Technology Reports | AI Agent | Natural Language Processing Technology | Voice Recognition/Synthesis

Special Articles on New AI Agent

# Multipurpose Dialogue Engine for Converting Natural Language into Instructions to Enable Services

Innovation Management Department Tetsuo Sumiya Koji Yamazaki
Service Innovation Department Noriyoshi Kamado Go Tanaka
Service Design Department Kosuke Kadono

Spoken words regularly used by people in daily life often entail ambiguous expressions. The Multipurpose dialogue engine has functions to analyze and interpret such diverse and unclear natural language and achieve services. It consists of a natural dialogue platform, a voice recognition function, a speech synthesis function, a service platform front-end, a dashboard for users and a dashboard for developers. This article provides a general overview of the multipurpose dialogue engine and describes its components.

## 1. Introduction

As well as having voice recognition/synthesis and natural language processing capabilities, multipurpose dialogue engine (**Figure 1**) is characterized by its ability to link external services through the Application Program Interface (API)*1. Thus, it enables users to have a wide range of new experiences through dialogue with users on various devices, content provision, and easy mounting of agents that operate devices.

Since launching voice agent services such as the "Shabette Concier" in 2012, the "natural dialogue engine" in 2015, and "Oshaberi Robot for biz" in 2016, NTT DOCOMO has built a log containing more than three billion user utterances from more than eight million people to improve the dialogue performance of its natural language processing



*1 API: An interface that enables software functions to be used by another program.

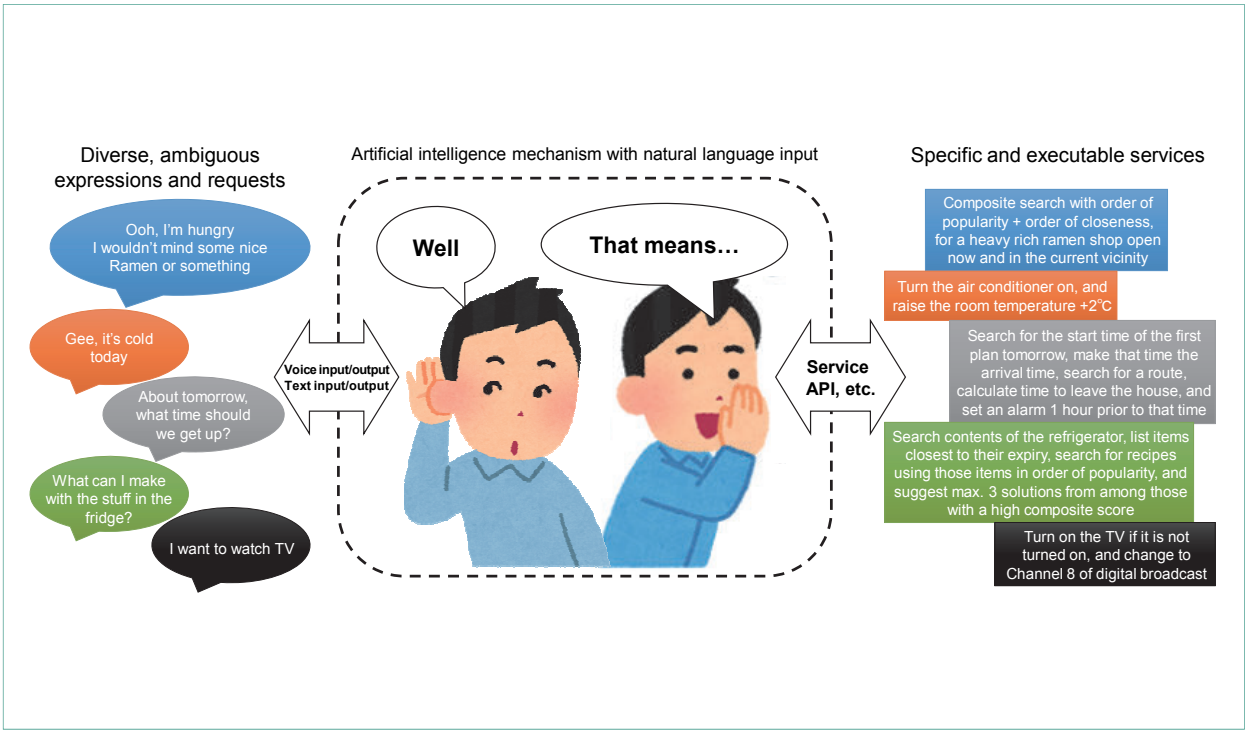


Figure 1 Multipurpose dialogue engine overview

technology. The multipurpose dialogue engine is the result of this six-year accumulation of know-how.

## 2. Multipurpose Dialogue Engine Technology

### 2.1 Multipurpose Dialogue Engine System Architecture

Multipurpose dialogue engine consists of the Service Platform Front-end (SPF), Automatic Speech Recognition (ASR), Natural Language Understanding (NLU) platform, a Text-To-Speech (TTS) speech synthesis[*2], a User DashBoard (UDS), and a Developer Dashboard (DDS). **Figure 2** shows the overall system flow of dialogue.

- Voice recognition from user utterance

  Normally with voice recognition, devices include a Software Development Kit (SDK)[*3]. According to user utterances, the device sends authentication tokens acquired in advance and either voice or text to the engine. At first, the SPF receives all user requests and verifies the authentication token. After the authentication token is verified, voice data is sent to ASR, and the recognition result response is given in text format.

- Linking external services from natural language processing

  NLU processes the recognition result with natural language processing and then recognizes tasks. According to the recognized tasks, NLU links to external services, and retrieves the information required by the user. For example, for a recognition result

*2 Speech synthesis: Technology for artificially creating speech data from text and verbally reading out text.

*3 SDK: A set of documents, tools, libraries, sample programs, etc. needed to create applications.

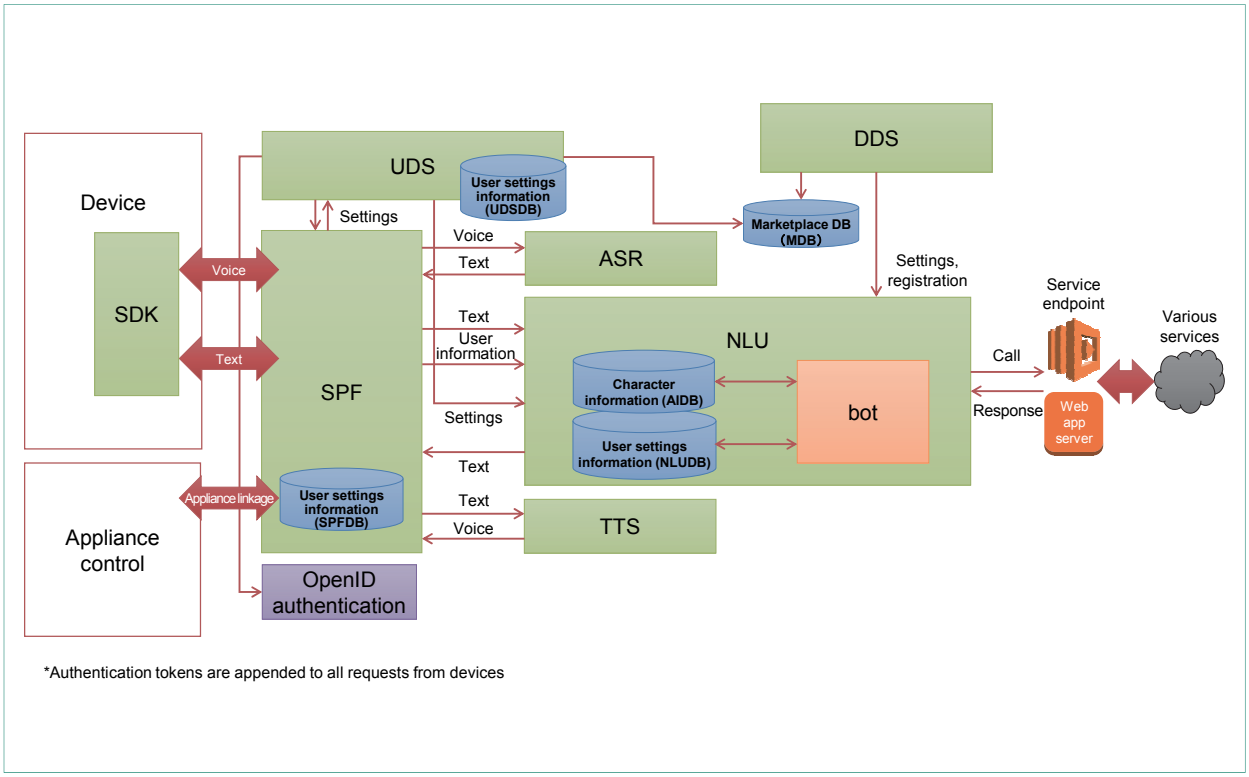


Figure 2 Multipurpose dialogue engine system configuration

for "what's today's weather?", NLU determines that it should link to a weather forecasting service, makes a request using an API for acquiring weather forecast information, and based on the response to the request, creates a text response for the user.

- Responding to users with speech synthesis

The response created for the user is sent to TTS, which converts the text into voice data using a voice model specified in advance. The created voice data is then sent to the device through SPF to provide a voice response to the user's inquiry. The device SDK and SPF maintain a WebSocket connection from the user's initial utterance through to the end of the conversation, which enables high-speed responses regardless of the fact that voice response data is larger than text and takes longer to load.

## 2.2 SPF

As mentioned, for continuous dialogue requests including voice and text from SDKs, the SPF has functions to generate appropriate responses in real time for each request by combining responses from the backend engine consisting of ASR, NLU and TTS connected at the latter stage to respond to SDKs. This software architecture is shown in **Figure 3**.

The SPF consists of two types of processing units called "Block" and "Edge". Block performs the processing required to generate a response to

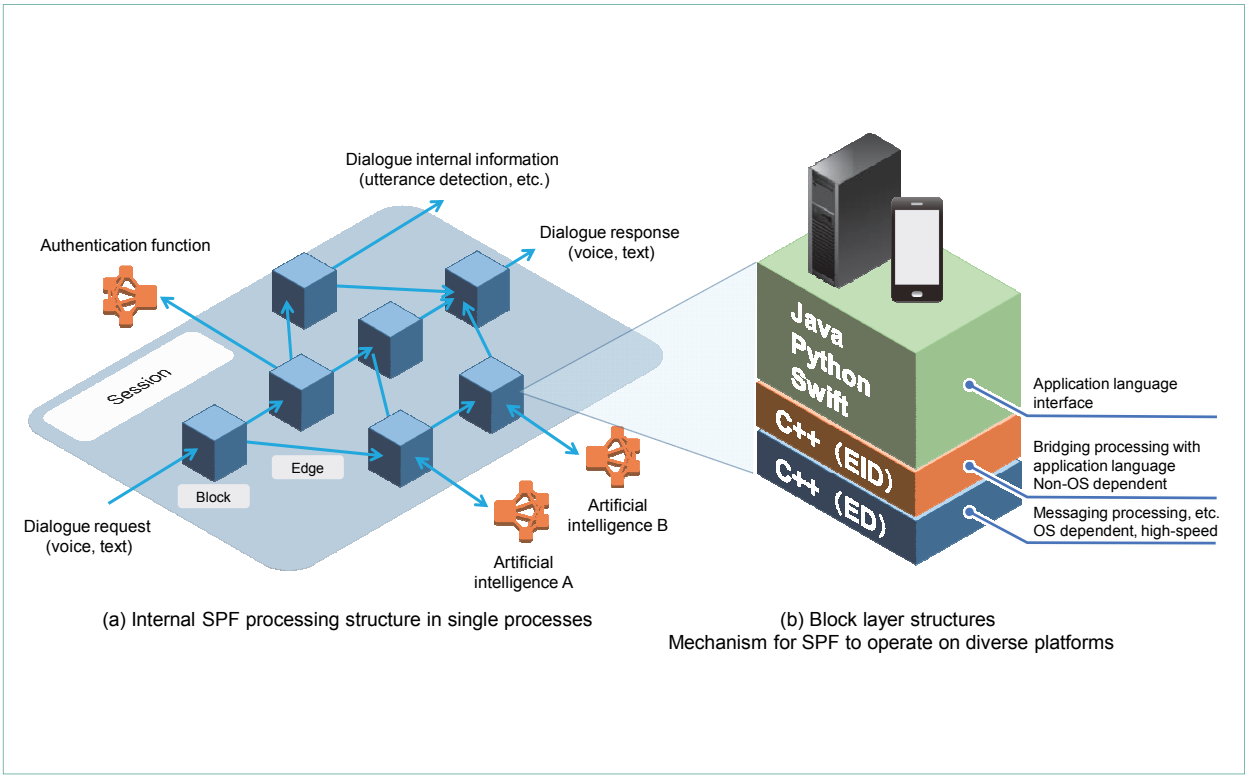


Figure 3 SPF software architecture

a request sent from a SDK, while Edge passes data between Blocks. By combining the results of multiple Blocks with Edge, the appropriate response can be returned to the SDK. This architecture increases the flexibility to change the processing within the SPF and allows an ever-evolving spoken dialogue technology to be quickly incorporated into the SPF.

Also, by separating processes as Block and Edge, it's possible to aggregate data shared between the Blocks on the Edge and raises the level of parallel processing. This enables processing of not only text but also enables real-time streaming*4, which requires parallel processing for large amounts of data such as voice.

The software layer structure of the Blocks is described in Fig. 3 (b). In SPF, the speed of Environment Dependent (ED) processing that requires high-speed and OS optimization is increased with C++ (and other languages which are dependent on the environment).

Also in SPF, through an environment and programming language non-dependence layer (the Environment-InDependent (EID) layer) wrapping the layer described above, it's possible to describe block processing in diverse programming languages. In this way, using SPF as the medium between programs written in different languages enables short service development time.

Using device SDKs with this architecture also achieves high speed and stable voice dialogue processing not only in server environments but also

*4 Streaming: A communication method for sending and receiving audio and video data over the network, whereby data is received and played back simultaneously.

in environments such as smartphones and embedded Linux devices*5.

## 2.3 NLU

NLU is part of the dialogue system that achieves the agents using a description language called xAIML. xAIML is a description language for creating an artificial intelligent dialogue agent on the NLU, and is based on AIML1.1 but with functionality expanded by utilizing NTT DOCOMO's natural language processing technologies*6.

With xAIML, when creating a single dialogue agent, it's possible to develop efficient systems by designing the respective functions of the dialogue agent as bots which can be linked to achieve various functions in the main agents*7 and expert agents*8 in the multipurpose dialogue engine.

1) Design Patterns for Creating Main Agents

NLU is a platform that can be freely customized and incorporated into services and products by developers to develop dialogue services and products able to converse with people, and thus enables quick and easy development of ideal dialogue agents by free combination of the components useful to dialogue system development. The multipurpose dialogue engine is able to create and link its two types of main and expert dialogue agents using this NLU.

A main agent is a main character in the dialogue system, and is the agent that controls dialogue with the user. When the user speaks to an agent (hereinafter referred to as "user utterance"), the main agent to connect is determined on identification of the user. Then, the user's intentions are interpreted*9 with the main agent from the content of utterances, and if tasks are executable in the main agent, they are executed. On the other hand, if the user makes an utterance such as "d Gourmet please" to call a particular expert agent, the user utterance is passed to the expert agent, and the subsequent processing and dialogue with the user are handled by that expert agent. In this way, the roles of main and expert agents are clearly defined, yet can be freely customized.

It's also possible to give priority to group scenarios such dialogue for tasks, calling expert agents, making commands or chatting so that scenarios can be prioritized for individual services. Partner companies can design and develop main agents their preferences. They cannot only freely combine the above scenarios but also create original main agents.

Expert agents are specialized for particular fields, and are called from the main agent. Anyone can easily design these using DDS, and release them onto the designated AI agent API marketplace once the developed expert agents have been screened. Then, users can freely add or delete released expert agents that they want to try, which enables users to individually manage their own expert agents with the UDS.

2) Achieving Expert Agents through a Connected bot Structure

Expert agents in the multipurpose dialogue engine are generated as independent bots using DDS. Users can freely select and enable expert agents through UDS to use the following expert agent-related functions.

- Input a specific word into the main agent to call an expert agent so that it can take over.
- User input is transferred to the called expert agent until dialogue with that expert

*5 Embedded Linux device: A kind of special-purpose device that is incorporated into mobile data terminals, digital appliances and other such products, and that has a CPU and software that runs on the Linux OS.

*6 Natural language processing technology: Technology to process the language ordinarily used by humans (natural language) on a computer.

*7 Main agent: The agent at the forefront of dialogue with users. Service providers can create main agents with the characters they like, and use them to connect services with devices.

agent finishes.

- Outputs from expert agents that are against public order or morals can be filtered, and the attempted output sent to the administrative module of DDS.
- System administrators or expert agent developers can suspend the functions of an expert agent.

In the multipurpose dialogue engine, the above capabilities are achieved by designing functions to manage expert agents of individual users, pass user input to a bot in a dialogue with the user, filter inappropriate language and manage expert agent status as independent bots (**Figure 4**).

Also, the main agents created using the multipurpose dialogue engine can easily implement the functions to offer expert agents by using these bots.

## 2.4 UDS

UDS enables settings for user device authentication and expert agents. Users can use both GUI*10 and REST API*11 to authenticate devices.

**Figure 5** shows an image of UDS in the trial environment. UDS offers the following functions.

- Authentication: The dashboard supports both d Account authentication with OpenID Connect (OIDC)*12 and Google*13 authentication.
- Device list/new registration: Users can register device IDs mapped to devices and map

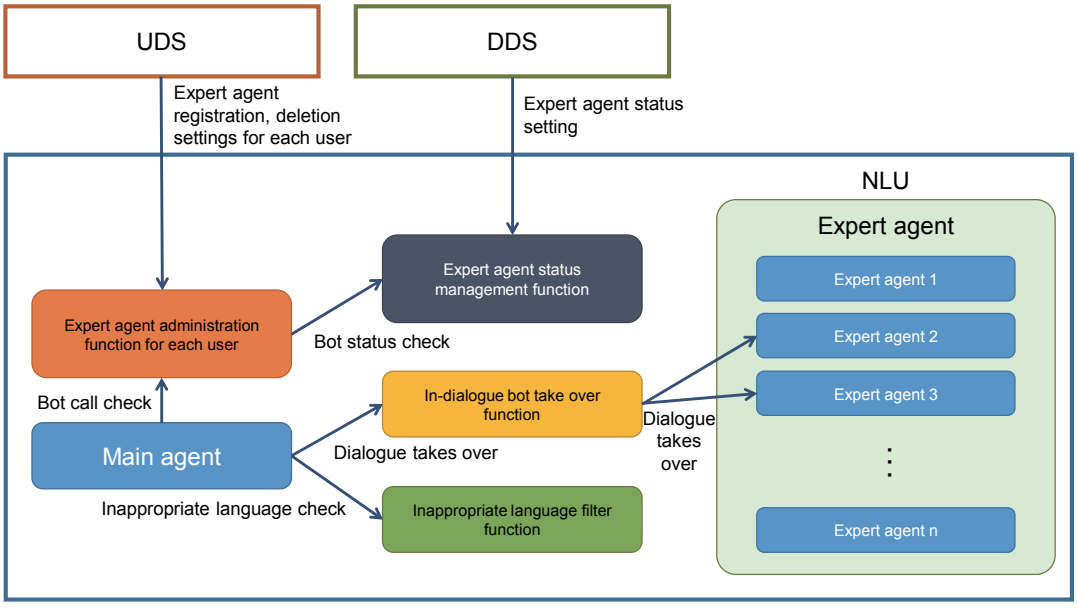


Figure 4 Bot structure to achieve expert agents

*8 Expert agent: An agent specialized for a service and called by a main agent. Service providers can freely provide existing main agents (robots, bots, etc.) with services.

*9 Intention interpretation: Technology that uses machine learning and so forth to determine the user's intention from the user's utterances (natural language). User intentions are called "tasks." For example, all the utterances "What's tomorrow's weather?," "I wonder if tomorrow will be fine?," and "Is it going to rain tomorrow?" are judged as weather tasks.

*10 GUI: A superior type of interface that offers visibility and intuitive operability by expressing operations and objects visually on a screen.

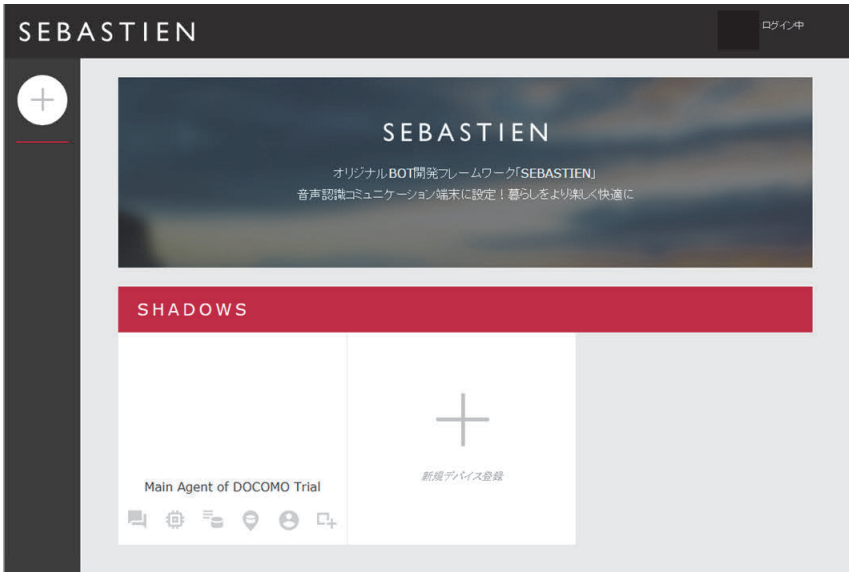


Figure 5　User dashboard image

them to users. Users can display a list of currently registered devices and delete devices if required.

- Dialogue history: Users can display a history of dialogue with main and expert agents.
- Home device linkage: By linking to IoT access control engine, users can make settings to perform administrative tasks such as registering or tagging linked appliances from UDS or operate appliances from expert agents.
- Expert agent configuration: This enables registration and listing of expert agents with DDS. It's necessary to register expert agents with DDS in advance to use them. If an expert agent needs to link an account, OAuth 2.0 authorization is required when registering the expert agent.

## 2.5 DDS

DDS offers expert agent creation functions [2].

1) Design Concept Behind the Expert Agent Creation Function Provided with DDS

The final objective of an expert agent is to provide a specific service to the user. Thus, dialogue design is for that purpose.

Also, dialogue design does not require any specialist knowledge about natural language processing or dialogue design - tools have been designed to enable the process to be completed just by entering setting values on a screen, which enables developers who have never created a dialogue service to create agents simply and quickly.

2) Expert Agent Dialogue Design

Expert agents created with DDS adopt task-oriented dialogue design with slot filling*14 suitable for service provision. The specific dialogue sequence

*11 REST API: An API conforming to REST. REST is a style of software architecture developed based on design principles proposed by Roy Fielding in 2000.
*12 OIDC: A mechanism to link IDs so that authentication can be done with one ID for using various Web sites on the Internet and mobile applications, etc.
*13 Google: A trademark or registered trademark of Google LLC.
*14 Slot filling: A function to extract required parameters from text when executing tasks. For example, to execute a weather search task, the "time" and "place" parameters of "today" and "Tokyo" are extracted from the text "What's the weather in Tokyo today?".

is as follows.

(1) The expert agent extracts Intent*15 and parameters (slots) from user utterances.
(2) If the user makes an ambiguous utterance, the expert agent asks the user again to confirm the user's intent before executing a task.
(3) Finally, the specific intent information (intent, slot) is sent to a set external program (API) with a POST request*16.
(4) Based on the received intent information, the external program executes a service.
(5) The external program returns a response (dialogue wording) for the user according to the results of executing the service.
(6) The expert agent gives the response from the program to the user.

## 3. Conclusion

This article has described the multipurpose dialogue engine which interprets natural language including diverse and ambiguous expressions uttered by users, and converts them to specific executable services. With the concept of main and expert agents, this system offers users with new experiences of dialogue with various devices, content provision and device operation. Going forward, we would like to engage in research and development such as common dialogue technologies required across multiple platforms and advanced intent interpretation technologies in various fields, and take initiatives to develop agent generation technologies to enable high-level dialogue and advance APIs.

### REFERENCES

[1] SEBASTIEN Web site (In Japanese).
https://users.sebastien.ai/
[2] Expert Agent Developer Dashboard Web site (In Japanese).
https://developers.sebastien.ai/

*15 Intent: Refers to tasks defined in expert agents.
*16 POST request: A type of request sent to a Web server from a client (such as a Web browser) with HTTP communications, for sending data from a client to a program etc. specified with a URL. While requests such as GED or HEAD are only headers, POST requests have body sections in which the data desired for sending is described. This is used for sending large amounts of data or files to servers.