Teleprint

Product Overview

Table of Contents

Introduction

Product Description

Device Web Connector allows a connection between a local resource such as a printer or serial port and a remote service. This is done over an outbound connection running the same protocols as a web browser, with the goal that no special firewall permissions are required.

Comparison to Site-to-Site VPNs

A common method to connect local resources to a remote service is building a site-to-site VPN or other form of Wide Area Networking. If the local resource and remote service are part of the same organization, these connections are typically already in place.

When the remote resource resides in another organization, the creation of site-to-site VPNs can still be done but requires co-ordination of two IT teams.

For large organizations the building of a site-to-site VPN will require approvals and change control, which can take a significant amount of time and expense. For small organizations the same task may require an outside contractor.

Aside from setup time, the use of site-to-site VPNs often results in a non-standardized set of configurations to maintain, because each peer organization has their own set of standards and requirements.

By using Device Web Connector, the device connection can be made via an application running on a normal Windows computer without additional IT involvement.

Comparison to Destination NAT / Pin-Holes

Another common method to connect local resources to a remote service is to configure Network Address Translation on the local firewall to allow a direct connection from external services. This is commonly refered to as a "pin-hole".

In most cases this is combined with a restriction based on source IP as a rudamentory form of access control

The same comparison exists as with site-to-site VPNs, with the added caveat that these connections are not encrypted. In many common cases transmission of information without encryption across the open internet violates contractual requirements on protection of Personally Identifiable Information.

Use Cases

Specialized Printers

Specialized printers such as label printers or high-speed band printers are often driven by specialized software that requires a direct connection. Device Web Connector allows the software to function as it was designed even if the printer is at a remote site.

In this use case, the Java-based website or C++-based nexus application can be deployed on-premises as a "liason" service, and the service will publish a TCP port for each printer. Any printing applications can be pointed to that network port as if it were a local network printer.

Direct Printing from Web Applications

Direct printing can be built into your existing web application and handled entirely by your application, using whatever language your application is built in. As long as your language and framework allow the use of websockets, direct printing should work.

If you don't want to implement the logic of talking to our client application, you can deploy our Java-based application and include it in your application via an iframe. The Java application listens on a TCP socket that your application can print to, and authorization is handed off via a JSON Web Token.

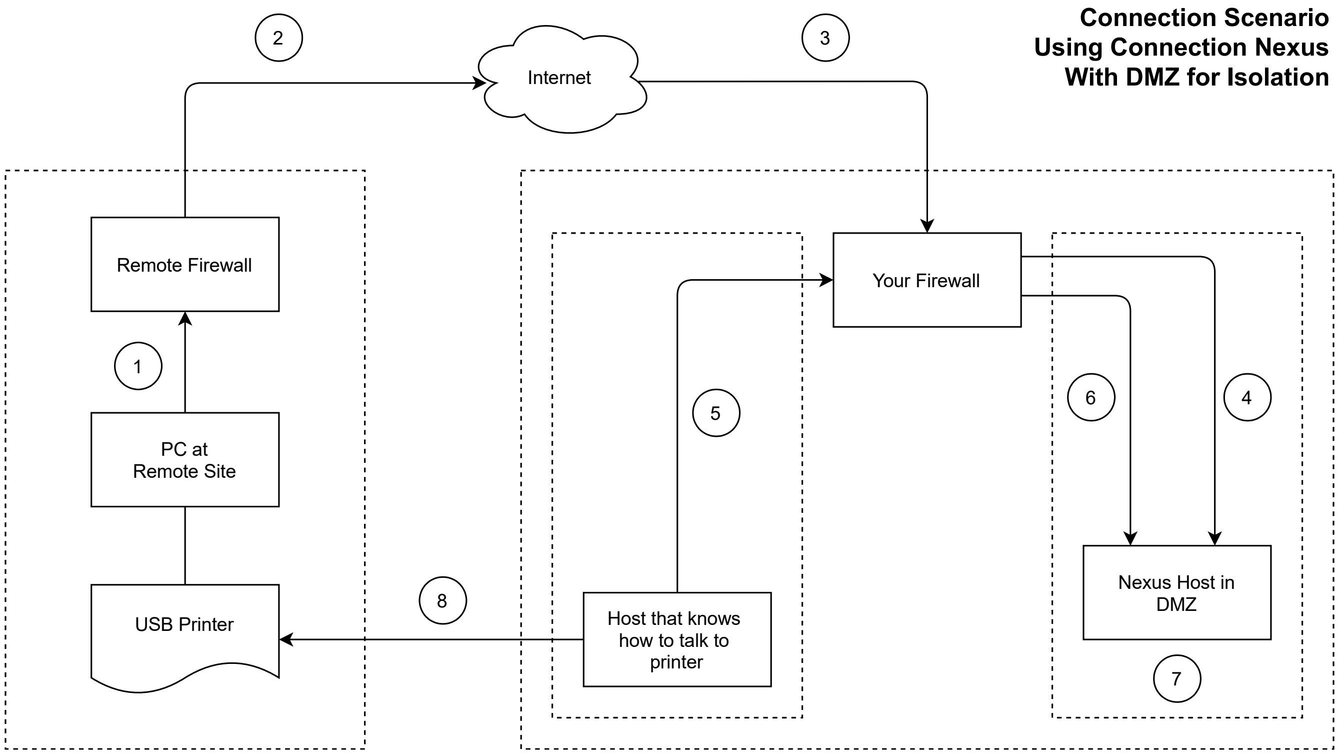
Using Device Web Connector allows your web application team to focus on their main technology platform, and avoid wasting energy building client-side applications to deal with a tiny fraction of their problem space.

Push Printing across Organizational Boundaries

In situations involving controlled documents, certificates, or tags, it is often a requirement that print jobs must be pushed directly from the service end. Using Device Web Connector allows this to be done without the expensive and time-consuming setup associated with site-to-site VPNs.

Data-Plane Diagram

The following diagram illustrates basic data-plane flow for Device Web Connector, using the recommended configuration of the liason service in a dedicated DMZ:



1) The Device to Web Connector software initiates an HTTPS connection through the remote firewall

2) HTTPS connection exits customer site like any other HTTPS traffic

3) HTTPS connection is routed to your firewall.

4) HTTPS connection is forwarded by your firewall to the Connection Nexus software running in your DMZ firewall zone

5) A printing source connects to port 9101 on the IP of the Connection Nexus server

6) Your firewall allows port 9101 to nexus in DMZ

7) Connection Nexus forwards traffic between requester and provider

8) Printing source believes it is directly connected to printer through a print server

Browser-Based Configuration

In a browser-based configuration, the connection process is managed entirely by a website. Typically a websocket is created to keep state in sync between the javascript running in the browser and the connection handling on the web server.

For documentation purposes, integration with your existing web application or website is broken into two cases:

Loose Integration	Your application runs more or less unchanged. Your application can submit jobs to a TCP port. Device session state is handled in an iframe.
Tight Integration	Your application launches the client application directly. Your application handles all management. Your application directly handles the data plane.

Loose Integration

In the "loose integration" scenario, the pre-made "dtserver" application is run as a separate web application which only handles device connections. This is integrated with your application by creating an iframe which passes a JSON Web Token (JWT) to authorize use of services.

While this doesn't offer as much control as using the client directly from your application, this might be appropriate to move to a functional configuration faster while working on a better integration, or it might be entirely sufficient for your needs.

This is also a good fit for "push printing" applications if the client can be authenticated by a CMS or similar system.

For more information on the loose integration scenario, consult the DWC Java App Reference document.

Tight Integration

In a tight integration scenario, your application uses the client application for data-plane traffic and handles the control-plane itself. For more information on direct integration with the DWC client installation, consult the DWC Client Reference.

Non-Browser Based Configuration

In the non-browser based configuration scenario, the application is installed directly on end-user machines, and configured to connect either manually or by distributing a JSON configuration file.

A C++ implementation known as the Communications Nexus is provided which can provide a network service behind a firewall, accept connetions from the client applicaiton, and connect the two together.

For further information on the nexus method, please see the document DWC Nexus Reference.

Security Concerns

Remote Access

By design Device Web Connector does not include any remote access or remote support functionality.

Software Signatures

All Teaglu software is signed by an Organizationally Verified digital signature. Both executables and MSI installation files are signed.

Software Installation Modes

Device Web Connector client package comes in two varieties – one is designed to be installed system wide, and the other within a user account without system privileges.

Typically the per-user version of the Device Web Connector client is pushed by integrated systems, because it is not possible to determine which version is required ahead of time, and prompting the user would result in a more confusing user experience.

Client Updates

The client application communicates its version number to the head-end in the form of A) a member in the dtlaunch probe message, and B) an HTTP header in the application-level API. Head-end applications are encouraged to deny connectivity to older client versions to force an upgrade.

If the per-machine installation is used, administrators may choose to A) install Teaglu Software Update to keep software up to date with a given channel, or B) manage software versions manually.