SOP-DEP-001_Deployment_v1.0.0

SOP-DEP-001_Deployment_v1.0.0

STANDARD OPERATING PROCEDURE

SOP-DEP-001 - OberaConnect Tools Web Interface Deployment Procedure

Field	Value
Document ID	SOP-DEP-001
Version	1.0.0
Status	DRAFT
Effective Date	12/02/2025
Next Review	06/02/2026
Classification	[ ] Public [x] Internal [ ] Confidential
Review Cycle	[ ] Quarterly [x] Semi-Annual [ ] Annual
Process Owner	[TO BE COMPLETED]
Author	[TO BE COMPLETED]
Approved By	[TO BE COMPLETED]
Department/Team	IT Operations

Table of Contents

11. Approval

---

1. Purpose

This document outlines the standard procedure for deploying, configuring, and managing the OberaConnect Tools web interface. The objective is to ensure a consistent and reliable deployment for internal team access.

2. Scope

In Scope: - Manual and automated (systemd) deployment of the web application. - Configuration of the server, including port, workers, and logging. - Firewall setup for network access. - Optional SSL/HTTPS configuration. - Basic troubleshooting and maintenance procedures. - Instructions for end-user access and basic feature usage.

Out of Scope: - Application code development or modification. - Database administration or schema changes. - Operating system installation and hardening. - Advanced network configuration beyond firewall port opening.

Applies To: - System Administrators, IT Operations personnel, and any team members responsible for deploying or maintaining the OberaConnect Tools application.

3. Definitions & Acronyms

Term	Definition
Gunicorn	A Python Web Server Gateway Interface (WSGI) HTTP server.
systemd	A system and service manager for Linux operating systems.
UFW	Uncomplicated Firewall, a frontend for `iptables` on Ubuntu.
firewalld	The default firewall management tool on CentOS and RHEL.
WSL	Windows Subsystem for Linux.
SSL/HTTPS	Secure Sockets Layer / Hypertext Transfer Protocol Secure, for encrypted web traffic.
PID	Process ID, a unique identifier for an active process.
SOP	Standard Operating Procedure.
IP	Internet Protocol, a numerical label assigned to each device connected to a computer network.

4. Roles & Responsibilities

Role	Responsibility	Authority Level
System Administrator	Executes the deployment, configuration, and maintenance procedures outlined in this SOP.	Execute
Process Owner	Owns and approves this SOP, ensuring it remains accurate and relevant.	Approve
Team Member	Accesses and utilizes the deployed OberaConnect Tools web interface as an end-user.	Inform

5. Prerequisites & Requirements

- A server with a supported Linux distribution (e.g., Ubuntu, CentOS, RHEL).
- User account with `sudo` (administrative) privileges.
- The OberaConnect Tools project files must be present in the designated project directory (e.g., `/opt/oberaconnect-tools/` or as specified by IT).
- A configured Python virtual environment within the project directory (`venv/`).
- Network connectivity between the server and end-user machines.

6. Procedure

6.1 Manual Application Startup

This method is used for development or temporary instances.

1. **Navigate to the project directory:** `bash cd /opt/oberaconnect-tools`
2. **Start the server in Development Mode:** For development, use the `--dev` flag. This typically enables debugging and auto-reloading. `bash ./start_server.sh --dev`
3. **Start the server in Production Mode:** For standard operation, run the script without flags. This uses Gunicorn for better performance. `bash ./start_server.sh`

6.2 Systemd Service Configuration (Auto-Start)

This procedure configures the application to start automatically on server boot.

1. **Copy the service definition file** to the systemd directory. `bash sudo cp /opt/oberaconnect-tools/oberaconnect-tools.service /etc/systemd/system/`

2. **Reload the systemd daemon** to recognize the new service. `bash sudo systemctl daemon-reload`
3. **Enable the service** to start on boot. `bash sudo systemctl enable oberaconnect-tools`
4. **Start the service** immediately. `bash sudo systemctl start oberaconnect-tools`
5. **Manage the service** using the following commands:
    - Check status: `sudo systemctl status oberaconnect-tools`
    - Stop service: `sudo systemctl stop oberaconnect-tools`
    - Restart service: `sudo systemctl restart oberaconnect-tools`

6.3 Application Configuration

Modify the `gunicorn_config.py` file to adjust server settings.

1. **Change the listening port:** `bind = "0.0.0.0:5000"` (Change 5000 to the desired port).
2. **Adjust the request timeout:** `timeout = 120` (Change the request timeout in seconds).
3. **Restart the service** to apply changes. `bash sudo systemctl restart oberaconnect-tools`

6.4 Firewall Configuration

Allow traffic on the application's port. The default is 5000.

- **For UFW (Ubuntu):** `bash sudo ufw allow 5000/tcp`
- **For firewalld (CentOS/RHEL):** `bash sudo firewall-cmd --add-port=5000/tcp --permanent sudo firewall-cmd --reload`
- **For Windows Firewall (with WSL):** Create a new inbound rule to allow TCP connections on the specified port for the WSL virtual network adapter.

6.5 SSL/HTTPS Configuration (Optional)

Enable encrypted traffic to the web interface.

1. **Generate a self-signed certificate** (or obtain one from a Certificate Authority like Let's Encrypt). `bash openssl req -x509 -nodes -days 365 -newkey rsa:2048 \ -keyout /opt/oberaconnect-tools/ssl/server.key \ -out /opt/oberaconnect-tools/ssl/server.crt`
2. **Edit `gunicorn_config.py`** to point to the certificate files. `python keyfile = "/opt/oberaconnect-tools/ssl/server.key" certfile = "/opt/oberaconnect-tools/ssl/server.crt"`
3. **Update the bind address** to the standard HTTPS port. `python bind = "0.0.0.0:443"`
4. **Restart the service.** `bash sudo systemctl restart oberaconnect-tools`
5. **Update firewall rules** to allow traffic on port 443.

6.6 Backup Procedure

Backup critical application data.

1. **Identify data files to backup:**
   - `/opt/oberaconnect-tools/contracts_tracking/contracts_data.json`
   - `/opt/oberaconnect-tools/call_flows_processed/`
   - `/opt/oberaconnect-tools/call_flows_generated/`
2. **Execute the backup command** to create a compressed archive. `bash cd /opt/oberaconnect-tools/ tar -czf oberaconnect-tools-backup-$(date +%Y%m%d).tar.gz \ contracts_tracking/ \ call_flows_processed/ \ call_flows_generated/`
3. Move the resulting `.tar.gz` file to a secure backup location.

6.7 End-User Access and Usage

Instructions for team members to use the deployed tools.

1. **Find the server's IP address:** `bash hostname -I`
2. **Access the web interface** by navigating a browser to `http://[SERVER-IP]:5000` (replace `[SERVER-IP]` with the address from the previous step and use the correct port if changed).
3. Use the application features as described in the "Team Access Instructions" section of the source document.

7. Verification & Quality Checks

- Verify the service is running without errors: `sudo systemctl status oberaconnect-tools` shows an "active (running)" state.
- The application is accessible from the server itself at `http://localhost:5000`.
- The application is accessible from another computer on the same network at `http://[SERVER-IP]:5000`.
- A call flow can be successfully generated and downloaded from the "Call Flow Generator" tab.
- Contract renewal data is visible and accurate on the "Contract Tracker" tab.
- Log files are being written to `/opt/oberaconnect-tools/logs/`.

8. Troubleshooting

Issue	Cause	Resolution
Port Already in Use	Another application is listening on the same port.	Find and stop the conflicting process: `lsof -i:5000` followed by `kill -9 <PID>`.

Issue	Cause	Resolution
Permission Denied	The user running the service does not have read/execute permissions for the project files.	Check permissions with `ls -la`. Correct ownership with `sudo chown -R <service_user>:<service_group> /opt/oberaconnect-tools/`.
Service Won't Start	Configuration error, code issue, or environment problem.	Check detailed service logs: `sudo journalctl -u oberaconnect-tools --no-pager -n 50`. Test Gunicorn directly: `cd /path/to/project`, `source venv/bin/activate`, `gunicorn -c gunicorn_config.py call_flow_web:app`.
Can't Access from Network	Firewall is blocking the port, server is not running, or incorrect IP is being used.	1. Verify service is running locally (`curl http://localhost:5000`). 2. Check server firewall rules. 3. Confirm the server IP address with `hostname -I`. 4. Ensure the client and server are on the same network.

9. Related Documents

- `gunicorn_config.py` - Server worker and network configuration.
- `oberaconnect-tools.service` - Systemd service definition file.
- [SOP-BCK-001 - System Backup and Recovery Procedures]

10. Revision History

Version	Date	Author	Changes
1.0.0	12/02/2025	[TO BE COMPLETED]	Initial release based on `DEPLOYMENT_GUIDE.md` version 1.0.

11. Approval

Role	Name	Signature	Date
Author			
Technical Reviewer			
Approver			