PTC Tempo Thermal Cyclers

API Reference Guide

Catalog #12015382
12015392
12015394

BIO-RAD

PTC Tempo 96, PTC Tempo 384, and PTC Tempo Deepwell

API Reference Guide

PTC Tempo API Activation Code

BIO-RAD

Bio-Rad™ Technical Support

The Bio-Rad Technical Support department in the U.S. is open Monday through Friday, 5:00 AM to 5:00 PM, Pacific Time.

Phone: 1-800-424-6723, option 2

Email: Support@bio-rad.com (U.S./Canada Only)

For technical assistance outside the U.S. and Canada, contact your local technical support office or click the Contact us link at bio-rad.com.

Notice

The PTC Tempo thermal cycler uses the open source software, which contains software licensed under the following licenses as well as others:

- GNU General Public License v 1.0, 2.0, 3.0
- GNU Lesser General Public License v 2.0, 2.1, 3.0
- BSD License

Certain OPEN LICENSES require that the source materials be made available to recipients or other requestors under the terms of the same OPEN LICENSE. The corresponding open source software can be downloaded at:

http://bio-rad.com/PTCTempo-opensource

To view the OPEN LICENSES

1. On the Home screen, tap Tools and then tap About.
2. On the About screen, select Legal Notices in the System section.

Revision History

Document	Date	Description of Change
PTC Tempo Thermal Cycler API Reference Guide (Doc ID #10000156544)	July 2023	Initial Release

Table of Contents

Chapter 1 Introduction to the Bio-Rad PTC Tempo Thermal Cycler API

The PTC Tempo thermal cycler is Bio-Rad's newest conventional thermal cycler designed with an automated lid that allows the instrument to seamlessly integrate into any automated laboratory workflow.

The PTC Tempo thermal cycler Application Programming Interface (API) is a RESTful service that enables applications to control the PTC Tempo 96, PTC Tempo Deepwell, and PTC Tempo 384 thermal cycler and monitor its status. Client applications use the PTC Tempo thermal cycler API to send HTTP or HTTPS requests and parse each response. The API uses JSON as its communication format.

Supported PTC Tempo thermal cycler API requests include:

- Open, close, and get the status of the thermal cycler's mechanized lid (opened, opening or closed)
- Run or stop a protocol run
- Get the status of the thermal cycler (including lid errors)
- Query a list of templates or public protocols
- Query protocol run reports

This guide provides an overview of the PTC Tempo thermal cycler API, specifies the resources available, and details the request and response behavior of each resource. It also provides the steps to start and stop automation and to create an Automation user. It assumes that the reader has a working knowledge of REST API requests and responses and is familiar with the PTC Tempo thermal cycler.

Note: You can configure your application to send API requests to more than one PTC Tempo thermal cycler in an automated workflow. Each thermal cycler is identified by a unique IP address that is included in an API request.

Note: In order for your application to send automation API requests, you must use PTC Tempo thermal cycler automation API version 1.0.0.

Important: Cybersecurity is the protection of assets in cyberspace from cyberattacks. Cybersecurity is Bio-Rad’s ability to secure its people, information, systems, and reputation in cyberspace. Cyberspace is the always-on, technologically interconnected world; it consists of people, organizations, information, and technology.

Fast reaction is important with cybersecurity issues! If you suspect that there may be a cybersecurity issue concerning your instrument or that cybersecurity has been breached at your site, contact your Bio-Rad representative for technical support immediately.

Chapter 2 Starting PTC Tempo Automation

To use the PTC Tempo thermal cycler Automation API, you must

- Activate the PTC Tempo thermal cycler Automation API license key
- Create the Automation user
- Start automation

These tasks are explained in detail in the sections that follow.

Activating PTC Tempo API and Automation

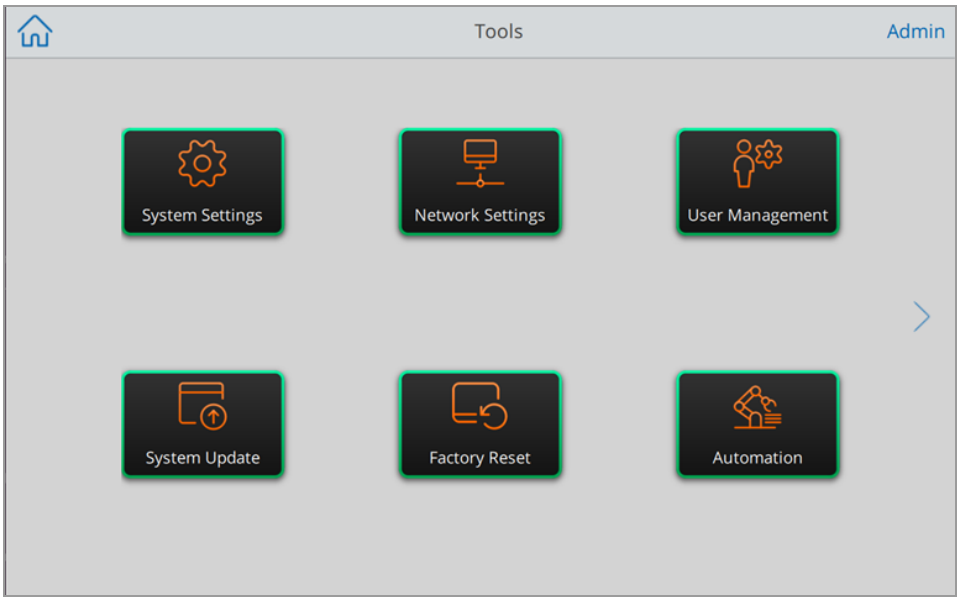
To activate the PTC Tempo API and to initially enable automation, an Admin user must enter a license key using the PTC Tempo thermal cycler alphanumeric keypad. When the system validates the license key, you can send URL requests to the server and enable automation.

Note: The license key appears on a sticker that ships with this API Reference Guide. If you do not have the license key, obtain it from your Bio-Rad customer service representative before attempting to start automation.

To activate the PTC Tempo thermal cycler API license key

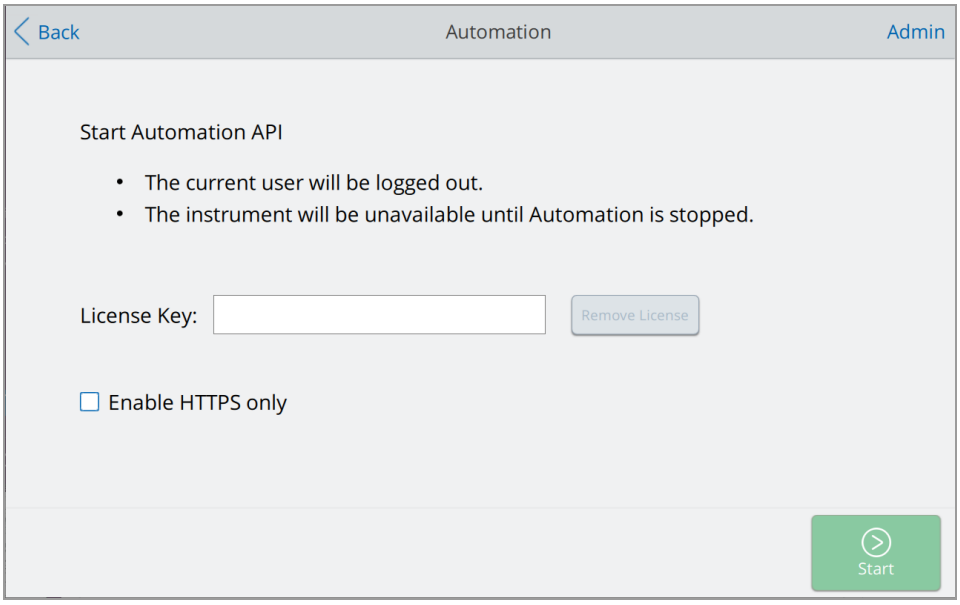
1. Log in as an Admin user or as a user with admin privileges.
2. On the Home screen, tap Tools to open the Admin Tools screen.

   The Tools screen appears.



3. Tap Automation.

   The Automation screen appears.



4. Tap the License Key field. In the alphanumeric keypad, enter the license key that appears on the sticker.

   Note: The license key cannot exceed 20 characters.

5. Tap OK to confirm.
6. Tap Start to start Automation and (if required) create the Automation user.

The PTC Tempo thermal cycler is ready for automation.

Creating the Automation User

A unique Automation user controls the PTC Tempo thermal cycler automation by sending requests to the automation server via the PTC Tempo thermal cycler API. An Admin user must start automation on the PTC Tempo thermal cycler before the Automation user can use the Automation API and prepare the HTTP server to receive API requests. The system creates an Automation user when an Admin user initially starts automation. This user requires a password.

Note: The PTC Tempo thermal cycler allows for only one Automation user.

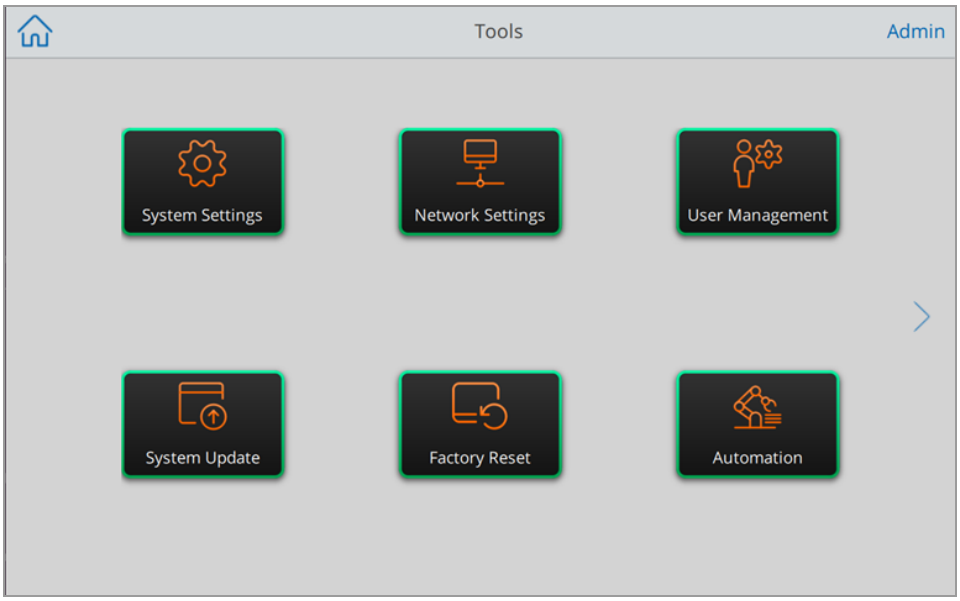
Important: The Automation user profile name is reserved for the user who will run the PTC Tempo thermal cycler automation API. If this user name already exists, protocols created by this user will be available for the automation API.

Note: In order to prevent the instrument status from displaying on BR.io, Bio-Rad recommends that you unlink the PTC Tempo thermal cycler from BR.io for all users before initiating API automation runs. See the PTC Tempo Thermal Cycler Instrument Guide for more information.

To create an Automation user

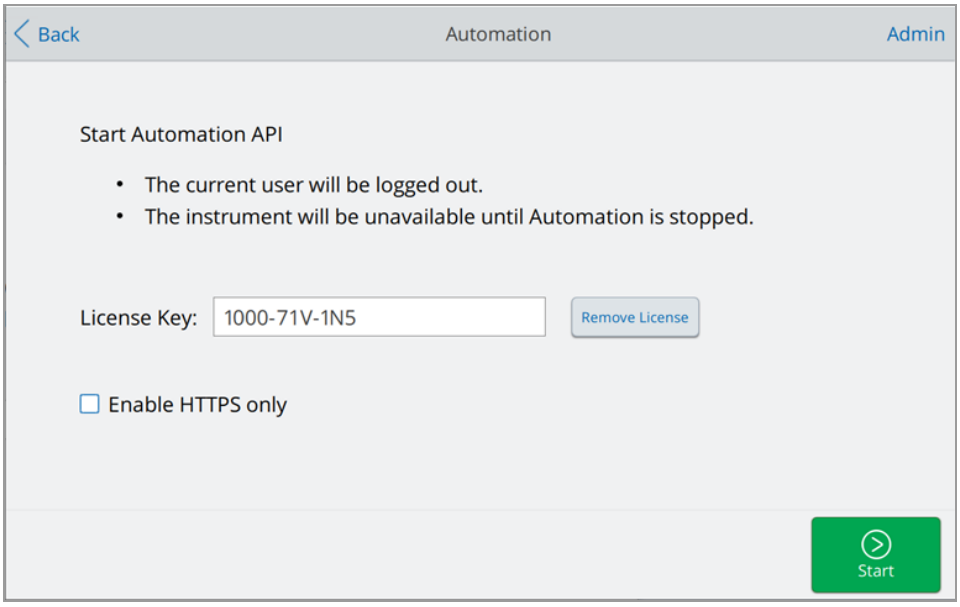
1. Log in as an Admin user or as a user with admin privileges.
2. On the Home screen, tap Tools to open the Admin Tools screen.

   The Tools screen appears.



3. Tap Automation.

   The Automation screen appears.

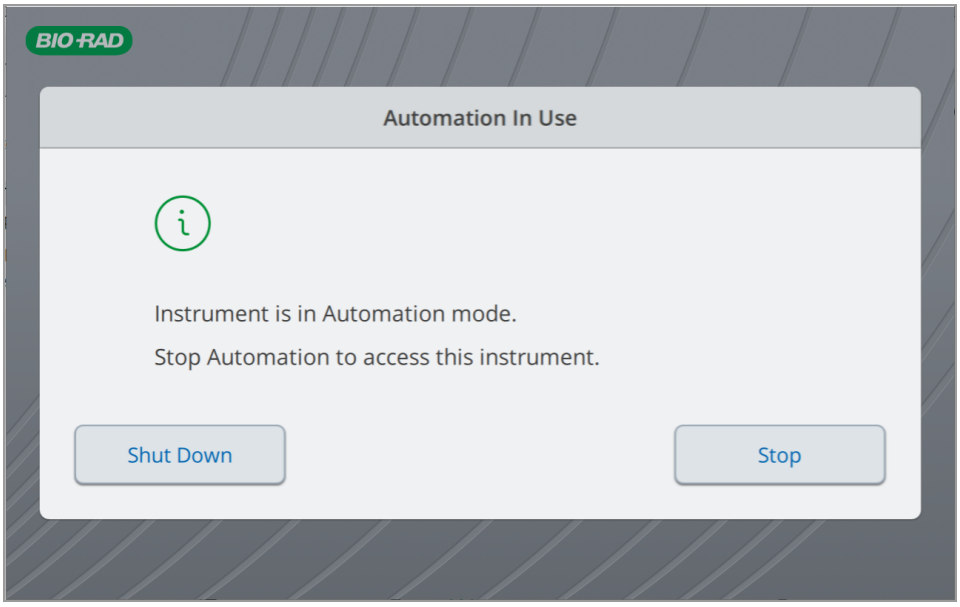


4. (Optional) The Automation API works with both HTTP and HTTPS. Select the Enable HTTPS only checkbox if your site requires only encrypted requests with HTTPS. The setting persists until the Admin user clears the checkbox.

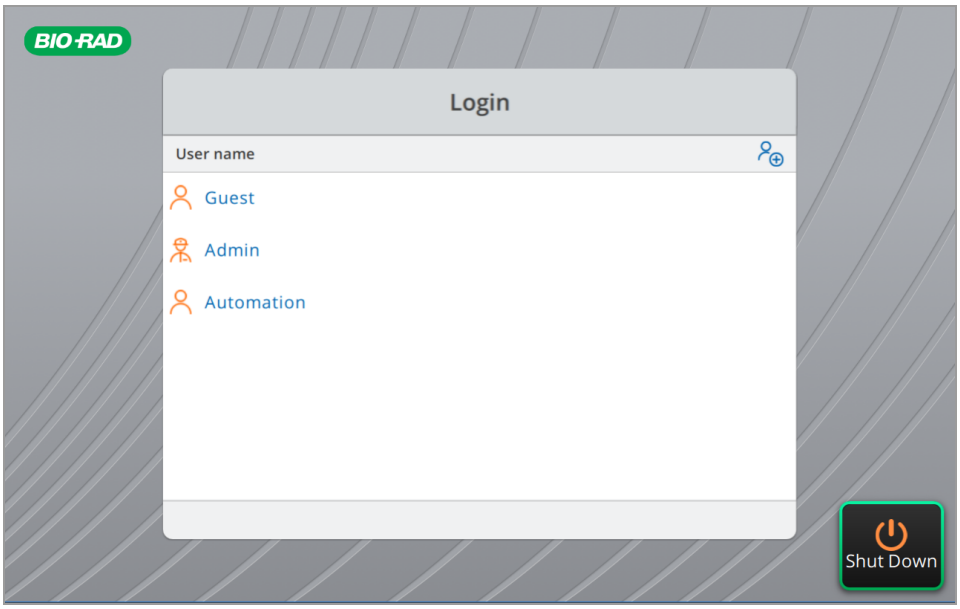
   Tap Start.

5. The Create Password dialog box appears. Enter and confirm a password for the new user in the alphanumeric keyboard that appears, and then tap Save Password.

6. The system creates the new Automation user and starts automation. After the system creates the Automation user, the Admin user is logged out, and the Automation in Use screen appears.



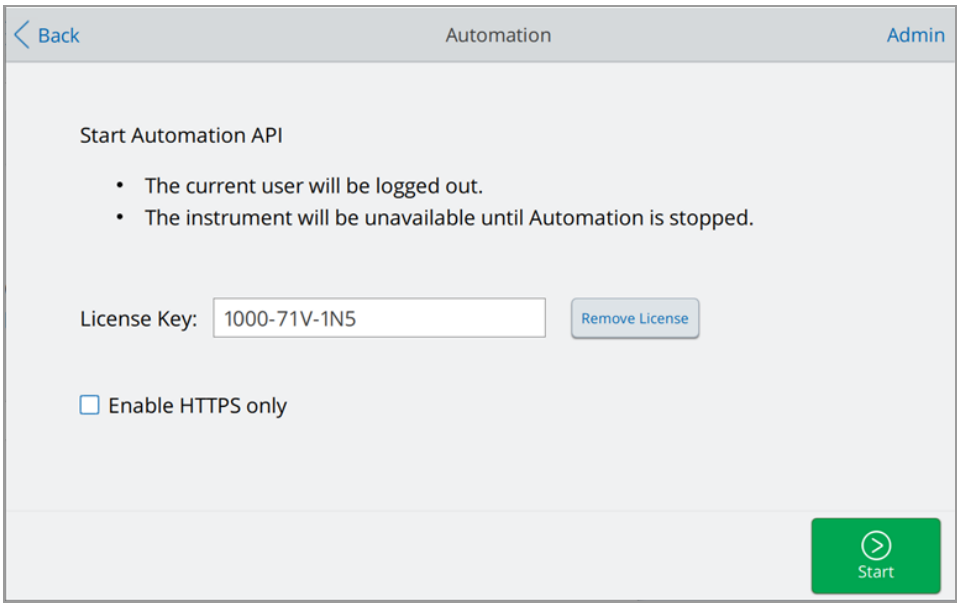
Note: After the system creates the Automation user, that user appears in the list of users on the Login screen. You can log in to the PTC Tempo thermal cycler as the Automation user to create and save protocols to the Automation folder. However, only the Admin user can initialize automation.



Starting Automation

To start PTC Tempo automation

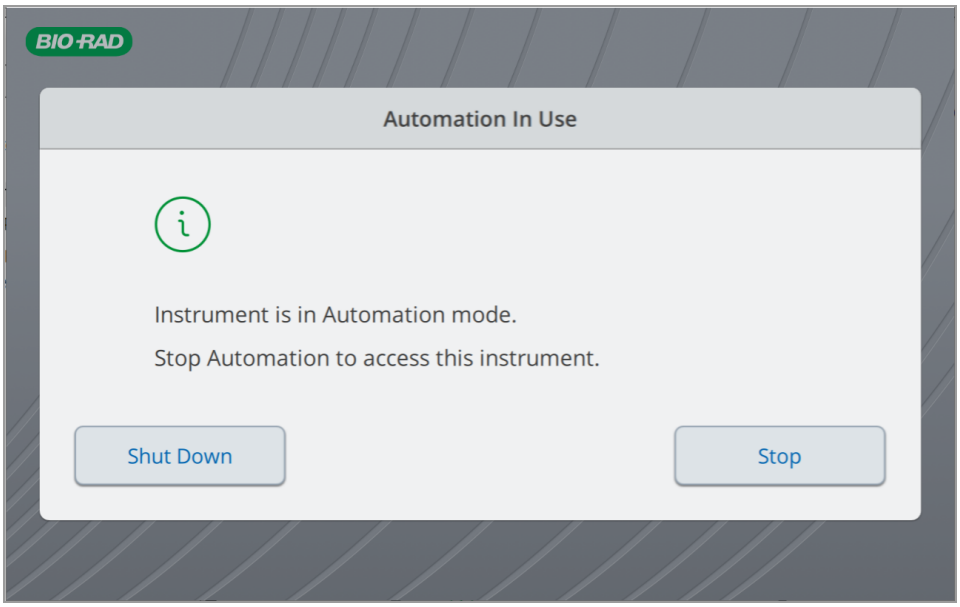
1. Log in as an Admin user or a user with admin privileges.
2. On the Home screen, tap Tools to open the Admin Tools screen.
3. On the Tools screen, tap Automation.
4. On the Automation screen, tap Start.



Tip: If the Automation user does not already exist, the system creates the Automation user and prompts the Admin to assign a password to the Automation user account.

If Enable HTTPS only was previously selected, it will remain selected until the Admin user clears the checkbox.

5. The system logs out the Admin user and launches automation. The Automation in Use screen appears, which remains visible until a user stops automation or shuts down the instrument.



Note: If a protocol is running, you cannot shut down the thermal cycler. In this case, the Shut Down button is disabled. Stop the protocol before shutting down the thermal cycler.

Important: You must enable automation in order to ping the PTC Tempo thermal cycler. Pinging the thermal cycler can help troubleshoot any API connection issues that might occur during automation.

Stopping Automation

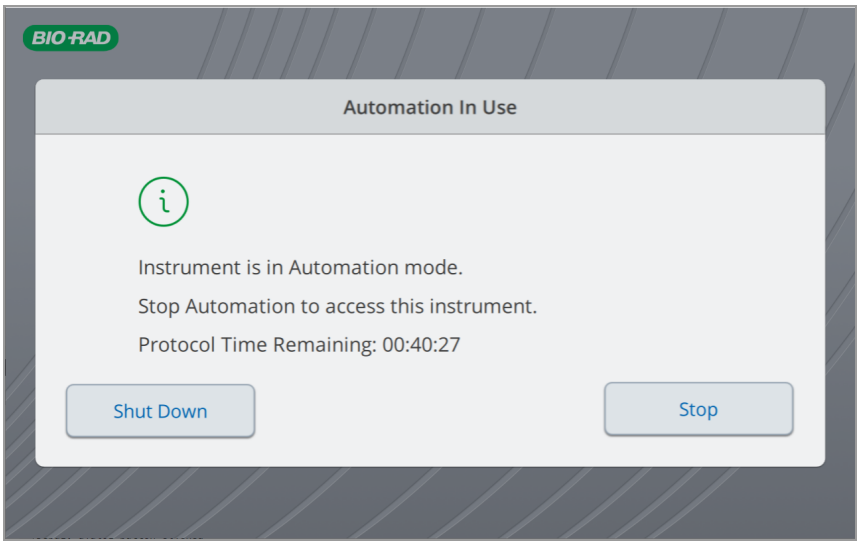
Any user can stop the automation by tapping Stop on the Automation In Use screen. When the user stops the automation, any protocol run in progress will also stop.

Note: Stopping automation also stops any active automation protocol runs.

The user also has the option to shut down the instrument during automation. A user cannot shut down the thermal cycler while a protocol run is in progress.

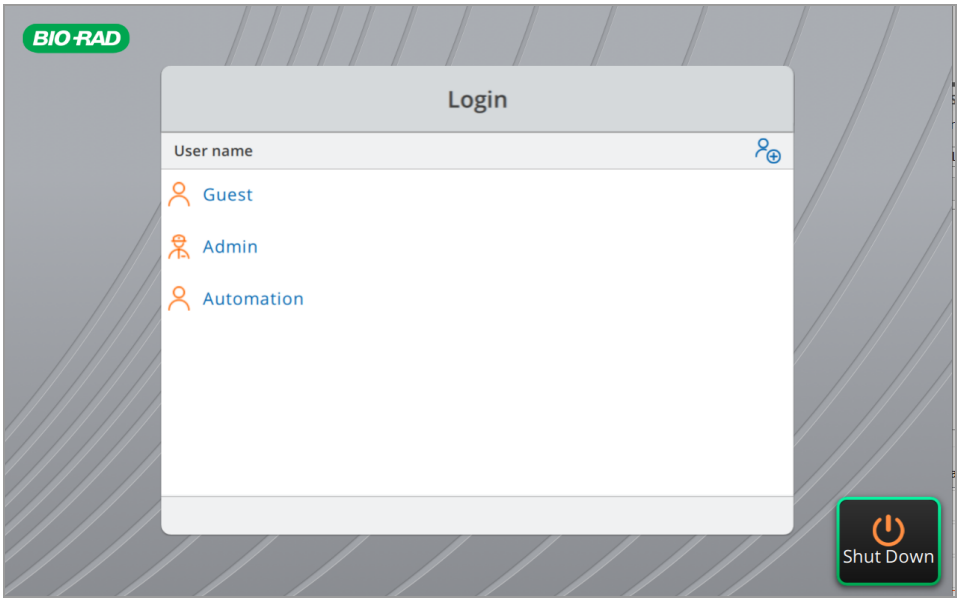
To stop PTC Tempo automation

1. In the Automation in Use message box, tap Stop.



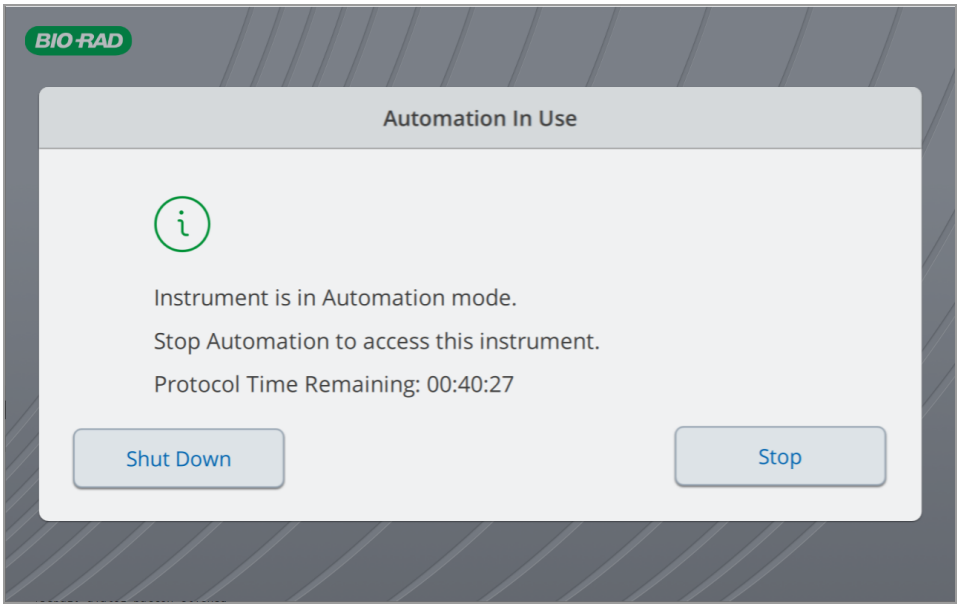
Note: Stopping automation will not remove the Automation user.

2. In the Stop Automation confirmation dialog, tap Yes to confirm that you want to stop automation.
3. The Login screen appears. To start automation, log in as an Admin user and follow the steps in Starting Automation on page 12.



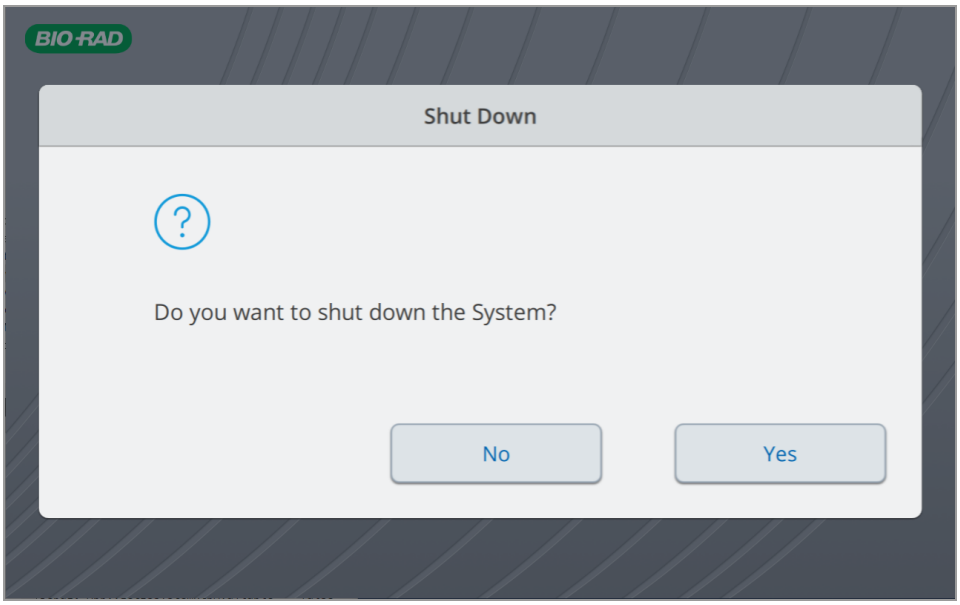
To shut down the PTC Tempo thermal cycler during automation

1. In the Automation in Use message box, tap Shut Down.



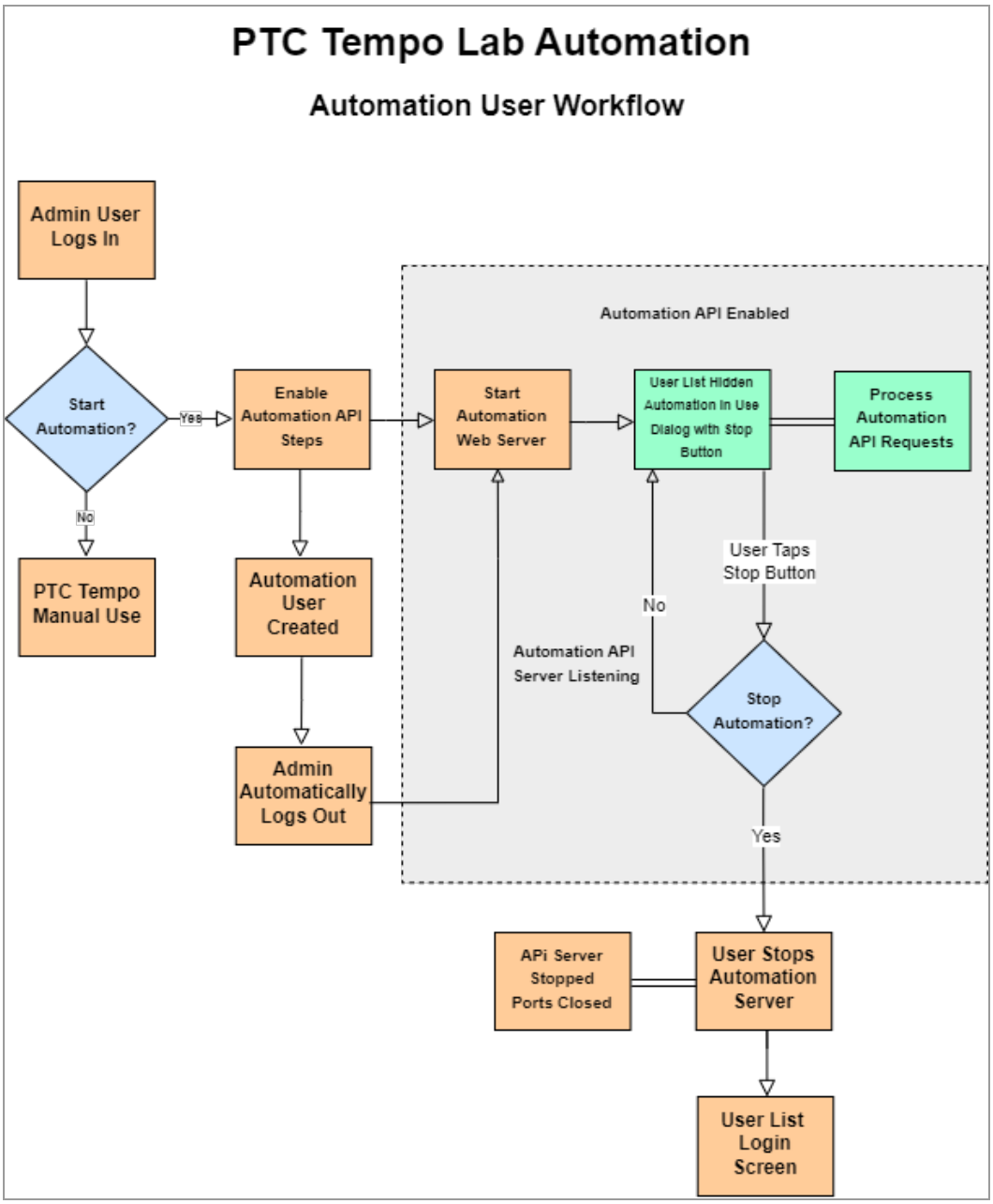
Note: If a protocol is running, you cannot shut down the thermal cycler. In this case, the Shut Down button is disabled. Stop the protocol before shutting down the thermal cycler.

2. In the Shut Down dialog, tap Yes to shut down the instrument.



Automation User Workflow

The following diagram illustrates how automation is enabled when an Automation user logs in and how automation is disabled when a user stops automation.


PTC Tempo Lab Automation
Automation User Workflow
Admin User Logs In
Start Automation?
Yes
No
Enable Automation API Steps
PTC Tempo Manual Use
Automation User Created
Admin Automatically Logs Out
Automation API Enabled
Start Automation Web Server
User List Hidden Automation In Use Dialog with Stop Button
Process Automation API Requests
User Taps Stop Button
No
Automation API Server Listening
Stop Automation?
Yes
APi Server Stopped Ports Closed
User Stops Automation Server
User List Login Screen

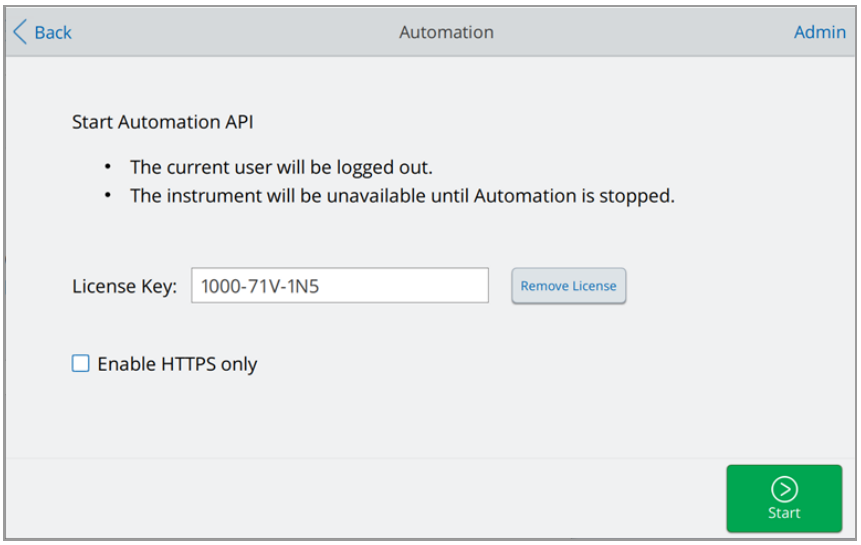
Deactivating the Automation API

An Admin user can deactivate PTC Tempo thermal cycler automation and remove the automation API license key.

WARNING! Deactivating the automation API removes the license key and disables the PTC Tempo thermal cycler's ability to interact with the automated systems in your laboratory. Take care when deactivating the automation API.

To deactivate PTC Tempo Automation

1. Log in as an Admin user or a user with admin privileges.
2. On the Home screen, tap Tools to open the Admin Tools screen.
3. On the Tools screen, tap Automation.
4. On the Automation screen, tap Remove License.



5. Click OK to confirm.
6. Automation is deactivated and the license key is removed.

Note: Deactivating automation does not remove the Automation user.

Chapter 3 PTC Tempo API Theory of Operation

This chapter explains how to use the API to run a protocol on the PTC Tempo thermal cycler in an automated environment.

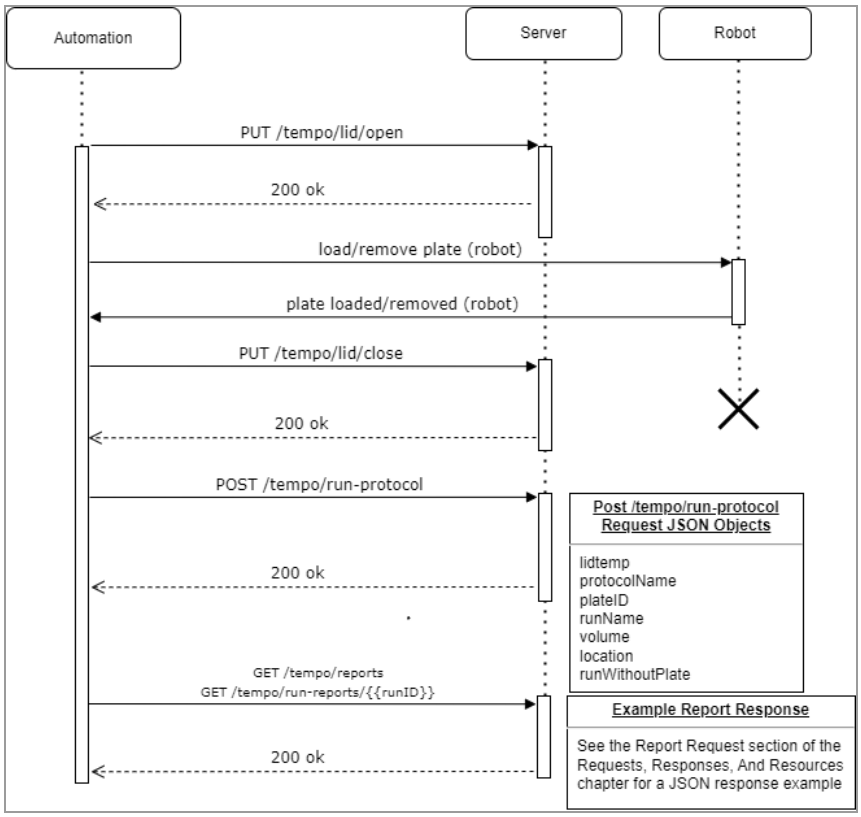
To run a protocol using the PTC Tempo API

1. Send a GET /tempo/lid API request to retrieve the status of the instrument lid
2. Send a PUT /tempo/lid/open API request to open the instrument lid.

   Important: The lid status must be open before you load a plate onto the instrument sample block.
3. An external robotic instrument loads a plate onto the PTC Tempo thermal cycler sample block.
4. Send a PUT /tempo/lid/close API request to close the instrument lid.
5. Send a POST /tempo/protocol-run API request to start the protocol run.
6. Send the following API commands during a protocol run:
   a. PUT /tempo/protocol-run/pause (pauses the run)
   b. PUT /tempo/protocol-run/resume (resumes the run)
   c. PUT /tempo/protocol-run/skip (skips a step in a run)
   d. PUT /tempo/protocol-run/stop (stops a run)
7. Send a the following report retrieval commands:
   a. GET /tempo/reports (retrieve all reports stored in the Automation user folder)
   b. GET /tempo/run-reports/count (retrieve the total number of reports stored in the Automation user folder)
   c. GET/tempo/run-reports/{{runID}} (retrieve a specific run report)
8. If necessary, the API repeats the protocol run cycle.

Workflow

The following diagram illustrates how the API implements basic plate-loading and protocol-running tasks in an automated laboratory workflow.



Chapter 4 Requests, Responses, and Resources

The PTC Tempo thermal cycler API supports the following instrument tasks:

- PTC Tempo thermal cycler lid requests
  - Open the PTC Tempo thermal cycler mechanized lid
  - Close the PTC Tempo thermal cycler mechanized lid
  - Retrieve the PTC Tempo thermal cycler lid status (opened or closed)
- PTC Tempo thermal cycler protocol run requests
  - Start a protocol run
  - Pause a protocol run
  - Resume a protocol run
  - Skip to the next step of a protocol run
  - Stop protocol run
  - Retrieve the status of an active or completed protocol run
- Retrieve PTC Tempo thermal cycler version information, response status, and list of thermal cycler errors
  - Retrieve the PTC Tempo thermal cycler version and run error information
  - Retrieve the PTC Tempo thermal cycler request response status
  - Retrieve a list of PTC Tempo thermal cycler errors
  - Clear all PTC Tempo thermal cycler errors

    Important: This request does not resolve PTC Tempo thermal cycler thermal cycler errors. To resolve errors, follow the instructions in the error message displayed on the thermal cycler.
- Retrieve a list of PTC Tempo thermal cycler protocol names
  - Retrieve a protocol in the Automation user's My Files folder
  - Retrieve a public protocol

- Retrieve a template protocol
- Retrieve PTC Tempo thermal cycler run reports
  - Retrieve a list of all reports
  - Retrieve a sub-list of reports by limit and offset
  - Retrieve a specific report
  - Retrieve the total number of reports in a list
- (Optional) Retrieve certificates and reset certificates used to encrypt HTTPS requests and responses.

Authentication

The PTC Tempo thermal cycler employs basic HTTP authentication to verify the Automation user credentials. Each endpoint request must include an authentication header with the Automation user's user name and password in order for the request to process successfully.

The request will return an error code of 401 Not Authorized if the Automation user credentials are incorrect.

Important: The PTC Tempo automation server limits the number of requests that fail authentication. After 10 requests that fail authentication, the server will block all requests made by new IP addresses and clients, and only previously authenticated IP addresses and clients can send requests and receive responses.

You can wait for the automation server to exit the blocking state after 20 minutes, or you can restart either the automation server or the PTC Tempo thermal cycler and re-attempt authentication. This feature is intended to thwart any unwanted clients from guessing your password or conducting brute force attacks.

Request and Response URL Schema

The PTC Tempo thermal cycler API requests resources in a URL format and returns responses in a JSON format.

Requests

The PTC Tempo thermal cycler API sends resource requests in a URL format. The URL contains the following variable parts:

- HTTP or HTTPS protocol
- Thermal cycler IP address

- Endpoint (example: "/tempo/lid")
- Resource ID, depending on the endpoint
- One or more parameters, depending on the endpoint

For example, the URL request to retrieve a specific quantity of reports is https://[IP address]/tempo/run-reports?limit=2. The base service URL is https://[IP].

Responses

The PTC Tempo thermal cycler API responds with a standard HTTP status code and an optional body in JSON format. If the response contains an error message, it will always be at the "error" key within the body.

For example:

Status Code	Response Body

400 Bad Request	{
	"error" : "Error in JSON. Could not find lidTemp."
	}

Lid Requests

These API requests control the PTC Tempo thermal cycler lid and retrieve its status (opened or closed).

Important: These JSON response values can appear following the "lid" key: Opened, Opening, Closing, Closed.

PUT Open Lid

Category	Description	
Resource	PUT /tempo/lid/open	
Summary	This request opens the PTC Tempo thermal cycler lid. If the API returns "error" as a value in the lid or instrument status parameters of the JSON response, send a GET /tempo/errors request to identify the error.	
Example Request	https://[*IP address*]/tempo/lid/open	

Example Response	<pre>{
	  "lid": "opening",
	  "status": "idle"
	}</pre>	

Status Codes	**Status Code**	**Description**
	`200 OK`	Request successfully executed
	`400 Bad Request`	This error appears if the request body does not contain the "lid" key or has any value besides "open" or "close"

PUT Close Lid

Category	Description
Resource	PUT /tempo/lid/close
Summary	This request closes the PTC Tempo thermal cycler lid. If the API returns "error" as a value in the lid or instrument status parameters of the JSON response, send a GET /tempo/errors request to identify the error.
Example Request	https://[*IP address*]/tempo/lid/close

Example Response	{
	"lid": "closing",
	"status": "idle"
	}

Status Codes	**Status Code** — **Description** 200 OK — Request successfully executed 400 Bad Request — This error appears if the request body does not contain the "lid" key or has any value besides "open" or "close"

GET Lid Status

Category	Description
Resource	GET /tempo/lid
Summary	This request returns the current status of the PTC Tempo thermal cycler lid: opening, opened, closing, or closed. **Note:** Poll the response of this endpoint until the lid status returns "opened" or "closed" after sending a PUT /tempo/lid/open or a PUT /tempo/lid/close request.
Request JSON Objects	N/A
Example Request	https://[*IP address*]/tempo/lid

Example Response	`{`
	`  "lid": "closed",`
	`  "status": "idle"`
	`}`

Status Codes	**Status Code** — **Description** `200 OK` — Request successfully executed

Protocol Run Requests

These APIs instruct the PTC Tempo thermal cycler to start, pause, resume, skip, and stop protocol runs and retrieve the instrument status (idle or running).

POST Start Protocol Run

Category	Description
Resource	POST /tempo/protocol-run
Summary	This request loads and starts a protocol run from the Public folder, the Templates folder, or an Automation user folder. If the request executes successfully, the API returns a 200 OK status and a brief summary of the protocol run details. If the API returns "error" as a value in the lid or instrument status parameters of the JSON response, you can send a GET /tempo/errors request to identify the error. **Note:** Some protocol parameters are required, and others are optional. **Note:** For protocols set with an infinite hold, the "hold" and "remaining" time status values start at zero and will continue to increment. Disregard the “hold” and “remaining” time response values in this situation.

Request JSON Objects (Example)	{
	"lidTemp": 40,
	"protocolName": "IPRF1KB",
	"plateID": "barcode",
	"runName": "example",
	"volume": 8,
	"location": "public",
	"runWithoutPlate": true
	}

Parameters	Object	Description	Type
	lidTemp	(Optional). The PTC Tempo thermal cycler lid temperature for a designated protocol run. The thermal cycler lid temperature at the start of a protocol run is reported in the JSON response. The request can contain the following lidTemp object values: ■ off – the thermal cycler lid remains at the ambient temperature when the run starts.	Integer, String

Category		Description	
		■ default – the thermal cycler lid uses the default lid temperature for the device. For most devices, the default lid temperature is 105ºC, but for 384 well instruments, the default temperature is 95ºC. ■ If the lidTemp value is missing, the protocol run starts using the temperature from the protocol file. ■ If the lidTemp value is an integer, the protocol run starts with that value (even if it differs from the temperature in the protocol file). ■ If the lidTemp is set to below the minimum allowed value for the instrument, the temperature is set to the minimum. ■ If the lidTemp is set to above the maximum allowed value for the instrument, the temperature is set to the maximum.	
	protocolName	(Required). The protocol's alphanumeric identifier	String

Category		Description	
	plateID	(Optional) The alphanumeric sample plate identifier (entered manually or read by a bar code scanner)	String
	runName	(Optional). The name of the run file, separate from the protocol name	String
	volume	(Optional). The sample volume amount in milliliter The volume can be "default," an integer, or the volume key can be missing. ■ If the value is "default," the thermal cycler uses the default volume for that instrument (20 µl for 96-well instruments, 10 µl for 384-well instruments, and 50 µl for Deepwell instruments). ■ If the volume value is an integer, the protocol run starts with that volume (even if it differs from the volume in the protocol file). ■ If the volume value is missing, the protocol run starts using the volume from the protocol file. ■ If the volume is set to below the minimum allowed value for the instrument, the volume is	Integer

Category		Description	
		set to the minimum. ■ If the volume is set to above the maximum allowed value for the instrument, the volume is set to the maximum.	
	location	(Required). The folder where the protocol is located. ■ If the location is "public", locates the protocol in the Public folder ■ If the location is "user", locates the protocol in the Automation user's My Files folder ■ If the location is "templates", locates the protocol in the Templates folder	String

Category		Description	
	runWithoutPlate	(Optional) If the value is "true," this object requests the thermal cycler to start the run without a plate loaded onto the sample block. The runWithoutPlate value can be "true," "false," or missing. ■ If the value is "true," the PTC Tempo thermal cycler software allows the Automation user to start a run without a plate loaded onto the thermal cycler sample block. ■ If the value is "false," or is missing, the PTC Tempo thermal cycler software requires that a plate is loaded onto the thermal cycler sample block.	String

Example Request	**URL**
	https://[IP address]/tempo/protocol-run
			JSON Objects
			{
			"lidTemp": 105,
			"protocolName": "IPRF1KB",
			"plateID": "barcode",
			"runName": "example",
			"volume": 10,
			"location": "public",
			"runWithoutPlate": "true"
			}

<table>
<tr><th>Category</th><th colspan="3">Description</th></tr>
<tr><td>Example Response (Idle)</td><td colspan="3"><pre>{
  lid": "closed",
  "lidTemp": 40,
  "status": "idle",
  "steps": 4,
  "time": "2023-02-08T15:01:29-08:00",
  "volume": 8
}</pre></td></tr>
<tr><td>Status Codes</td><td>Status Code</td><td>Description</td><td>Example JSON</td></tr>
<tr><td></td><td>200 OK</td><td>Request successfully executed</td><td>N/A</td></tr>
<tr><td></td><td>400 Bad Request</td><td>This request is invalid when:
The key type (number or string) is not correct.
The JSON body does not contain values for all required keys: protocolName, location
The JSON object value runWithoutPlate is "false" or is missing, and not plate is loaded onto the thermal cycler sample block.
If the thermal cycler is not in an "idle" state when the Automation user tried to start a run.</td><td><pre>{
  "error": Error in JSON body. Lidtemp should be off, default, missing, or an integer"
}</pre>or<pre>{
  "error": "Error in JSON. Could not find location."
}</pre>or<pre>{
"error": "Instrument may not start without a plate unless client sets runWithoutPlate value."
}</pre></td></tr>
</table>

Category		Description	
	404 Not Found	Error, server cannot find the requested resource. This error appears when the protocolName value does not match any existing protocols	{ "error": "Protocol was not found", location": "Public", "protocolName": "A12345" }
	500 Internal Server Error	Error, server encountered a request it cannot handle. This error appears when the software received a valid request to stop, but was unable to send a request to the instrument firmware	{ "error": "Error occurred when starting protocol run." "location": "Public", "protocolName": "A12345" }
	501 Not Implemented	Error, server does not support request functionality. This error appears if the location object value is "network"	{ " Error in JSON body. Network location is not supported in the Automation API." }

PUT Protocol Run (Pause)

Category	Description	
Resource	PUT /tempo/protocol-run/pause	
Summary	This request pauses a protocol run.	
Request JSON Objects	N/A	
Example Request	https://[*IP address*]/tempo/protocol-run/pause	
Example Response	N/A	
Status Codes	**Status Code**	**Description**
	`200 OK`	Request successfully executed
	`400 Bad Request`	This error appears when a pause request is sent, and no protocol is running
	`404 Not Found`	This error appears if a method other than a PUT request is sent
	`500 Internal Server Error`	This error appears when the software received a valid request to pause, but was unable to send a request to the instrument firmware

PUT Protocol Run (Skip Step)

Category	Description	
Resource	PUT /tempo/protocol-run/skip	
Summary	This request skips the current step in a protocol run.	
Request JSON Objects	N/A	
Example Request	https://[*IP address*]/tempo/protocol-run/skip	
Example Response	N/A	
Status Codes	**Status Code**	**Description**
	`200 OK`	Request successfully executed
	`400 Bad Request`	This error appears when a skip step request is sent and no protocol is running
	`404 Not Found`	This error appears if a method other than a PUT request is sent
	`500 Internal Server Error`	This error appears when the software received a valid request to skip the step, but was unable to send a request to the instrument firmware

PUT Protocol Run (Resume)

Category	Description	
Resource	PUT /tempo/protocol-run/resume	
Summary	This request resumes a paused protocol run.	
Request JSON Objects	N/A	
Example Request	https://[*IP address*]/tempo/protocol-run/resume	
Example Response	N/A	
Status Codes	**Status Code**	**Description**
	`200 OK`	Request successfully executed
	`400 Bad Request`	This error appears when a resume request is sent but the protocol run is not paused.
	`404 Not Found`	This error appears if a method other than a PUT request is sent
	`500 Internal Server Error`	This error appears when the software received a valid request to resume, but was unable to send a request to the instrument firmware

PUT Protocol Run (Stop)

Category	Description	
Resource	PUT /tempo/protocol-run/stop	
Summary	This request aborts a protocol run.	
Request JSON Objects	N/A	
Example Request	https://[*IP address*]/tempo/protocol-run/stop	
Example Response	N/A	
Status Codes	**Status Code**	**Description**
	`200 OK`	Request successfully executed
	`400 Bad Request`	This error appears when a stop request is sent and no protocol is running
	`404 Not Found`	This error appears if a method other than a PUT request is sent
	`500 Internal Server Error`	This error appears when the software received a valid request to stop, but was unable to send a request to the instrument firmware

GET Protocol Run Status

Category	Description
Resource	GET /tempo/protocol-run
Summary	This request determines whether the instrument is in a "running" or "idle" state. If the instrument is running, the JSON response displays the active protocol parameters. **Note:** For protocols set with an infinite hold, the "hold" and "remaining" time status values start at zero and will continue to increment. Disregard the “hold” and “remaining” time response values in this situation.
Request JSON Objects	N/A
Example Request	https://[*IP address*]/tempo/protocol-run

Example Response (Idle)	{
	"lid": "closed",
	"status": "idle",
	"time": "2023-05-03T09:04:36-07:00"
	}

Example Response (Running)	{
	"lid": "closed",
	"status": "running",
	"time": "2023-05-03T09:04:36-07:00"
	}

Status Codes	**Status Code**: 200 OK — **Description**: Request successfully executed

Thermal Cycler Status and Error Requests

These requests determine whether the PTC Tempo thermal cycler is responding to API requests, provides a list of errors that occurred during an automated protocol run, and clears out any error information stored in the PTC Tempo thermal cycler.

GET Tempo Response Status

Category	Description	
URL	GET /tempo/ok	
Summary	This request verifies that the thermal cycler is responding to API requests. The server responds with a 200 OK status code if the thermal cycler is responding.	
Example Request	https://[*IP address*]/tempo/ok	
Status Codes	**Status Code**	**Description**
	`200 OK`	Success, current request has completed

GET Tempo Information

Category	Description
URL	GET /tempo
Summary	This request returns PTC Tempo thermal cycler version information.
Example Request	https://[*IP address*]/tempo

Example Response	{ "device":
	"details": {
	"automationAPI": "1.0.0",
	"diskFreeSpace": "127,829 MB",
	"firmwareVersion": "1.2.3 ",
	"lidFirmwareVersion": "2.1.3 ",
	"percentageDiskFreeSpace": "26%",
	"powerManagerFwVersion": "3.2.3 ",
	"softwareVersion": "2.0.0.3",
	"systemImageVersion": "N/A"
	},
	"instrumentName": "C2000",
	"model": "PTCTempo96",
	"serialNumber": "CC00622",
	"type": "PTCTempo",
	"ver": "1.2.3 "
	},
	"lid": "closed",
	"status": "idle",
	"time": "2023-05-03T09:12:26-07:00"
	}

Status Codes	**Status Code** — **Description** 200 OK — Success, current request has completed

GET Tempo Errors

Category	Description
URL	GET /tempo/errors
Summary	This request provides a list of thermal cycler and thermal cycler lid errors. The response will always contain the cyclerFaultCount and lidFaultCount keys, regardless of the number of faults. The response will also contain cyclerFaults and lidFaults keys and their arrays only if the counts are greater than zero. The following can also occur: ■ The lidFaultCount and cyclerFaultCount values might not match the number of entries in the lidFaults and cyclerFaults array if thermal cycler software is unable to retrieve all of the fault information from the firmware due to an error in the firmware. ■ The lid status might still appear as “error” even if no lid faults are reported. This occurs if the client calls a PUT /tempo/errors/clear endpoint, but the firmware isn’t able to clear all errors. There might be a physical problem with the lid that keeps the instrument or lid in an error state even after attempting to clear faults.
Example Request	https://[*IP address*]/tempo/errors

Category	Description

Example Response	{
	"cyclerFaultCount":2
	"cyclerFaults": [
	(
	"block": 0,
	"description": "Front zone temperature error",
	"info": 0,
	"number": 307,
	"severity": "abort",
	"timestamp": "Tue Mar 14 20:33:06 2023"
	}
	"block": 0,
	"description": "Front zone temperature error",
	"info": 0,
	"number": 303,
	"severity": "abort",
	"timestamp": "Tue Mar 14 20:33:06 2023"
	}
	],
	"lidFaultCount": 1,
	"lidFaults": [
	{
	"block": 0,
	"description": "Hinge motor over current",
	"info": 0,
	"number": 10012,
	"severity": "warning"
	"timestamp": "Tue Mar 14 20:33:06 2023"
	}
	]
	)

Status Codes	**Status Code** / **Description** 200 OK — Success, current request has completed 500 Internal Server error — Error, software cannot query the firmware for fault information. Despite this error, the response will contain information about as many fault events as it can obtain from the firmware.

PUT Clear Tempo Errors

Category	Description	
URL	PUT /tempo/errors/clear	
Summary	This request clears any error information stored in the PTC Tempo thermal cycler. The response will return a 200 OK status code. The lid status may still appear as “error” even if no lid faults are reported. This occurs if the client calls a PUT /tempo/errors/clear endpoint, but the firmware is not able to clear all errors. There might be a physical problem with the lid that keeps the instrument or lid in an error state even after attempting to clear faults. **Important:** This request might not clear the same errors in the thermal cycler firmware. To clear thermal cycler firmware errors, restart the PTC Tempo thermal cycler. If the error persists in the firmware, contact Bio-Rad Technical Support.	
Example Request	https://[*IP address*]/tempo/errors/clear	
Status Codes	**Status Code**	**Description**
	`200 OK`	Success, current request has completed
	`500 Internal Server Error`	Error, software cannot send the request to the firmware to clear faults

Retrieve a List of Protocol Names

This section explains how to use the PTC Tempo API to retrieve a list of protocols in the Public folder, Automation user folder, or the Templates folder.

Tip: For information about protocol locations, see the PTC Tempo thermal cycler Instrument Guide.

GET Default Templates

Category	Description
Resource	GET /tempo/protocols/templates
Summary	This request retrieves a list of protocol templates.
Request JSON Objects	N/A
Example Request	https://[*IP address*]/tempo/protocols/templates

Example Response	{
	"location": "Templates",
	"protocolNames": [
	{
	"lastModified": "2022-12-15T22:37:34",
	"name": "IPRF15KB"
	},
	{
	"lastModified": "2022-12-15T22:37:34",
	"name": "IPRF1KB"
	},
	{
	"lastModified": "2022-12-15T22:37:34",
	"name": "IPRF8KB"
	}
	]
	}

Status Codes	**Status Code** — **Description** 200 OK — Success, current request has completed

GET Public Protocols

Category	Description
Resource	GET /tempo/protocols/public
Summary	This request retrieves a list of previously completed public protocols.
Request JSON Objects	N/A
Example Request	https://[*IP address*]/tempo/protocols/public

Example Response	<pre>{ "location": "public",
	"protocolNames": [
	{ "lastModified": "2022-09-12T18:12:58Z",
	"name": "IPRF15KB"
	},
	{ "lastModified": "2022-09-12T18:12:58Z",
	"name": "IPRF1KB"
	},
	}</pre>

Status Codes	**Status Code** — **Description** 200 OK — Request successfully executed

GET User Protocols

Category	Description
Resource	GET /tempo/protocols/user
Summary	This request retrieves a list of protocols saved in the PTC Tempo thermal cycler Automation user's folder (the My Files folder)
Request JSON Objects	N/A
Example Request	https://[*IP address*]/tempo/protocols/user
Example Response	{ "location": "Automation", "protocolNames": [{ "lastModified": "2022-09-12T18:12:58Z", "name": "IPRF15KB" }, { "lastModified": "2022-09-12T18:12:58Z", "name": "IPRF1KB" }, }
Status Codes	**Status Code** — **Description** 200 OK — Request successfully executed

Report Requests

These API requests instruct the PTC Tempo thermal cycler to generate a list of protocol run reports, to retrieve a range of run reports, and for an Admin user to retrieve the Automation user's protocol run report.

Retrieving a Range of Run Reports

The PTC Tempo thermal cycler can retrieve a list of run reports. The "?limit" parameter specifies the number of run reports in the range. The "?offset" parameter specifies the number of records to skip so they are not included in the report range.

GET Run Reports

Category	Description		
Resource	GET /tempo/reports		
Summary	This request retrieves a list of all PTC Tempo thermal cycler run reports stored in the Automation user's folder. You can add the ?offset and ?limit parameters to this endpoint to retrieve customized set of run reports.		
Request JSON Objects	N/A		
Optional Parameters	**Parameter**	**Description**	**Example**
	?limit	Retrieves a specific number of reports (up to 10) in an array of reports.	https://[IP address]:tempo/run-reports?limit=3
	?offset	Specifies how many reports the system should skip (and which report to start with) before generating an array (or the next array) of reports.	https://[IPaddress]/tempo/run-reports?limit=3&offset=20
	?offset and ?limit	Retrieves an array of reports	https://[IP Address]/tempo/run-reports?limit=3&offset=20
Example Request	https://[*IP address*]/tempo/reports		
Example Response	`{ "reports": { "blockName": "", "loggedInUser": "Automation", "plateID": "123436", "protocolName": "IPRF1KB", "runDate": "2022-08-25T18:25:04.000", "runID": "caf133de-e1cb-458d-89d7-7aef302be7ec", "runName": "Grow DNA" } }`		

Category	Description	
Status Codes	**Status Code**	**Description**
	`200 OK`	Request successfully executed

GET Total Number of Run Reports

Category	Description
Resource	GET /tempo/run-reports/count
Summary	This request allows for a user logged into the PTC Tempo thermal cycler to retrieve the total number of run reports in an Automation user's folder. The response status is always 200 OK, even if there are no reports. **Note:** The GET Reports response returns a list of all reports in the Automation user's folder. Each report contains the runID parameters, which you can get and use in this endpoint's resource.
Request JSON Objects	N/A
Example Request	https://[*IP address*]/tempo/run-reports/count

Example Response	`{`
	`  "count": 25,`
	`  "username": "Automation"`
	`}`

Status Codes	**Status Code** — **Description** `200 OK` — Request successfully executed

GET Report with a Specific Run Identifier

Category	Description		
Resource	GET /tempo/run-reports/{{runID}}		
Summary	This request retrieves a protocol run report by using the protocol's runID. **Note:** The GET Reports response returns a list of all reports in the Automation user's folder. Each report contains the runID parameters, which you can get and use in this endpoint's resource. This request allows for a user logged into the PTC Tempo thermal cycler to retrieve a protocol run report by using the protocol's runID.		
Request JSON Object	N/A		
Parameters	**Object**	**Description**	**Type**
	runID	The protocol run identifier	String
Example Request	https://[*IP address*]/tempo/run-reports/run-reports/50aba0f9-911d-48bd-bf2d-5e51f3eac727		

Category	Description

Example Response	{
	"run": {
	"elapsedTime": "35",
	"endDateTime": "2023-03-01T17:41:26-08:00",
	"errorText": "No errors reported.",
	"instrumentDetails": {
	"blockName": "",
	"firmwareVersion": "1.2.0 ",
	"instrumentName": "C2000",
	"instrumentType": "PTC Tempo 384",
	"serialNumber": "CC00622",
	"softwareVersion": "0.0.0.0 Local GC-NoSha"
	},
	"labLocation": "",
	"labName": "",
	"plateID": "barcod2",
	"protocol": {
	"lidTemp": {
	"mode": "custom",
	"temp": 105
	},
	"protocolName": "ProtocolRunTest",
	"steps": [
	{
	"temp": 98,
	"time": 30,
	"type": "temp"
	},
	{
	"temp": 98,
	"time": 5,
	"type": "temp"
	},
	],
	"vol": 10
	},
	"protocolName": "ProtocolRunTest",
	"runDetails": [
	{
	"additionalDetails": "",
	"dateTime": "2023-03-01T17:39:14-08:00",
	"duration": "00:00:30",
	"repeat": "1",
	"stepNumber": "1",
	"stepSettings": "98.0"

Category	Description


```
      },
      {
        "additionalDetails": "Skip.",
        "dateTime": "2023-03-01T17:39:32-08:00",
        "duration": "--",
        "repeat": "1",
        "stepNumber": "1",
        "stepSettings": "--"
      },
      {
        "additionalDetails": "",
        "dateTime": "2023-03-01T17:39:32-08:00",
        "duration": "00:00:05",
        "repeat": "1",
        "stepNumber": "2",
        "stepSettings": "98.0"
      },
      {
        "additionalDetails": "Protocol completed.",
        "dateTime": "2023-03-01T17:41:26-08:00",
        "duration": "--",
        "repeat": "1",
        "stepNumber": "7",
        "stepSettings": "--"
      }
    ],
    "runErrorState": "1",
    "runName": "example2",
    "runStatus": "Completed without errors",
    "runStatus2": "",
    "startDateTime": "2023-03-01T17:39:14-08:00",
    "userName": "Automation"
  }
}
```


Status Codes

Status Code	Description	Example JSON
200 OK	Request successfully executed	N/A
404 Not Found	Error, runID not found in run reports	{ "error":"runID not found in run reports. <2e1bb94c-ecb8-4c00-8778-

Category	Description	
		1be64383da3" }

Certificate Requests

These optional requests:

- Retrieve the public self-signed RSA 2048 public certificate used for TLS encryption (GET /tempo/certificate)
- Generate and reset the TLS self-signed RSA 2048 certificate key pair used to encrypt all HTTPS requests and responses.

GET Certificate

Category	Description	
Resource	GET /tempo/certificate	
Summary	This request retrieves the public self-signed RSA 2048 public certificate used for TLS encryption.	
Example Request	https://[*IP address*]/tempo/certificate	
Response	Response displays certificate in a text/plain content-type format.	
Status Codes	**Status Code**	**Description**
	`200 OK`	Request successfully executed

POST Reset Certificate

Category	Description	
Resource	POST /tempo/certificate	
Summary	This request generates and resets the TLS self-signed RSA 2048 certificate key pair used to encrypt all HTTPS requests and responses.	

Request JSON Object	`{`
	`"certificate": "reset"`
	`}`	

Example Request	https://[*IP address*]/tempo/certificate	
Response	Response displays certificate in a text/plain content-type format.	
Status Codes	**Status Code**	**Description**
	`200 OK`	Request successfully executed

Appendix A Status Codes

All PTC Tempo thermal cycler Security Edition API resources return one of six codes. Appendix A lists the codes and their most common definition.

Table 1. PTC Tempo API status codes

Code	Definition
`200 OK`	Request successfully executed
`400 Bad Request`	Error, request content or context is invalid
`401 Unauthorized`	Error, request lacks valid authentication credentials
`404 Not Found`	Error, server cannot find the requested resource
`500 Internal Server Error`	Error, server encountered a request it cannot handle
`501 Not Implemented`	Error, server does not support the request functionality

Bio-Rad Laboratories, Inc.

Life Science Group

Website *bio-rad.com* ***USA*** *1 800 424 6723* ***Australia*** *61 2 9914 2800* ***Austria*** *00 800 00 24 67 23* ***Belgium*** *00 800 00 24 67 23*
Brazil *4003 0399* ***Canada*** *1 905 364 3435* ***China*** *86 21 6169 8500* ***Czech Republic*** *00 800 00 24 67 23* ***Denmark*** *00 800 00 24 67 23*
Finland *00 800 00 24 67 23* ***France*** *00 800 00 24 67 23* ***Germany*** *00 800 00 24 67 23* ***Hong Kong*** *852 2789 3300*
Hungary *00 800 00 24 67 23* ***India*** *91 124 4029300* ***Israel*** *0 3 9636050* ***Italy*** *00 800 00 24 67 23* ***Japan*** *81 3 6361 7000*
Korea *82 080 007 7373* ***Luxembourg*** *00 800 00 24 67 23* ***Mexico*** *52 555 488 7670* ***The Netherlands*** *00 800 00 24 67 23*
New Zealand *64 9 415 2280* ***Norway*** *00 800 00 24 67 23* ***Poland*** *00 800 00 24 67 23* ***Portugal*** *00 800 00 24 67 23*
Russian Federation *00 800 00 24 67 23* ***Singapore*** *65 6415 3188* ***South Africa*** *00 800 00 24 67 23* ***Spain*** *00 800 00 24 67 23*
Sweden *00 800 00 24 67 23* ***Switzerland*** *00 800 00 24 67 23* ***Taiwan*** *886 2 2578 7189* ***Thailand*** *66 2 651 8311*
United Arab Emirates *36 1 459 6150* ***United Kingdom*** *00 800 00 24 67 23*

10000156544 Ver A (12019776) US/EG *23-0363 0423 Sig 0123*