Interfacing Cordra and Other DOIP Compliant Servers to AI/LLMs

Introduction

This document provides an introduction to CNRI's demonstration project to interface Cordra® software and other DOIP compliant digital object servers to LLMs.

Cordra is highly configurable software for managing digital objects with resolvable identifiers at scale. An instance of Cordra may be accessed via the Digital Object Interface Protocol (DOIP). DOIP provides a way to interact with any type of information represented in digital object form, and structured as a digital object, including the execution of any relevant operations involving that object. Cordra provides a way to manage those objects and access them via DOIP. Especially significant for LLM interactions, both in LLM training and LLM output, is the inherent functionality of Cordra/DOIP to associate persistent identifiers with objects and to easily include provenance and audit logs. Note that both the provenance and audit logs are associated with the object as opposed to the service providing access to the object. This means that essential information persists with the object even if it moves from one environment to another.

LLMs and other AI-based services provide advanced automated reasoning which is typically trained on large amounts of text or other data, can perform analysis based on that data, and can output the results of that analysis in human-readable form. Interaction with these services is through everyday human language, useable by most anyone.

Connecting data sources to LLMs via DOIP would provide some immediate, and perhaps revolutionary, advantages over using either on its own. An LLM that knows how to interact with and use DOIP could, in principle, query any Cordra instance or other DOIP service. It could then discover relevant objects, determine their provenance, perform relevant operations for the task at hand, and post the output of that analysis. That interaction could include execution of a set of operations on one object or across multiple objects, returning the result back to the DOIP service in object form. Using DOIP and one or more LLMs at Internet scale, e.g., traversing a global network of scientific data centers providing data via DOIP, this combination could lead to insights and results difficult or impossible for humans to reach on their own, with or without programming experience.

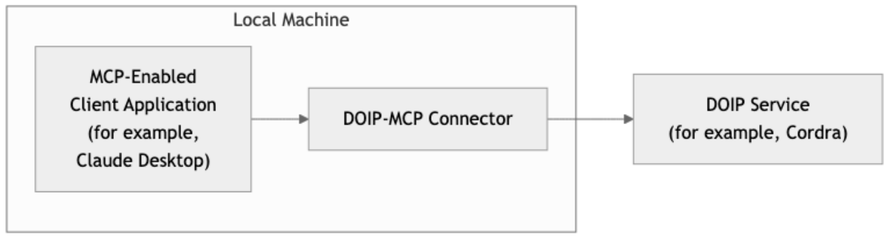
One obvious challenge with the above scenario is that each individual LLM would need to acquire the knowledge to use DOIP. That problem was potentially solved by the recent introduction of the Model Context Protocol (MCP), created by Anthropic, with support promised from many of the other major players, provides a common protocol for interacting with LLMs and other AI services. Creating an interface between MCP and DOIP would enable the 'LLM to any DOIP service' connection described above and would not have to be repeated for every new LLM. While MCP appears to have wide support, our general approach could certainly be applied to any competitive protocols that came into prominence.

CNRI has written demonstration software that provides this interface and plans to make it available. The rest of this document briefly describes its use, including some example use cases.

Technical Details about the Software

We call this software the *DOIP-MCP Connector* (*Connector)*. It acts as a bridge between a separate MCP-enabled client application, and a separate DOIP service. The MCP-enabled client application is assumed to be a piece of software, generally AI or LLM powered, with which a user interacts and which is also able to act as an MCP client; the particular example we use in the demonstration is *Claude Desktop*, but, in principle, the *Connector* would work with other examples. The DOIP service is server software which manages digital objects and listens for DOIP requests to perform operations on those objects; the particular example we use in the demonstration is Cordra, but in principle the *Connector* would work with other examples. The DOIP-MCP Connector provides DOIP service access to the MCP-enabled client application.

The initial release of the *DOIP-MCP Connector* software needs to be installed alongside the MCP-enabled client application which will use the *Connector* to make requests to the DOIP service.


Local Machine
MCP-Enabled Client Application (for example, Claude Desktop)
DOIP-MCP Connector
DOIP Service (for example, Cordra)

Requests made by the MCP-enabled client application to the *Connector,* on the local machine, use MCP; requests made by the *Connector* to the DOIP service, across the Internet, use DOIP.

For demonstration purposes our example MCP-enabled client application is Claude Desktop, a user-facing desktop application that would be installed on a user's machine; the *DOIP-MCP Connector* would also be installed on the same machine. Future releases may allow running the *Connector* as an MCP network service alongside a DOIP service. Here, only DOIP is used across the network, and MCP communication is entirely local to the client.

Each of (1) the MCP-enabled client application, (2) the *DOIP-MCP Connector*, and (3) the DOIP service needs appropriate configuration in order for the client application to successfully perform DOIP operations. In simple cases this configuration is just the editing of small text files and could be completed in minutes.

The MCP-enabled client application must be configured to indicate how to start the *Connector* interface, which is run as a subprocess.

The *DOIP-MCP Connector* itself must be configured to indicate the DOIP service that it will talk to, as well as the authentication that it will use when making DOIP requests. This version of the *Connector* supports both username/password and public/private key authentication. In this initial release, a single instance of the *DOIP-MCP Connector* will talk to a single DOIP service; interacting with multiple DOIP services requires multiple individually configured instances of the *Connector*.

Finally, in order to make custom operations available through the Connector, the DOIP service needs to provide an operation with identifier: *20.DOIP/Op.ListOperationsForTypeWithDetails* to provide details about custom operations that the MCP-enabled client application can access. The *DOIP-MCP Connector* will perform this operation on the service object (that is, the DOIP service considered itself as a digital object) and on each digital object of type Schema that it can find via search at the configured DOIP service. This operation will return a list of all operations that can be performed on objects of particular types, along with details which will be supplied to the MCP-enabled client application such as a natural language description and an input schema for the operation.

Documentation on the necessary configuration is provided with the DOIP-MCP Connector software README file.

Example Use

We discuss here some possible ways to use the *DOIP-MCP Connector* with a natural language input tool like Claude Desktop.

1. Natural language search of Cordra. Without needing to know the search syntax used by a Cordra instance, a user can just ask “What documents are stored in a Cordra that contain the phrase …” and the AI tool will translate this into formal search syntax.

2. Creation of Cordra objects from a description. If the *DOIP-MCP Connector* is configured to allow write operations, a user could say “Create a Cordra object of type XYZ that has the following information.” The AI tool can look up the structure required for objects of type XYZ and determine how to translate the natural language description into a digital object of the appropriate type.

3. Performing custom operations. Our demonstration of the *Connector* talks to a Cordra instance with objects of type Sensor having a custom operation to read the sensor; the digital object corresponds to a real-world sensor which can take measurements on request. By asking the AI tool “Using Cordra, what is the temperature in the server room?” the AI tool will look for an object of type Sensor which has a description indicating it can retrieve the temperature in the server room, and then use the custom operation to retrieve the temperature.