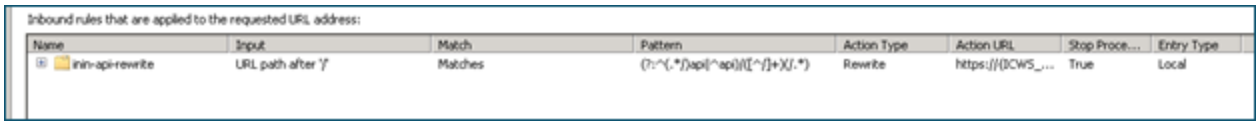
Outbound rule 1 This rule allows the cookies required by ICWS and the client to be located where the client needs them.	
Name	inin-cookie-paths
Precondition	<None>
Matching scope	Server Variable
Variable name	RESPONSE_Set_Cookie
Variable value	Matches the Pattern
Using	Regular Expressions
Pattern	`(.*)Path=(/icws.*)`
Ignore case	checked
Action type	Rewrite
Value	`{R:1}Path=/{WEB_APP}api/{ICWS_HOST}{R:2}`
Replace existing server variable value	checked
Stop processing of subsequent rules	unchecked

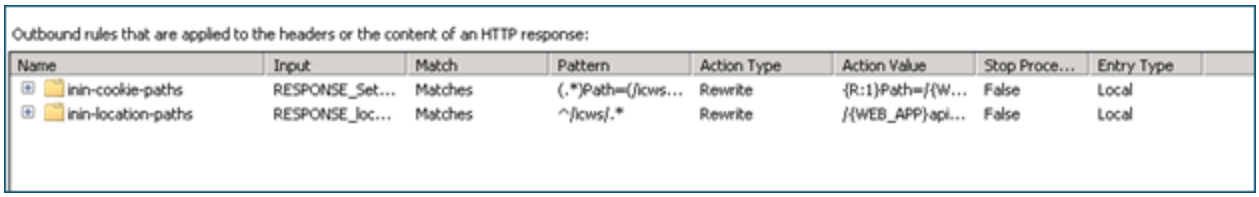


Outbound rule 2 This rule adjusts the location header	
Name	inin-location-paths
Precondition	<None>
Matching scope	Server Variable
Variable name	RESPONSE_location
Variable value	Matches the Pattern
Using	Regular Expressions
Pattern	`^/icws/.*`
Ignore case	checked
Action type	Rewrite
Value	`/{WEB_APP}api/{ICWS_HOST}{R:0}`
Replace existing server value	checked
Stop processing of subsequent rules	unchecked



When you are finished, you will have one inbound rule and two outbound rules:





8. (Optional) Increase the cache sensitivity thresholds if you have application load performance issues.
   a. In **Configuration Editor**, select the **system.webServer/serverRuntime** section.
   b. Update **frequentHitThreshold** to `1`.
   c. Update **frequentHitTimePeriod** to `00:10:00`.
9. Enable static content caching for Interaction Connect:
   The following table summarizes the cache settings. Steps to configure cache settings follow.

   Note: `Client/addins` and `client/config` do not exist in a new installation. If you plan to user `servers.json` or create custom add-ins, use the cache settings below for those folders.

   Recommended cache settings

Directory or file	Expire web content:
client/lib	After 365 days
client/addins	Immediately
client/config	Immediately
client/index.html	After 15 minutes

   a. Expand the site in the tree view and select the `client/lib` folder.
   b. Double click the **HTTP Response Headers** module.
   c. Click **Set Common Headers**.
   d. In the dialog box:
      i. Check **Expire Web content**.
      ii. Select **After** and enter `365 days` in the fields.

e. Click **OK**.
f. Repeat steps 1-5 for the `client/addins` and `client/config` directories with the following difference:
   Select **Immediately** under **Expire web content**.
g. Right-click on the client folder and select **Switch to Content View**.
h. Select **index.html**.
i. In the Actions pane click **Switch to Features View**.
j. Repeat steps b-e for `index.html` (which should now show up under `client`). This time set the **After** expires field to **15 minutes**.

10. Verify that the following MIME types are defined in the IIS Manager for the ININApps Site by double clicking the **MIME Types** module:
    - .css -> text/css
    - .gif -> image/gif
    - .html -> text/html
    - .jpg -> image/jpeg
    - .js -> application/x-javascript
    - .json -> application/json
    - .otf -> application/octet-stream
    - .png -> image/png
    - .svg -> image/svg+xml
    - .ttf -> application/octet-stream
    - .woff -> font/x-woff
    - .woff2 -> application/font-woff2
11. (Optional) If you want to set the CIC server that the CIC Web Applications connect to, follow the instructions in "servers.json".
12. Restart the IIS server by using the `iisreset` command line application.
13. Verify that all applications work as expected.

The basic installation tasks are complete, and the CIC Web Applications should be functional. To enable encryption, see Enable HTTPS between the web browser and IIS and Enable HTTPS between IIS and CIC.

Enable HTTPS between the web browser and IIS

Complete the following tasks to encrypt the connection between the web browser and the web server:

- Step 1: Add a Certificate to the Web Server
- Step 2: Bind the Certificate to the HTTPS port
- Step 3: Enable SSL on the Site

Step 1: Add a Certificate to the Web Server

You can use either a self-signed certificate or a third-party certificate. For information about all the steps to enable HTTPS between the web browser and IIS, see Enable HTTPS between the web browser and IIS.

Create a self-signed certificate

If you choose a self-signed certificate, client workstations need to trust that certificate after it is installed on the web server. For this reason, self-signed certificates are usually used only for testing.

To create a self-signed certificate

1. On the web server, open **IIS Manager**.
2. In the **Connections** pane, select the CIC web applications server.
3. Double-click the **Server Certificates** module.
4. In the **Actions** pane, click **Create Self-Signed Certificate**.
5. In the **Create Self-Signed Certificate** window:
   a. Type a name for the certificate.
   b. Select **Web Hosting** for the certificate store.
6. Click **OK**.

Use a third-party certificate

To use a third-party certificate, create a certificate signing request.

Create certificate signing request

To create a certificate signing request

1. On the web server, open **IIS Manager**.
2. In the **Connections** pane, select the CIC web applications server.
3. Double-click the **Server Certificates** module.
4. Click **Create Certificate Request** to create a certificate signing request (CSR).
5. In the **Request Certificate** window, provide the information for your organization.
6. For **Common name**, type the fully-qualified domain name of the server (for example, www.example.com) and then click **Next**.
7. Choose the appropriate Cryptographic Service Provider Properties and then click **Next**. Ask your third-party certificate authority (CA) which options to choose.
8. Type a file name and location for the CSR and then click **Finish**.
9. Send the generated CSR to your CA for signing.

Complete certificate request

To complete the certificate request

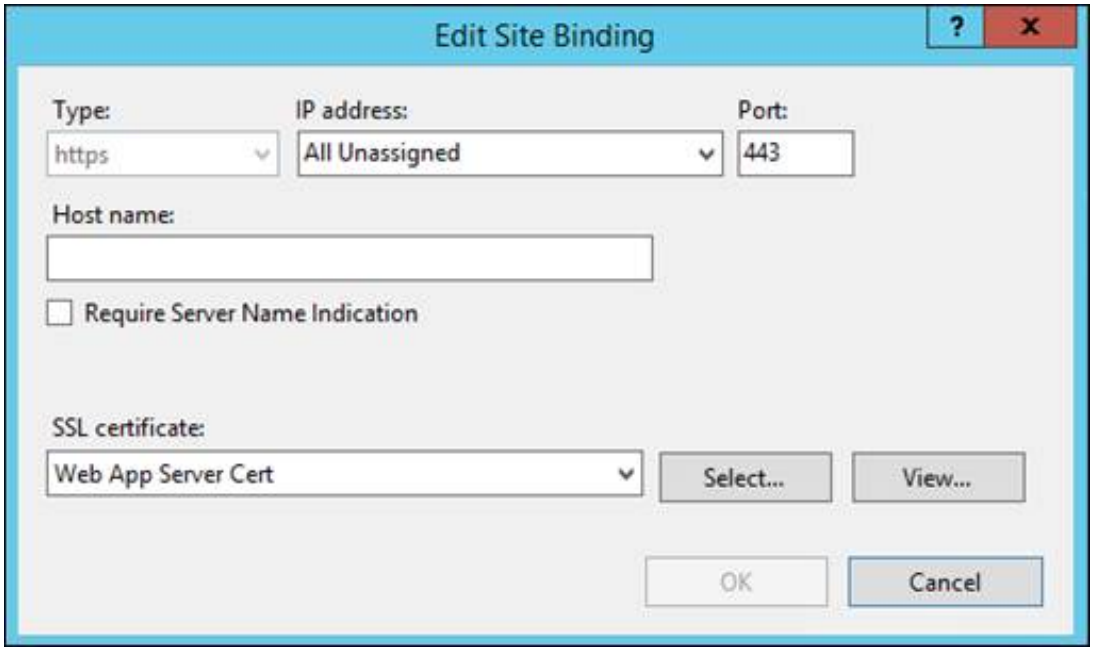
1. Copy the signed certificate you received from the certificate authority to your web server.
2. In IIS Manager, open the **Server Certificates Module**.
3. Click **Complete Certificate Request**.
4. In the Specify Certificate Authority Response window:
   - Select the signed certificate you copied to your web server.
   - Enter a friendly name for the certificate.
   - Select **Web Hosting** for the certificate store.
   - Click **OK**.

Step 2: Bind the Certificate to the HTTPS port

For information about all the steps to enable HTTPS between the web browser and IIS, see Enable HTTPS between the web browser and IIS.

To bind the certificate to the HTTPS port

1. In the **Connections** pane, click the Site for the CIC Web Applications, named **ININApps** in this document.
2. In the **Actions** pane, click **Bindings** and then click **Add**.



3. Change the Type to **https**.
4. In the **SSL certificate** list, select the certificate you previously created or imported and then click **OK**.
5. Click **Close**.

Step 3: Enable SSL on the Site

For information about all the steps to enable HTTPS between the web browser and IIS, see Enable HTTPS between the web browser and IIS.

To enable SSL on the site

1. In the **Connections** pane, click the Site for the CIC Web Applications, named ININApps in this document.
2. Double-click the **SSL Settings** module.
3. Check **Require SSL**.
4. In the **Actions** pane, click **Apply**.
5. Complete the steps to Enable HTTPS between IIS and CIC.

If you used a self-signed certificate, client workstations need to trust the certificate.

Enable HTTPS between IIS and CIC

Tip: Best practice is to use HTTPS from CIC to IIS and from IIS to the web browser, or from IIS to the web browser only. Securing traffic from IIS to CIC only can cause issues with **Secure** cookies.

Complete the following tasks to encrypt the connection between the web server and the CIC server. Ensure that you complete the steps to Enable HTTPS between the web browser and IIS first.

Step 1: Change inbound rule to use HTTPS

To change the inbound rule to use HTTPS

1. On your web server, open IIS Manager and then expand Sites.
2. Select your web site (for example, ININApps).
3. Double-click the **URL Rewrite** module.
4. Open the Inbound Rule **inin-api-rewrite**.
5. In the **Rewrite URL** field, change the Rewrite URLto use HTTPS:
   a. Change the protocol to **https**.
   b. Change the port to **8019**.
6. In the **Actions** pane, click **Apply**.

Step 2: Trust the CIC server HTTPS certificate

Note: If the `Servername_Certificate.cer` file has a Certificate Chain, then you must trust **all** the certificates in the Chain. Check to see if **Issued To** and **Issued By** are different names. If you do not trust all the certificates in the chain, the certificate can't be validated and the SSL handshake will fail. Repeat this task for each Session Manager device in your environment (including both CIC Servers and any Off-Server Session Managers.)

To trust the CIC server HTTPS certificate

1. Locate the HTTPS certificate on your CIC server. The default location is `\I3\IC\Certificates\HTTPS`.
2. Copy **Servername_Certificate.cer** to your web server.
3. On your web server, locate the copied HTTPS certificate and then double-click the certificate.
4. Click **Install Certificate**.
5. Select **Local machine** and then click **Next**.
6. Select **Place all certificates in the following store**.
7. To choose the certificate store, click **Browse** and select **Trusted Root Certification Authorities**.
8. Click **OK**.
9. Click **Next**.
10. Click **Finish**.

Install CIC Web Applications on Apache

To install CIC Web Applications on Apache

1. Create a folder in the document root of your web server for the CIC Web Applications. Verify that your web server software has the appropriate permissions for that newly created folder.

   > **Note:**
   > In this example, the folder is `ININApps`.

2. Download the CIC Web Applications zip file from https://my.inin.com/products/Pages/Downloads.aspx. All the web applications are contained in this single `zip`.
3. Unzip the `CIC Web Applications` folder.
4. Navigate to the `web_files` folder inside the unzipped `CIC Web Applications` folder.
5. Copy all of the folders inside of `web_files`. Each folder contains one CIC web application.
6. Paste the folders copied in the previous step into the directory you created in step 1. Doing so places the appropriate directory structure and files for all CIC Web Applications on your web server.

The following actions take place in the Apache server's `/conf/httpd.conf` file.

1. Set the following minimally required modules to be loaded:
   - One or more auth* modules that are appropriate for your web server
   - dir_module modules/mod_dir.so
   - env_module modules/mod_env.so
   - expires_module modules/mod_expires.so
   - headers_module modules/mod_headers.so
   - mime_module modules/mod_mime.so
   - proxy_module modules/mod_proxy.so
   - proxy_http_module modules/mod_proxy_http.so
   - rewrite_module modules/mod_rewrite.so
   - setenvif_module modules/mod_setenvif.so
2. Change the DocumentRoot as well as the single <Directory> section to point to the CIC Web Applications folder
3. Change the DirectoryIndex property to contain `index.html` and `index.htm`
4. If LimitRequestBody is set to something other than **0**, ensure that you increase it to something greater than or equal to 20971520 (bytes).
5. Set up the proxy rewrite rules:
   a. If Apache is set up for HTTP (possibly in addition to HTTPS), add the following URL rewrite rules at the bottom of `httpd.conf`:

```
RewriteEngine On
RewriteRule "^(/.*|)/api/([^/]+)(/.*)" "http://$2:8018$3"


 [P,E=WEB_APP:$1,E=ICWS_HOST:$2,E=ICWS_PATH:$3,E=HTTP_HOST:%
{HTTP_HOST},E=REQUEST_URI:%{REQUEST_URI},E=SCHEME:%{REQUEST_SCHEME}]
# If you are securing the ICWS host(s), use the following configuration


 in place of the RewriteRule above:
# SSLProxyEngine on
# RewriteRule "^(/.*)/api/([^/]+)(/.*)" "https://$2:8019$3"


 [P,E=WEB_APP:$1,E=ICWS_HOST:$2,E=ICWS_PATH:$3,E=HTTP_HOST:%
{HTTP_HOST},E=REQUEST_URI:%{REQUEST_URI},E=SCHEME:%{REQUEST_SCHEME}]
Header edit Set-Cookie "(.*)Path=(/icws.*)" "$1Path=%{WEB_APP}e/api/%
{ICWS_HOST}e$2"
Header edit Location "^(/icws.*)" "%{WEB_APP}e/api/%{ICWS_HOST}e$1"
SetEnvIf "ININ-ICWS-Original-URL" ".+" HAVE_ININICWSOriginalURL
RequestHeader set "ININ-ICWS-Original-URL" "%{SCHEME}e://%{HTTP_HOST}e%
```

```
{REQUEST_URI}e"


 env=!HAVE_ININICWSOriginalURL
```

b. If Apache is set up for HTTPS (possibly in addition to HTTP), add the following URL rewrite rules to the appropriate VirtualHost section(s) of `httpd-ssl.conf` or `httpd-sni.conf`:

```
RewriteEngine On
RewriteRule "^(/.*|)/api/([^/]+)(/.*)" "http://$2:8018$3"


 [P,E=WEB_APP:$1,E=ICWS_HOST:$2,E=ICWS_PATH:$3,E=HTTP_HOST:%
{HTTP_HOST},E=REQUEST_URI:%{REQUEST_URI},E=SCHEME:%{REQUEST_SCHEME}]
# If you are securing the ICWS host(s), use the following configuration


 in place of the RewriteRule above:
# SSLProxyEngine on
# RewriteRule "^(/.*)/api/([^/]+)(/.*)" "https://$2:8019$3"


 [P,E=WEB_APP:$1,E=ICWS_HOST:$2,E=ICWS_PATH:$3,E=HTTP_HOST:%
{HTTP_HOST},E=REQUEST_URI:%{REQUEST_URI},E=SCHEME:%{REQUEST_SCHEME}]
Header edit Set-Cookie "(.*)Path=(/icws.*)" "$1Path=%{WEB_APP}e/api/%
{ICWS_HOST}e$2"
Header edit Location "^(/icws.*)" "%{WEB_APP}e/api/%{ICWS_HOST}e$1"
SetEnvIf "ININ-ICWS-Original-URL" ".+" HAVE_ININICWSOriginalURL
RequestHeader set "ININ-ICWS-Original-URL" "%{SCHEME}e://%{HTTP_HOST}e%
{REQUEST_URI}e"


 env=!HAVE_ININICWSOriginalURL
```

6. Configure caching for Interaction Connect. Add the following to `conf/httpd.conf`:

```
<DirectoryMatch "/client/">
ExpiresActive On
<FilesMatch "index.html?$">
ExpiresDefault "access plus 15 minutes"
</FilesMatch>
<FilesMatch ".(?:js|css|jpe?g|ico|png|gif|svg|ttf|woff|otf|eot|mp3|wav|ogg)$">
ExpiresDefault "access plus 1 year"
</FilesMatch>
</DirectoryMatch>
<DirectoryMatch "/client/(?:addins|config)/">
<Files "*">
Header Set Cache-Control "no-cache"
</Files>
</DirectoryMatch>
<DirectoryMatch "/client/help/">
ExpiresActive Off
</DirectoryMatch>
```

7. To **prevent cross-frame scripting or clickjacking**, add the following line to `conf/httpd.conf`. Set the options to DENY, SAMEORIGIN, or ALLOW-FROM *origin*. For more information, see Security Considerations.

```
header always set X-Frame-Options "DENY"
```

8. First, identify the directories that should be blocked. Then to **disable directory scanning**, do both of these. For more information, see Security Considerations.

> **Note**: In addition to the recommended configuration changes, you should also assign the appropriate ACLs (Access Control Lists) to the directories on the web server used by your organization.

- Add the following to `conf/httpd.conf`.

```
</DirectoryMatch>
Options FollowSymLinks;
<DirectoryMatch>
```

- Create a ..htaaccess file in the related application directory. In the .htaccess file write:

```
Options - Indexes
```

9. (Re)start the Apache process.
10. Verify that all applications work as expected

Enable HTTPS between the web browser and Apache

Complete the following steps to enable HTTPS between the web browser and Apache.

Step 1: Generate certificate signing request

Note: OpenSSL is an open source tool and is not supported by Genesys. For more information about installing and using Open SSL, see https://www.openssl.org/

To use OpenSSL to generate a CSR

1. Create a destination folder to make finding the CSR and private key easier and then open the command prompt.
2. Open terminal.
3. Go to the folder where you want to generate the CSR.
4. Type the following command: `openssl req -out CSR.csr -new -newkey rsa:2048 -nodes -keyout PrivateKey.key`
5. Follow the prompts to provide the information about the end point for the CSR and then close the window. This process generates two files: `CSR.csr` and `PrivateKey.key`
6. Send the CSR file to the third-party CA to be signed. Do not send the private key. The third-party CA will provide a signed certificate and a CA certificate.

Step 2: Add the signed certificate to the server

1. Copy the signed certificate(s) you received from the CA to your web server.
2. Locate the Apache server's configuration file (for example, `/conf/httpd.conf`).
3. Add the certificate information to the Virtual Hosts section for port 443.

```
 #IP Address of your server and port for HTTPS.
<VirtualHost 192.168.0.1:443>
DocumentRoot /var/www/website
ServerName www.domain.com
SSLEngine on
#The main certificate for your server.
SSLCertificateFile /etc/ssl/crt/primary.crt
#The private key you generated when creating a certificate signing request.
SSLCertificateKeyFile /etc/ssl/crt/private.key
#The intermediate certificate your CA sent, if any.
SSLCertificateChainFile /etc/ssl/crt/intermediate.crt
```

4. Edit the three locations to match the filename and location of your certificates and then save your changes.
5. Restart Apache.
6. Complete the steps to Enable HTTPS between Apache and CIC.

Enable HTTPS between Apache and CIC

> **Tip:** Best practice is to use HTTPS from CIC to Apache and from Apache to the web browser, or from Apache to the web browser only. Securing traffic from Apache to CIC only can cause issues with **Secure** cookies.

Ensure that you completed the steps to Enable HTTPS between the web browser and Apache first.

To enable HTTPS between the CIC server and the web server when using Apache

1. Import the CIC HTTPS certificate to the web server as a trusted root certificate authority.

   > **Note:** Consult the documentation for your server operating system because instructions for adding a trusted root certificate authority vary by operating system.

2. Verify that your web server's configuration file includes a rewrite rule for HTTPS traffic over port 8019:

   Apache

   ```
   SSLProxyEngine on
   # RewriteRule "^(/.*)/api/([^/]+)(/.*)" "https://$2:8019$3"
   [P,E=WEB_APP:$1,E=ICWS_HOST:$2,E=ICWS_PATH:$3,E=HTTP_HOST:%{HTTP_HOST},E=REQUEST_URI:%
   {REQUEST_URI},E=SCHEME:%{REQUEST_SCHEME}]
   ```

   Nginx

   ```
   proxy_pass          https://$server:8019$icws_path$is_args$args;
   ```

Install CIC Web Applications on Nginx

To install CIC Web Applications on Nginx

1. Create a folder in the document root of your web server for the CIC Web Applications. Verify that your web server software (IIS, Apache, or Nginx) has the appropriate permissions for that newly created folder.

   Note: In this example, the folder is `ININApps`.

2. Download the CIC Web Applications zip file fromhttps://my.inin.com/products/Pages/Downloads.aspx. All of the web applications are contained in this single `zip`.
3. Unzip the `CIC Web Applications` folder.
4. Navigate to the `web_files` folder inside the unzipped `CIC Web Applications` folder.
5. Copy all of the folders inside of `web_files`. Each folder contains one CIC web application.
6. Paste the folders copied in the previous step into the directory you created in step 1. Doing so places the appropriate directory structure and files for all CIC Web Applications on your web server.
7. In Nginx `/conf/nginx.conf`, verify the following (minimally required) configuration entries are set or added (whether at the http or server level):

```
include                     mime.types;
default_type                application/octet-stream;
sendfile                    on;
keepalive_timeout           65;
gzip                        on;
gzip_types                  text/plain text/css application/javascript applications/json
                            image/svg_xml;
index                       index.html index.htm;
client_max_body_size        0;
autoindex                   on;
```

8. To **prevent cross-frame scripting or clickjacking**, add the following parameter to `conf/nginx.conf`. Set the options to DENY, SAMEORIGIN, or ALLOW-FROM *origin*. For more information, see Security Considerations.

```
add_header X-Frame-Options "DENY"
```

9. First, identify the directories that should be blocked. Then to **block all or some directories from directory scanning**, in `/conf/nginx.conf`, add the following. If you do not specify a directory, then the rule applies to all folders.

   Note: In addition to the recommended configuration changes, you should also assign the appropriate ACLs (Access Control Lists) to the directories on the web server used by your organization.

```
location /{YOUR DIRECTORY NAME}{
   autoindex off;
}
```

10. Set the `root` entry for the server to the CIC Web Applications folder
11. Add the following rewrite rules within the server object:

```
set $ininIcwsOriginalUrl $http_inin_icws_original_url;
if ($ininIcwsOriginalUrl !~ .+) {
set $ininIcwsOriginalUrl $scheme://$http_host$request_uri;
}
location ~* (?:^(.+)/api|^/api)/([^/]+)(/.+)$ {
set                     $web_app $1;
set                     $server $2;
set                     $icws_path $3;
proxy_cookie_path       /icws/ $web_app/api/$server/icws/;
proxy_redirect          /icws/

 $web_app/api/$server/icws/;
proxy_set_header        X-Forwarded-For

 $proxy_add_x_forwarded_for;
proxy_set_header        ININ-ICWS-Original-URL

 $ininIcwsOriginalUrl;
proxy_pass              http://$server:8018$icws_path$is_args$args;
            add_header
```

```
 P3P 'CP="CAO PSA OUR"';
   # If you are securing the ICWS host(s), use this rewrite

 rule instead:
   # proxy_passhttps://$server:8019$icws_path$is_args$args;
   proxy_buffering off;
   proxy_http_version 1.1;
}
```

Note: If the X-Forwarded-For header is already being set to $proxy_add_x_forwarded_for at a higher level, it is not needed here. That header is, however, required for these API calls.

Note: All proxy_set_header directives must be placed at the same level within the configuration file, i.e. in the code above, the proxy_set_header directives both must either be inside of the `location` block or outside of the `location` block.

12. Add the following cache rules within the server object:

```
location ~ /client/ {
   location ~ /client/help/ {
      expires off;
   }
   location ~ /client/(?:addins|config)/ {
      add_header Cache-Control "no-cache";
   }
   location ~ index.html?$ {
      expires 15m;
   }
   location ~ .(?:js|css|jpe?g|ico|png|gif|svg|ttf|woff|otf|eot|mp3|wav|ogg)$ {
      expires 1y;
   }
}
```

13. (Re)start the Nginx process
14. Verify that all applications work as expected.

Enable HTTPS between the web browser and Nginx

Complete the following steps to enable HTTPS between the web browser and Nginx.

Step 1: Generate certificate signing request

Note: OpenSSL is an open source tool and is not supported by Genesys. For more information about installing and using Open SSL, see https://www.openssl.org/

To use OpenSSL to generate a CSR

1. Create a destination folder to make finding the CSR and private key easier. Open the command prompt.
2. Open terminal.
3. Use `cd` to navigate to the folder where you want to generate the CSR.
4. Type the following command: `openssl req -out CSR.csr -new -newkey rsa:2048 -nodes -keyout PrivateKey.key`
5. Follow the prompts to enter the information about the end point for the CSR and then close the window. This process generates two files: `CSR.csr` and `PrivateKey.key`
6. Send the CSR file to the third-party CA to be signed. Do not send the private key. The third-party CA will provide a signed certificate and a CA certificate.

Step 2: Add the signed certificate to the server

To add the signed certificate to the server

1. Copy the signed certificate(s) you received from the CA to your web server.
2. To use multiple certificates, Nginx requires that all certificates be in one file. To combine the certificates:

```
cat your_domain_name.crt ExampleCA.crt >> bundle.crt
```

3. Open the virtual hosts file and add the following information:

```
server {
listen   443;
ssl    on;ssl_certificate    /etc/ssl/your_domain_name.pem;
 (or bundle.crt)
ssl_certificate_key    /etc/ssl/your_domain_name.key;
server_name your.domain.com;
access_log /var/log/nginx/nginx.vhost.access.log;
error_log /var/log/nginx/nginx.vhost.error.log;
location / {
root    /home/www/public_html/your.domain.com/public/;
index  index.html;
}}
}
```

4. Edit the certificate locations to match the filename and location of your certificates and then save the file.
5. Restart Nginx.
6. Complete the steps to Enable HTTPS between Nginx and CIC.

Enable HTTPS between Nginx and CIC

> **Tip:** Best practice is to use HTTPS from CIC to Nginx and from Nginx to the web browser, or from NGIX to the web browser only. Securing traffic from Nginx to CIC only can cause issues with **Secure** cookies.

Ensure that you completed the steps to Enable HTTPS between the web browser and Nginx first.

To enable HTTPS between the CIC server and the web server when using Nginx

1. Import the CIC HTTPS certificate to the web server as a trusted root certificate authority.

   > **Note:** Consult the documentation for your server operating system because instructions for adding a trusted root certificate authority vary by operating system.

2. Verify that your web server's configuration file includes a rewrite rule for HTTPS traffic over port 8019:

   Nginx

   ```
   proxy_pass          https://$server:8019$icws_path$is_args$args;
   ```

Application Configuration

servers.json

To restrict the CIC servers users can choose, add a file called `servers.json` under the `config` directory in the `client` folder under the `CIC Web Applications` folder. If users are logging into one server, the configuration is as follows:

```
{
  "version": 1,
  "servers": [
    { "host": "salesic",
 "displayName": "Sales" }
  ]
}
```

If users can choose from a predefined list of CIC servers:

```
{
  "version": 1,
  "servers": [
    { "host": "salesic",
 "displayName": "Sales", "order": 0
},
    { "host": "supportic",
 "displayName": "Support", "order": 1 }
  ]
}
```

Note: The *host* is the CIC server, not the OSSM. The old key `hostName` is deprecated but still supported.

This file requires strict JSON. Hanging commas or single quotes will cause errors.

If no servers are specified, (e.g., servers: []), then the server selection page during log on will have a text field to enter the CIC server to connect to. If a single server is specified, the server selection page will be skipped. If two or more servers are specified, then the server selection page will have a list box to select a server.

altHostHints

In 2015 R4 and following, `servers.json` supports listing alternate hosts for a given server entry, which is helpful in switchover scenarios when the primary server is not reachable or the switchover pair is in a DNS round-robin. The new **altHostHints** property lists alternate servers the application can use.

```
{
  "version": 1,
  "servers": [
    { "host": "notReachableSalesIc",
 "displayName": "Sales",
 "altHostHints": [ "salesic1", "salesic2"  ] }
  ]
}
```

In a CIC environment with OSSMs, it is advantageous to put the all of the OSSMs in the `altHostHints` array.

With a round-robin switchover pair, do not place the short host name in `host` (e.g. `server1`). Instead use fully qualified domain names in both `host` and `altHostsHints`. List the primary server in `hosts` and the backup servers in `altHostHints`.

For example, here is a configuration for server1 (without OSSMs):

```
{
  "version": 1,
  "servers": [
    { "host": "server1.example.com",  "displayName": "Indianapolis",
"altHostHints":  [ "server2.example.com" ] }
]
}
```

The `host` property should not be changed because settings will depend on it in 2015 R4.

Once we successfully connect with the client, the alternate hosts are cached from what the server gives us.

Application updates (subsequent installs)

New versions of the CIC Web Applications are located at https://my.inin.com/products/Pages/Downloads.aspx. Download and unzip the new version of the applications before completing the instructions that follow.

Back up existing applications

Back up Genesys web application files before performing updates. Backing up the files allows you to roll back the installation if necessary.

All CIC Web Applications should exist in a single directory, which should be zipped and backed up. Genesys recommends adding the date and release number to the zip file's name.

Upgrade Interaction Connect

As of 2016 R1, Interaction Connect supports add-ins. When upgrading Interaction Connect, back up the add-ins and config directories before upgrading Interaction Connect.

In the following instructions, `[client]` is the Interaction Connect root directory on your web server. For example, if Interaction Connect is in a directory named `client`, back up `client/addins` and `client/config`.

To upgrade Interaction Connect

1. Back up the `[client]/addins` and the `[client]/config` directories.
2. Download and unzip the new version of Interaction Connect. See steps 2-4 in "All server types: download, unzip, and copy application files" for detailed instructions.
3. In the unzipped folder, navigate to `web_files/client` and copy the contents inside of the `client` directory.

   > **Note:** If you copy the entire client directory, instead of the contents of the client directory, you will need to add the backed up `[client]/addins` and `[client]/config` directories back to the `[client]` directory after upgrading.

4. Move the copied files for the new version of Interaction Connect into the `[client]` directory for the CIC Web Applications on your web server, replacing files when they collide.
5. Verify that `[client]/addins` and `[client]/config` have the appropriate files.

Copy files to the web server

Copy only the application directories, leaving in place other files such as configuration files. In the following examples, `source` is a directory that solely contains the unzipped application directories.

Windows

```
ROBOCOPY \path\to\source \path\to\webserver\root /e
```

***nix**

```
rsync -a /path/to/source/ /path/to/webserver/root/
```

Rollbacks

To rollback an update, use the zip file you created as a backup to replace the files currently in place.

Load balancing multiple web servers

CIC Web Applications consist of static files, with no server-side session state. Therefore, affinity is not needed for web server requests. All requests for a single session do need affinity with an CIC server (or OSSM), but the URL rewrite rules take care of that for you.

For information about load balancing multiple web servers, see the following resources:

IIS: http://technet.microsoft.com/en-us/library/jj129390.aspx

Apache: http://httpd.apache.org/docs/2.2/mod/mod_proxy_balancer.html

Nginx: http://nginx.org/en/docs/http/load_balancing.html

Troubleshooting

On IIS, the CIC Web Application's main.js takes a long time to load when compression is enabled

Solution

- The Chrome **Network** tab is useful for determining which request is taking the longest. Turn on the **Disable caching** option in the Chrome **Network** tab.
- Ensure that output caching is enabled on IIS and is working properly. Perfmon contains several counters in the Web Service Cache category that monitor the state of the cache, including the size, number of files/URIs cached, and the hit rate.
- Check for I/O performance issues

On IIS, a slow request becomes faster after reloading the resource multiple times

Solution

Tweak the **frequentHitThreshold** and **frequentHitTimePeriod** parameters to ensure that IIS is retaining the cached responses.

Appendix A: IIS XML Configuration

To configure IIS XML

1. Add the following location entry in `ApplicationHost.config` at `%WINDIR%\system32\inetsrv\config\applicationHost.config`
   Put this location entry above or below the other location entry already present in order to avoid breaking `applicationHost.config`
   The location path needs to match the IIS Site name.

```
<location path="ININApps">
  <system.webServer>
      <rewrite>
          <allowedServerVariables>
            <add name="WEB_APP" />
            <add name="ICWS_HOST" />
            <add name="HTTP_ININ-ICWS-Original-URL" />
          <</allowedServerVariables>
      </rewrite>
  </system.webServer>
</location>
```

2. If there is not a `web.config` file, create a `web.config` file with the following content in the `CIC Web Applications` folder on your web server:

```
<configuration>
    <system.webServer>
        <rewrite>
            <rules>
                <rule name="inin-api-rewrite" enabled="true" stopProcessing="true">
                    <match url="(?:^(.*/)api|^api)/([^/]+)(/.*)" />
                    <serverVariables>
                        <set name="WEB_APP" value="{R:1}" />
                        <set name="ICWS_HOST" value="{R:2}" />
                        <set name="HTTP_ININ-ICWS-Original-URL"


value="{MapScheme:{HTTPS}}://{HTTP_HOST}{UNENCODED_URL}" replace="false" />
                    </serverVariables>
                    <action type="Rewrite" url="http://{ICWS_HOST}:8018{R:3}"
logRewrittenUrl="true" />                    <!--
                        If you are securing the ICWS host(s) with https, use


the following rewrite rule instead
                        <action type="Rewrite"
url="https://{ICWS_HOST}:8019{R:3}" logRewrittenUrl="true" />
                    -->
                </rule>
            </rules>
<outboundRules>
                <rule name="inin-cookie-paths">
                    <match serverVariable="RESPONSE_Set_Cookie"
pattern="(.*)Path=(/icws.*)" />
                    <action type="Rewrite"
value="{R:1}Path=/{WEB_APP}api/{ICWS_HOST}{R:2}" />
                </rule>
                <rule name="inin-location-paths">
                    <match serverVariable="RESPONSE_Location"
pattern="^/icws/.*" />
                    <action type="Rewrite"
value="/{WEB_APP}api/{ICWS_HOST}{R:0}" />
                </rule>
            </outboundRules>
            <rewriteMaps>
                <rewriteMap name="MapScheme">
                    <add key="on" value="https" />
```

```
                    <add key="off" value="http" />
                </rewriteMap>
            </rewriteMaps>
        </rewrite>
        <security>
            <requestFiltering allowHighBitCharacters="true" />
        </security>
        <httpCompression>
            <staticTypes>
                <add mimeType="application/x-javascript" enabled="true" />
            </staticTypes>
        </httpCompression>
    </system.webServer>
<location path="client/lib">
    <system.webServer>
        <staticContent>
            <clientCache cacheControlMode="UseMaxAge" cacheControl MaxAge="365.00:00:00" />
        </staticContent>
    </system.webServer>
</location>
<location path="client/nls">
    <system.webServer>
        <staticContent>
            <clientCache cacheControlMode="UseMaxAge" cacheControlMaxAge="365.00:00:00" />
        </staticContent>
    </system.webServer>
</location>
<location path="client/addins">
    <system.webServer>
        <staticContent>
            <clientCache cacheControlMode="DisableCache" />
        </staticContent>
    </system.webServer>
</location>
<location path="client/config">
    <system.webServer>
        <staticContent>
            <clientCache cacheControlMode="DisableCache" />
        </staticContent>
    </system.webServer>
</location>
<location path="client/index.html">
    <system.webServer>
        <staticContent>
    <clientCache  cacheControlMode="UseMaxAge"
cacheControlMaxAge="0.00:15:00" />
        </staticContent>
    </system.webServer>
</location>
</configuration>
```

3. To **prevent cross-frame scripting or clickjacking**, add the following custom header to web.config. Set the options to DENY, SAMEORIGIN, or ALLOW-FROM *origin*. For more information, see Security Considerations.

```
<system.webServer>
...
   <httpProtocol>
      <customHeaders>
         <add name="X-Frame-Options" value="deny" />
      </customHeader>
   </httpProtocol>
...
</system.webServer>
```

4. To **disable directory listing**, add this to web.config. For more information, see Security Considerations.

> **Note**: In addition to the recommended configuration changes, you should also assign the appropriate ACLs (Access Control Lists) to the directories on the web server used by your organization.

```
<system.webServer>
```

```
    <directoryBrowse enabled="false" />
</system.webServer>
```

- Create a .htaaccess file in the related application directory. In the .htaccess file write:

```
Options - Indexes
```

Change Log

The following table lists the changes to the *CIC Web Applications Installation and Configuration Guide* since its initial release.

Date	Change
01-October-2014	Initial Release
01-August-2015	• Added alternate hosts information for 2015 R4, updated cover page and copyright info • Added recommended cache settings for Interaction Connect • Added IIS Manager configuration steps and placed IIS XML configuration in Appendix A • Fixed reference to incorrect server variable name
01-September-2015	Added information about updating Interaction Connect.
01-November-2015	• Added "Configure HTTPS for IIS" • Added "Configure HTTPS for Apache" • Added "Configure HTTPS for Nginx"
01-December-2015	Added instructions for disabling response buffering in IIS install instructions.
01-May-2016	Reorganized installation instructions to clarify required and optional tasks.
01-August-2016	Added "Desktop alerts do not appear in Interaction Connect" section in *Troubleshooting* chapter. (Removed section in 2018; alerts supported in 2018 R3 and later.)
01-November-2016	In "Step 1: Change Inbound rule to use HTTPS", fixed typographical error in step 6.
01-February-2017	• Added Tip about best practice to Enable HTTPS between IIS and CIC, Enable HTTPS between Apache and CIC, and Enable HTTP between Nginx and CIC. • In the Step 3: Configure IIS section, added to step 4 of procedure, iii. Update Maximum allowed content length (bytes) to something greater than or equal to "20971520". • In "Install CIC Web Applications on Apache" section, inserted a new step: 4. If LimitRequestBody is set to something other than 0, ensure that you increase it to something greater than or equal to 20971520 (bytes). Then renumbered rest of procedure.
01-June-2017	• Added `.woff2 -> application/font-woff2` to MIME types in step 10 of "Configure IIS" procedure.
01-July-2017	• Rebranded this document to apply Genesys corporate lexicon.
01-August-2017	• Updated cover, copyrights and trademarks pages.
25-September-2018	Added to Prerequisites: If you use the same reverse proxy for CIC Web Applications and Genesys Intelligent Automation, then the Intelligent Automation rewrite rule must come before the CIC Web Application rule. For more information, see the PureConnect Integration with Genesys Intelligent Automation Technical Reference.
22-October 2018	Updated servers.json. Corrected location of servers.json file to read: To restrict the CIC servers users can choose, add a file called `servers.json` under the `config` directory in the client folder under the `CIC Web Applications` folder.
20-February-2019	Removed all references to appsettings.json; added note indicating that you must use the latest version of the CIC Web Applications.
22-February-2019	Updated web server software requirements.
28-February-2019	Corrected URLs for downloading the Application Request Routing extension and the URL Rewrite extension.
20-June-2019	Fixed incorrect symbol in code sample in step 9 of the Install CIC Web Applications on Nginx topic.
15-November-2019	Removed unrelated f ile names from code samples. Fixed formatting in code examples.
14-May-2020	Added new topic, Security Considerations. Added sections: Cross-Frame Scripting or clickjacking, Prevent directory scan, and File Uploads.Updated these topics: Appendix A: IIS XML Configuration, Install CIC Web Applications on Nginx, and Install CIC Web Applications on Apache. Added link to Appendix A to the Step 3: Configure IIS topic.

20-May-2020	Updated version number on title page.
21-July-2020	Fixed typo in Security Considerations topic; changed "attacked" to "attacker.".
4-November-2020	Fixed code samples in Install CIC Web Applications on Apache and Install CIC Web Applications on Nginx.
21-February-2023	Added two sections to the topic "Security Considerations": 1) Enable Strict Transport Security, 2) Configuring HSTS header in IIS /NGINX / Apache