**Worker Service as Windows Service (WSWS)**

**Overview:**

This documentation provides an overview and detailed description of the Worker Service as Windows Service (WSWS) project, which is a .NET Core Worker Service using the Serilog logging library for efficient and customizable logging. The project demonstrates how to create a background service that logs messages and runs as a Windows service.

**1. Introduction**

The Worker Service as Windows Service (WSWS) project is designed to showcase the implementation of a .NET Core Worker Service using the Serilog logging library. The worker service logs messages at various levels while running as a background task, and it can be installed as a Windows service.

**2. Project Structure**

The project consists of the following main components:

- Program.cs: Entry point of the application. Sets up the Serilog logger and configures the host for the worker service.

- Worker.cs: The worker service class that inherits from `BackgroundService`. Demonstrates how to log messages and perform background tasks.

**3. Getting Started**

To run the project on your local machine, follow these steps:

* Clone the project repository or copy the source code to your preferred directory.
* Open the project in your preferred development environment (e.g., Visual Studio, Visual Studio Code).
* Build the project to compile the source code.

**4. Logging Configuration**

The project uses the Serilog logging library for structured and customizable logging. In the `Program.cs` file, the `Log.Logger` is configured with the following settings:

- Minimum log level: Debug

- Minimum log level for Microsoft-related logs: Information

- Log enrichment from log context

- Log messages to the console

- Log messages to a file named "log.txt" with daily rollover

**5. Worker Service**

The worker service, defined in the `Worker.cs` file, demonstrates how to log messages using the injected `ILogger<Worker>` instance. The `ExecuteAsync` method continuously logs the message "Hello, my name is Shahman." at the Information log level, with a 1-second delay between logs.

**6. Installing as a Windows Service**

To install the Worker Service as Windows Service (WSWS) worker service as a Windows service, use the following command in a command prompt:

```

sc create <YourServiceName> binPath= "<YourAppExecutablePath>"

```

Replace `<YourServiceName>` with the desired name for the service and `<YourAppExecutablePath>` with the full path to the compiled executable.

**7. Troubleshooting**

If you encounter issues with service startup or operation, consider the following troubleshooting steps:

- Check the Windows Event Logs from eventViewer for service-related errors or warnings.

- Ensure that the service executable has the necessary permissions.

- Verify that dependencies are properly configured and accessible.

- Review the logging output for any clues about the issue.

**8. Conclusion**

The Worker Service as Windows Service (WSWS) project provides a practical example of creating a .NET Core Worker Service that utilizes the Serilog logging library for effective logging. It demonstrates how to structure a worker service, configure logging, and run the service as a Windows service. Please customize the documentation to match your project's specifics and provide any additional details that are relevant to your project's use case and requirements.