Connecting to DTP

Introduction

Connecting to DTP allows you to obtain the network license and send local analysis data to a centralized database. You can send the following information:

- Static Analysis and Metrics results (including suppression details)
- Unit Testing results (including traceability information)
- Line Coverage

DTP aggregates, analyzes, and prioritizes the data to help you optimize development processes, focus on the impact of changed code, and demonstrate full compliance traceability. See About the Parasoft Development Testing Workflow for details.

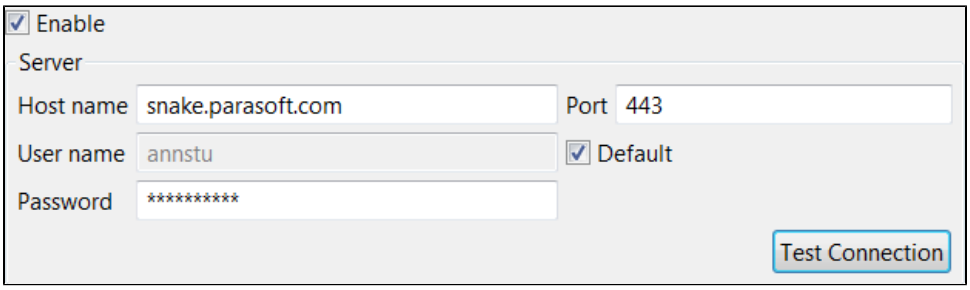
Version compatibility

If you connect your C/C++test instance to DTP, ensure that the DTP version your connecting to supports your C/C++test version. For example, upgrading C/C++test to version *10.4.3* requires upgrading DTP to the corresponding *5.4.3* version.

Connecting to DTP in the GUI

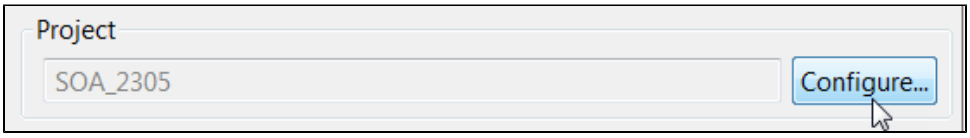
To connect Parasoft C/C++test to a Parasoft DTP server:

1. In your IDE, click **Parasoft** in the menu bar and choose **Options** (Visual Studio) or **Preferences** (Eclipse).
2. Choose **DTP**.
3. Check **Enable** and enter the DTP server information:



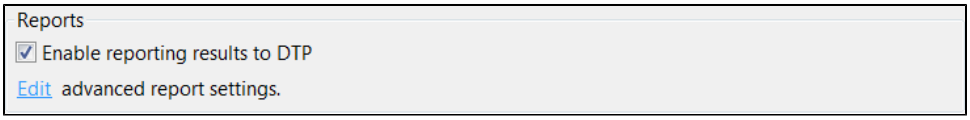
You can click **Test Connection** to verify the settings.
If you use OpenID Connect for authentication on DTP, provide the host name and the port of your DTP server, then configure the connection with the OpenID Connect server; see Configuring OpenID Connect in the UI. When you successfully connect to your OpenID Connect server, the DTP page will automatically display the information.

4. In the **Project** area, click the **Configure** button to open a dialog with all projects that are available in DTP and select the project you are currently working on.



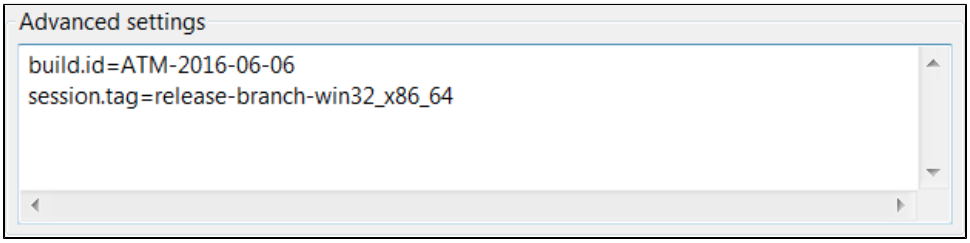
5. The active license is displayed in the **License** area. Depending on the license type, the license may be automatically configured when the connection to DTP is established. Click the **Configure** link to manually configure your license options if applicable.

6. In the **Reports** area, check **Enable reporting results to DTP** to enable sending static analysis findings, unit test results and coverage information to DTP.



   To publish reports to DTP, a valid license with one of the following options is required: DTP Publish or Automaton.

   You can click the **Edit** link to configure advanced report settings (`key=value`):

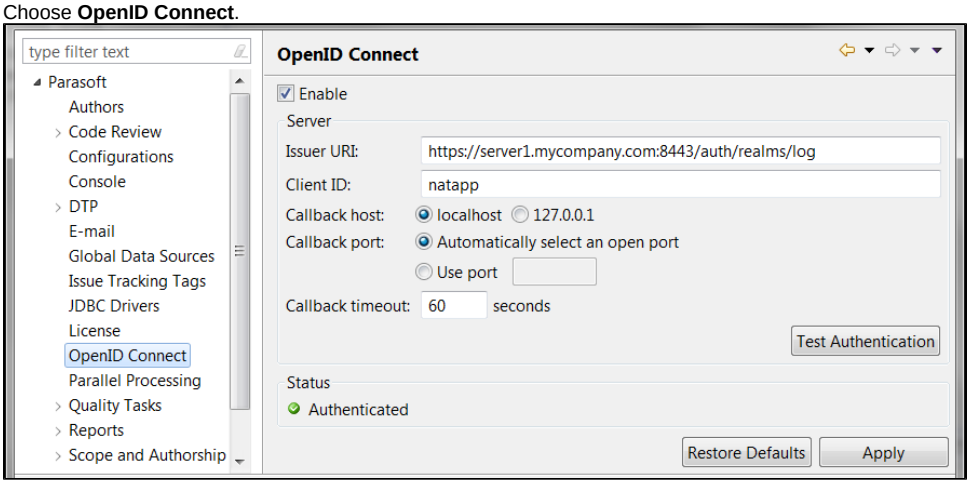


7. Click **Apply** to save the settings.

Configuring OpenID Connect in the GUI

DTP ships with support for OpenID Connect user authentication (see the DTP User Guide for details). If OpenID Connect is enabled for your DTP server, you must configure C/C++test to authenticate users via OpenID Connect.

1. In your IDE, click **Parasoft** in the menu bar and choose **Options** (Visual Studio) or **Preferences** (Eclipse).
2. Choose **OpenID Connect**.

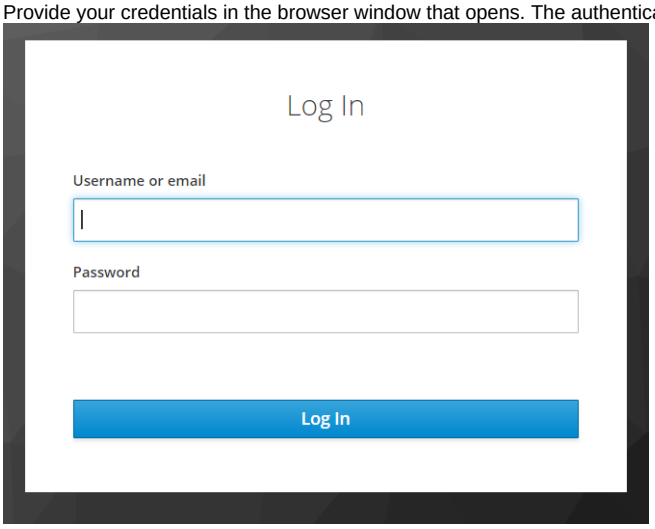


3. Check **Enable**.
4. Configure the following options:
   Issuer URI: The URI of your OpenID Connect server.
   Client ID: The ID registered on your OpenID Connect server.
   Callback host: The local callback host required to communicate with the OpenID Connect server. The following options are available:
   - **localhost**: The localhost address will be used for communication.
   - **127.0.0.1**: The loopback IP address 127.0.0.1 will be used for communication.

   Callback port: The callback port number for communication with the OpenID Connect server. The following options are available:
   - **Automatically select an open port**: Automatically selects an open port (recommended).
   - **Use port**: Allows you to manually specify the port number.

   Callback timeout: Specifies, in seconds, the maximum time the browser will wait for user credentials (see step 6).
5. Click **Test Authentication** or **Apply** to open the OpenID Connect authentication page in your browser.

6. Provide your credentials in the browser window that opens. The authentication page may resemble the following:



7. Close the browser window when the authentication confirmation is displayed and continue in your IDE.
8. Click **Apply** to apply the changes.

The **Status** panel displays the current OpenID Connect authentication status.

Connecting to DTP in the Command Line

To configure the connection to DTP when you perform testing with C/C++test in the command line mode:

1. Specify the connection settings in a custom `.properties` file.
2. Provide the path to the `.properties` file with the `-localsettings` command line option:

```
cpptestcli -localsettings [PATH_TO_.PROPERTIES_FILE]
```

Required Settings

The following settings are required to configure the connection:

- `dtp.enabled=true` - Enables the connection to DTP server
- `dtp.server=[HOST]` - Specifies the host name of the DTP server
- `dtp.port=[PORT]` - Specifies the port number of the DTP server. Commonly used port numbers are 443 and 8443.
- `dtp.user=[USERNAME]` - Specifies the username for DTP server authentication.
- `dtp.password=[PASSWORD]` - Specifies the password for DTP server authentication.
- `report.dtp.publish=true` - Enables reporting results to DTP server.

We highly recommend that you use an encoded password to ensure successful authentication and increase the level of security; see Creating an Encoded Password.

Optional Settings

- `dtp.project=[NAME]` - Specifies the name of the DTP project.
- `build.id=[IDENTIFIER]` - Specifies a build identifier used to label results.
- `session.tag=[TAG]` - Specifies a tag for signing result from the test session.
- `dtp.additional.settings=[KEY1\=VALUE1\nKEY2\=VALUE2...]` - Specifies advanced settings for reporting results to DTP.

Configuring OpenID Connect in the Command Line

DTP ships with support for OpenID Connect user authentication (see the DTP User Guide for details). If OpenID Connect is enabled for your DTP server, you must configure C/C++test to authenticate users via OpenID Connect.

Configure the following settings in the `.properties` file where the connection to your DTP server is configured:

- `oidc.enabled=true` - Enables user authentication via OpenID Connect.
- `oidc.issuer.uri=[URI]` - Specifies the URI of the OpenID Connect server where your DTP is registered.
- `oidc.client.id=[ID]` - Specifies the ID provided by your OpenID Connect server.
- `oidc.client.secret=[PASSWORD]` - Specifies the password provided by your OpenID Connect server.
- `oidc.keystore=[PATH]` - Specifies the path to the keystore file that stores the certificate to authenticate the user on the OpenID Connect server.
- `oidc.keystore.password=[PASSWORD]` - Specifies the password to the the keystore file that stores the self-signed client certificate.

We highly recommend that you use an encoded password to ensure successful authentication and increase the level of security; see Creating an Encoded Password.

See OpenID Connect Settings for details.

Creating an Encoded Password

C/C++test can encrypt your password, which adds a layer of security to your interactions with DTP. Run the following command to print an encoded password:

```
-encodepass [your password]
```

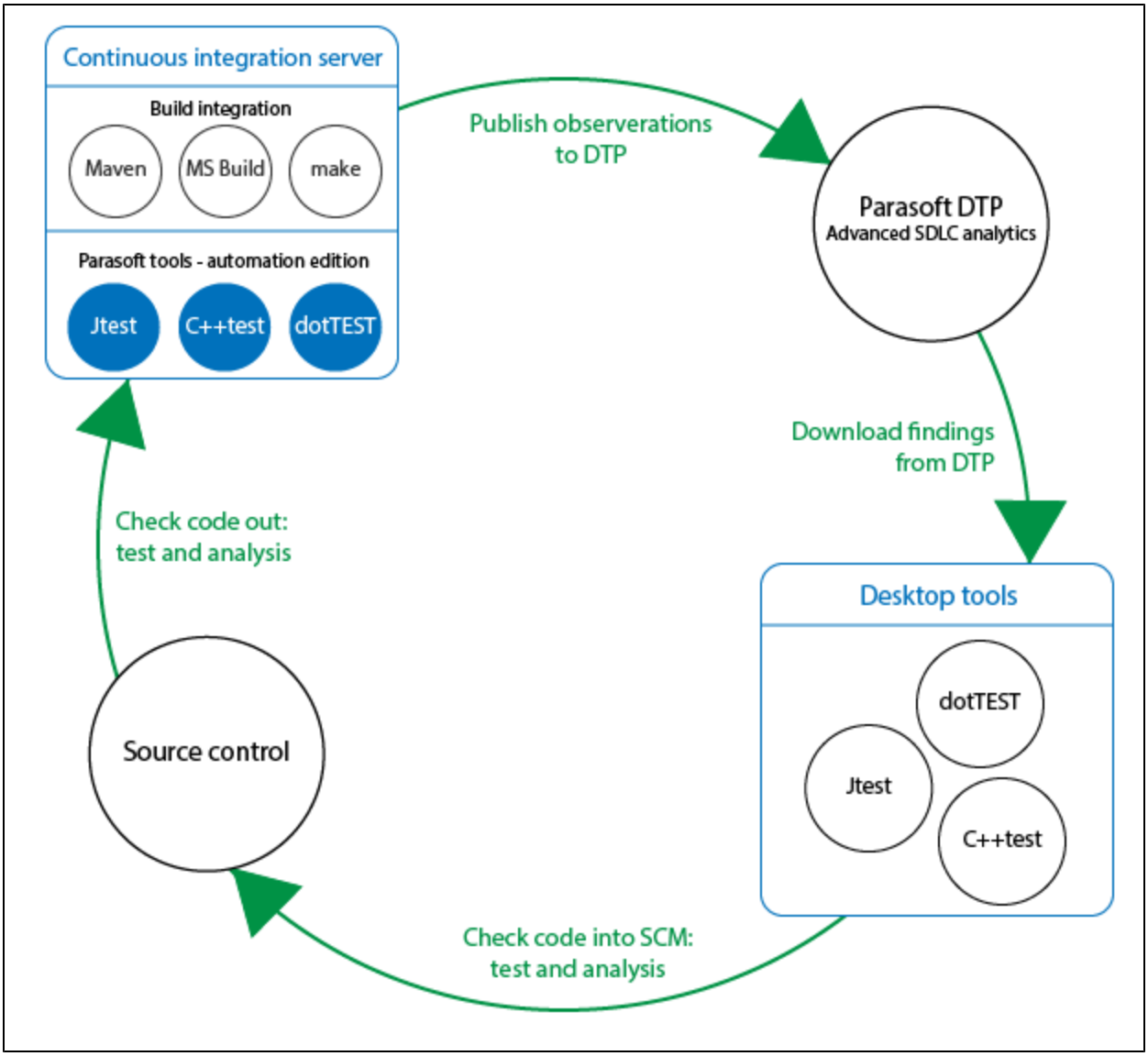
Copy the encoded password that is returned and use it to configure the connection in the `.properties` file. Examples:

- `dtp.password=[your encoded password]`
- `oidc.keystore.password=[your encoded password]`

About the Parasoft Development Testing Workflow

In addition to providing licensing and shared assets for testing and analyzing your software under development, Parasoft DTP collects and merges data points from Parasoft tools, third-party analysis tools, and external systems, such as bug tracking and requirements tracking systems. It aggregates and prioritizes data, as well as performs additional analysis to help you optimize development processes. Using your code analysis and test execution tool with DTP enables you to consistently apply quality practices across teams and throughout the SDLC.

The following illustration shows the general workflow.



Integrating Parasoft Tools with the Build

Parasoft tools ship with plugins for integration with your build tools (i.e., Maven, Ant, Gradle, MS Build, make, etc.). These integrations allow you to analyze code and send data to DTP automatically as part of the automated build processes and continuous integration (CI).

Capturing Observations

When the analysis tool is running, it captures massive amounts of detailed data associated with the code referred to as “observations.” Observations are code quality data, such as static analysis violations, unit test failures, metrics, etc., as well as logistical information about the code, such as authorship, scope, and source control location.

Converting Data into Findings

When observations are sent to DTP, they are converted into “findings” and stored in the database. Findings are observations that have been analyzed, normalized, and aggregated into actionable data.

Importing DTP Findings to the Desktop

You can import priorities and filtered findings from DTP directly into your IDE so that issues can be addressed. control.

Continuing the Cycle

When you check code back into source control, the continuous integration process picks up the change, and the workflow is repeated. This ensures that defects are detected and prevented from becoming software bugs later in the development process when the costs of remediation are much higher.