SODAS+ DIS

릴리스 ***3.0.0***

KAIST NCL

2022년 12월 23일

Introduction

본 문서는 SODA+ 시스템의 데이터허브 상호 운용 시스템 (DIS) 의 시스템 구조 및 라이브러리 정보를 포함한다. SODAS+ 는 오픈 데이터 생태계 구축 플랫폼으로, 생태계에 참여하여 데이터 공유를 지원하는 데이터 허브 (`DataHub`) 와 생태계의 관리 감독을 담당하는 거버넌스 시스템 (`Governance System`) 으로 구성된다. 그 중에서도 본 문서는 데이터 허브 가 SODAS+ 플랫폼에 하나의 데이터 공유 주체로 참여할 때 다른 데이터 허브 및 거버넌스 시스템 과의 상호 작용을 지원하는 **데이터 허브 상호운용 시스템 (DataHub Inter-operability System, DIS)** 의 구조 및 기능을 소개한다.

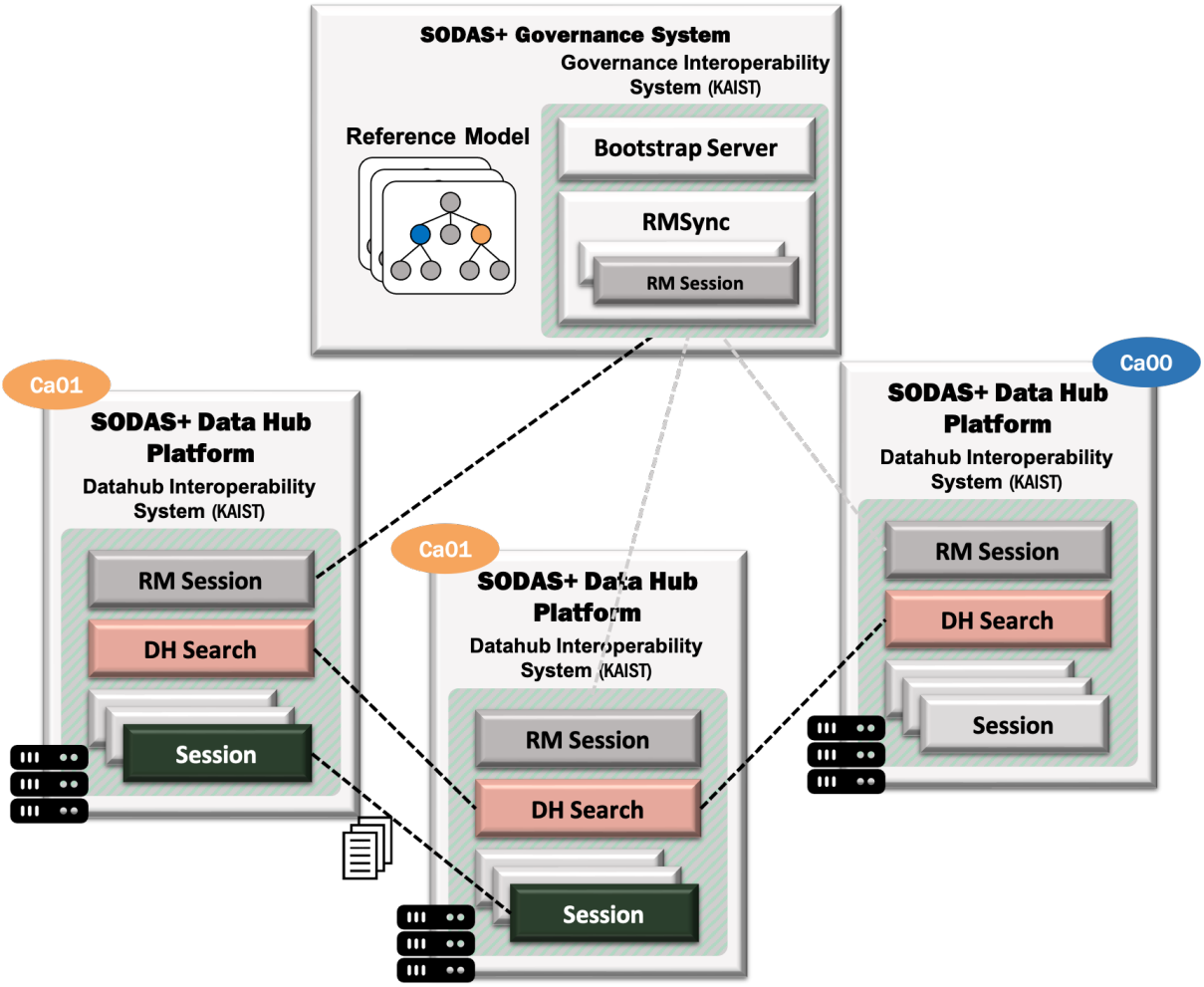


그림. SODAS+ 플랫폼 시스템 구성 예시

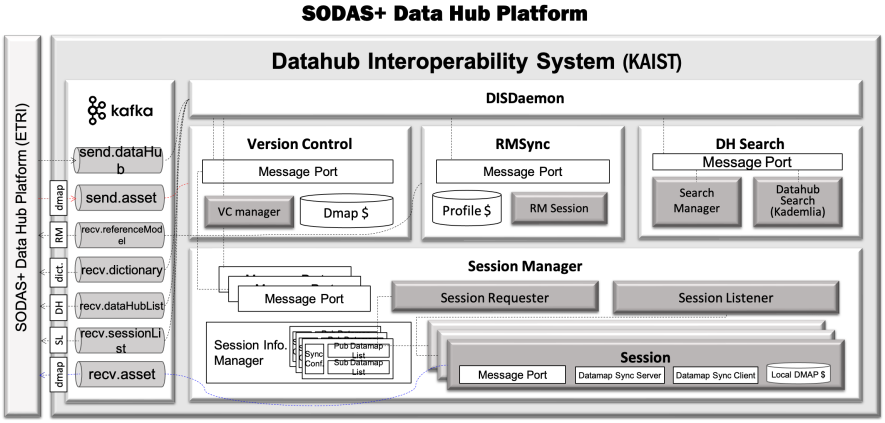


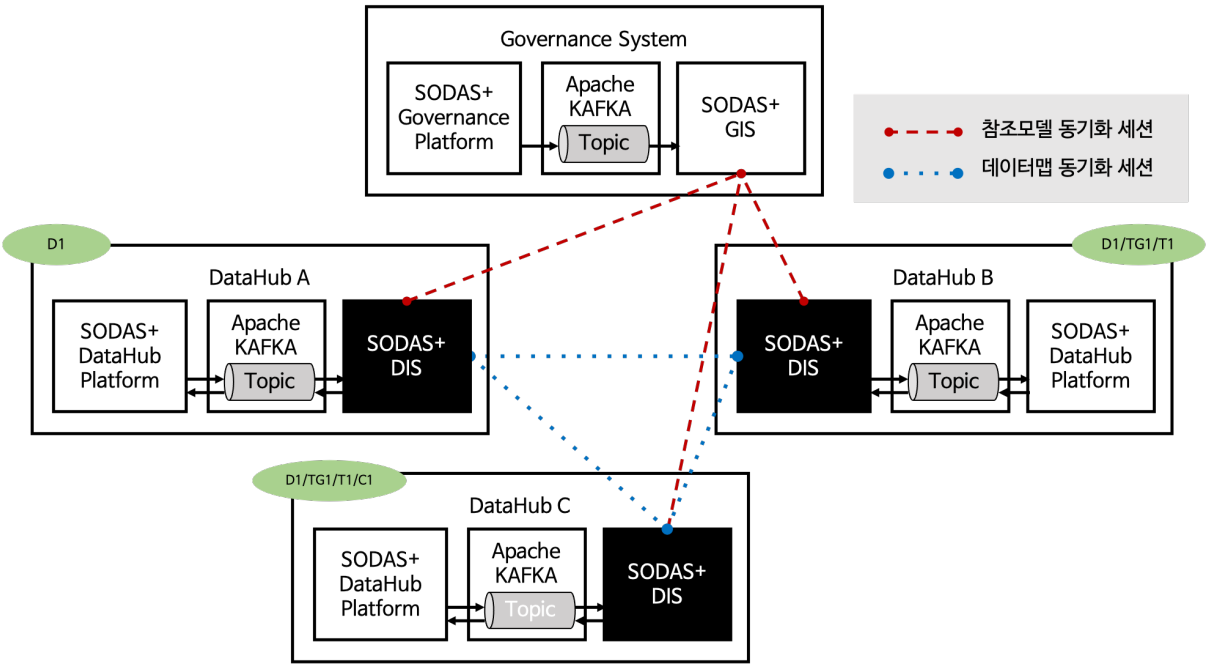
그림. 데이터 허브 상호 운용 시스템 (DIS) 정적 구조

CHAPTER 1

SODAS+ 오픈 데이터 생태계 개요

SODAS+ 는 오픈 데이터 생태계 구축 플랫폼으로, 생태계에 참여하여 데이터 공유를 지원하는 데이터 허브 *(DataHub)* 와 생태계의 관리 감독을 담당하는 거버넌스 시스템 *(Governance System)* 으로 구성된다. 데이터 허브 는 데이터 셋을 관리, 유통하는 주체로서 SODAS+ 에 참여하여 오픈 데이터 생태계를 형성한다. 이 때, SODAS+의 데이터 생태계는 거버넌스 시스템 에 의해 정의된 참조 모델 을 따르며, 생태계에 참여하는 데이터 허브들은 거버넌스 시스템 에서 지원되는 문법을 따라야 한다.

본 문서에서 다루는 데이터 허브 상호운용 시스템 (DIS) 은 본 생태계에서 데이터 허브와 데이터 허브 간, 그리고 데이터허브와 거버넌스 시스템 간 상호 운용을 지원하는 시스템으로 최종적으로는 생태계 내에 있는 데이터 허브 중 데이터허브 관리자가 지정한 관심 토픽 (Interest Topic) 에 대응되는 데이터 맵 정보를 실시간으로 동기화하여 데이터 허브를 사용하는 데이터 사용자들에게 생태계 내의 데이터 정보에 대한 빠르고 정확한 조회를 지원하는 것을 목표로 한다.



1.1 SODAS+ 플랫폼과 데이터 허브

SODAS+ 플랫폼은 독립적인 다수의 데이터 허브들이 보유한 데이터 셋 정보 동기화를 통한 분산 데이터 허브를 구축할 수 있도록 지원한다. 이 때, 개별 데이터 허브들은 보유하고 있는 데이터 셋 (data set)의 종류 (category)가 다를 수 있으며 그 종류는 SODAS+의 거버넌스 시스템에서 관리되는 **표준 참조 모델** 내에 정의된다.

표준 참조 모델 (Reference Model 또는, RM)은 SODAS+에서 공유되는 데이터 셋의 종류를 체계적으로 나누기 위한 모델로, 각 카테고리의 데이터 셋을 표현하는 문법을 관장하기 위한 프로파일 문법과 연동된다. 표준 참조 모델은 거버넌스 시스템 에 의해 관리되며 SODAS+에 참여하는 모든 데이터허브는 최초의 참여 시 거버넌스 시스템 에 질의를 통해 표준 참조 모델을 획득한 후 데이터 허브가 보유하고 있는 데이터 셋을 기반으로 SODAS+ 플랫폼 내 동기화 레벨을 설정한다. 다시 말해, SODAS+ 시스템에 참여하는 데이터 허브들은 참조 모델 내의 카테고리에 속하는 하나 이상의 데이터 셋을 보유하고 있으며, 특정 카테고리에 대응되는 데이터 셋을 보유하고 있는 데이터 허브에 대한 탐색이 요구된다.

따라서 SODAS+ 플랫폼에서는 분산된 데이터 허브를 탐색하고 동일 혹은 근접한 카테고리에 대응되는 데이터 셋을 가지고 있는 데이터 허브들 간의 세션 연결을 통해 보유 데이터 셋에 대한 정보 동기화를 지원해야 한다. 이를 위하여 **데이터 허브 상호 운용 시스템 (DIS)** 에서는 1) 데이터 허브들간의 분산 탐색과 2) 데이터 허브들 간의 데이터 맵 동기화 기능을 제공한다.

1.2 용어 정의

1.2.1 데이터 허브 (DataHub)

사용자에게 데이터를 제공하는 개체로 SODAS+ 플랫폼에 참여하여 보유한 발행 데이터를 공유하거나 SODAS+ 내에 존재하는 데이터를 구독하여 사용자들에게 데이터 정보를 제공한다.

1.2.2 거버넌스 시스템 (Governance System)

SODAS+ 플랫폼에서 공유되는 데이터의 문법 및 분류체계를 관리 감독하는 시스템

1.2.3 데이터맵 관리 시스템, 또는 데이터 허브 플랫폼 (DataHub Platform)

주어진 문법 및 오픈 참조 모델을 기반으로 데이터 맵을 생성하고 데이터 허브 사용자들의 요청을 받아들이는 플랫폼

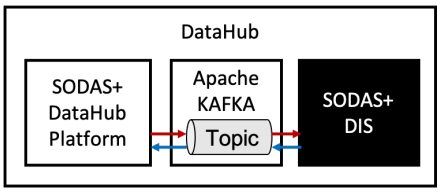
1.2.4 데이터허브 상호운용 시스템 (DataHub Interoperability System)

데이터맵 관리 시스템으로부터 전달된 요청을 처리하기위한 타 데이터허브 및 거버넌스 시스템과의 상호운용을 수행하는 개체로 본 문서에서 기술하는 핵심 시스템.

CHAPTER 2

SODAS+ 데이터 허브 상호운용 시스템 (DIS)

데이터 허브 상호 운용 시스템 (이하, DIS)은 SODAS+의 데이터맵 관리 시스템 (혹은 SODAS+ 데이터허브 플랫폼)과 연동되어 동작하며, 데이터맵 관리 시스템을 통해 획득되는 관리자의 요청사항을 관리하고 결과를 반환한다.



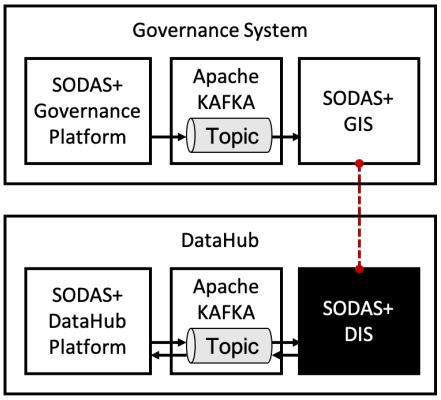
2.1 주요 기능

DIS는 다음의 세 가지 주요 기능을 제공한다:

1. 거버넌스 시스템에서 발행하는 표준 참조 모델 동기화
2. *SODAS+* 시스템 내에 있는 데이터 허브 탐색 지원
3. 탐색 된 데이터 허브 중 관심 정보가 같은 데이터 허브와의 실시간 동기화 지원

2.1.1 거버넌스 시스템에서 발행하는 표준 참조 모델 동기화

DIS는 SODAS+ 에 참여하는 데이터 허브가 표준 참조모델을 따를 수 있도록 거버넌스 시스템으로부터 최신 버전의 표준 참조 모델을 동기화하는 역할을 수행한다. 거버넌스 시스템은 SODAS+에서 유통되는 데이터를 관리하는 문법 및 분류체계를 관리하며 데이터 허브는 해당 규약을 기반으로하여 SODAS+ 생태계 내의 다른 데이터 허브들과 상호 작용을 수행할 수 있다.



거버넌스 시스템과의 오픈 참조 모델 동기화는 두 가지 상황에서 발생한다:

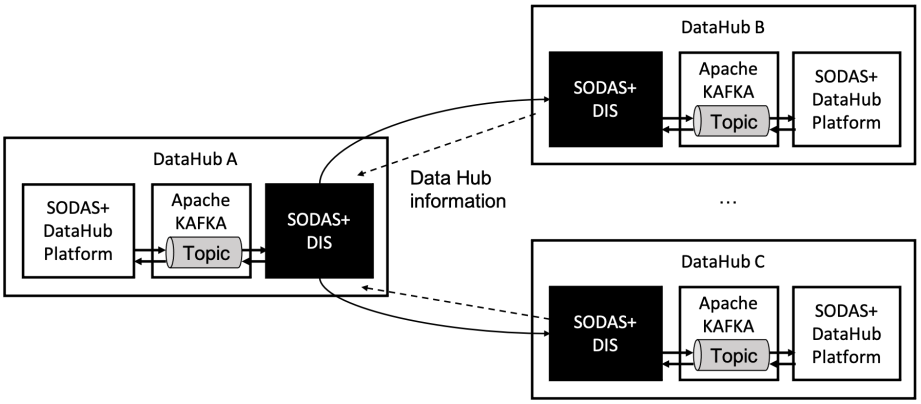
1. 데이터 허브가 처음 SODAS+ 플랫폼에 참여한 경우
2. 표준 허브 내에서 참조 모델의 업데이트가 발생한 경우.

첫 번째 경우 DIS로 *START* 이벤트가 들어오면 *RMSync* 는 거버넌스 시스템에 해당 데이터 허브의 SODAS+ 플랫폼 참여를 알린다. GIS는 해당 데이터 허브와의 참조 모델 동기화 세션을 맺으며 현재 거버넌스 시스템 내에 있는 모든 참조 모델을 데이터 허브로 전송한다. DIS의 참조 모델 동기화 세션은 거버넌스 시스템으로부터 전달받은 참조 모델을 데이터맵 관리 시스템 으로 전송한다.

두 번째 경우는 DIS가 GIS와 참조 모델 동기화 세션을 맺고있을 때 발생하며 GIS 내에 탐조 모델 업데이트 혹은 새로운 참조 모델 생성이 감지되는 경우 이를 DIS로 전송한다. 이 때, DIS로 전달되는 모든 참조 모델은 DIS 내의 버전 관리 모듈 (*VersionControl*)로 전달됨과 동시에 데이터맵 관리 시스템 으로 전달된다.

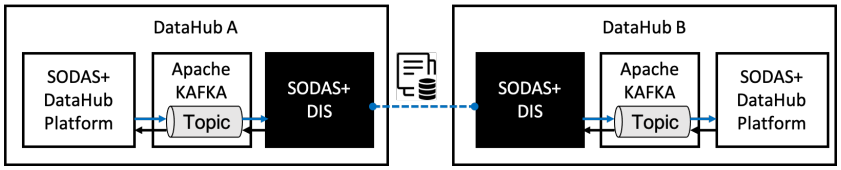
2.1.2 SODAS+ 시스템 내에 있는 데이터 허브 탐색 지원

DIS 는 SODAS+ 생태계에 존재하는 분산 데이터 허브의 탐색을 지원한다. SODAS+의 데이터 허브 네트워크는 중앙 서버 방식이아닌 P2P 방식으로 탐색되며, 분산 데이터허브를 탐색할 때 현재 설정된 데이터 허브의 관심 정보 (Interest)를 바탕으로 관심 정보가 동일하거나 유사한 (혹은 가까운) 데이터허브를 우선적으로 탐색하도록한다. 본 탐색 결과는 데이터허브 플랫폼 으로 반환되며, 탐색은 SODAS+네트워크에 데이터 허브가 들어올때마다 업데이트 된다. 탐색 결과를 바탕으로하여 DataHub 관리자는 자신의 DataHub와 실시간으로 데이터맵 동기화를 맺을 데이터 허브를 지정할 수 있다.



2.1.3 탐색 된 데이터 허브 중 관심 정보가 같은 데이터 허브와의 실시간 동기화 지원

DIS 는 관심 키워드가 일치하는 데이터 허브들과의 데이터맵 동기화를 지원한다. 데이터 허브 간에 동기화 세션이 채결되면 세션이 연결된 데이터 허브끼리는 한 쪽의 데이터 맵이 변경되는 경우 해당 데이터 맵 정보를 연결된 데이터 허브로 전송한다. 이 때, 전체 데이터 맵을 동기화하지 않고, 최근에 동기화 된 상태 정보를 바탕으로 증분 데이터만을 전송하여 동기화를 수행한다. 이 때 동기화는 채결된 세션의 관심정보 특성에 기반하여 진행되기 때문에 모든 데이터맵이 동기화되는 것이 아니라 관심 키워드에 대응되는 데이터맵 만을 선택적으로 동기화하도록 지원한다.



2.2 데이터 허브 상호운용 시스템 (DIS) 정적 구조

데이터 허브 상호 운용 시스템의 정적 구조는 아래 그림과 같다. 데이터 허브 상호 운용 시스템은 데몬 (*Daemon*) , 데이터 허브 분산 탐색 모듈 (*DHSearch*), 참조 모델 동기화 모듈 (*RMSync*), 데이터맵 동기화 모듈 (*SessionManager*), 데이터맵 버전 관리 모듈 (*VersionControl*)로 구성된다.

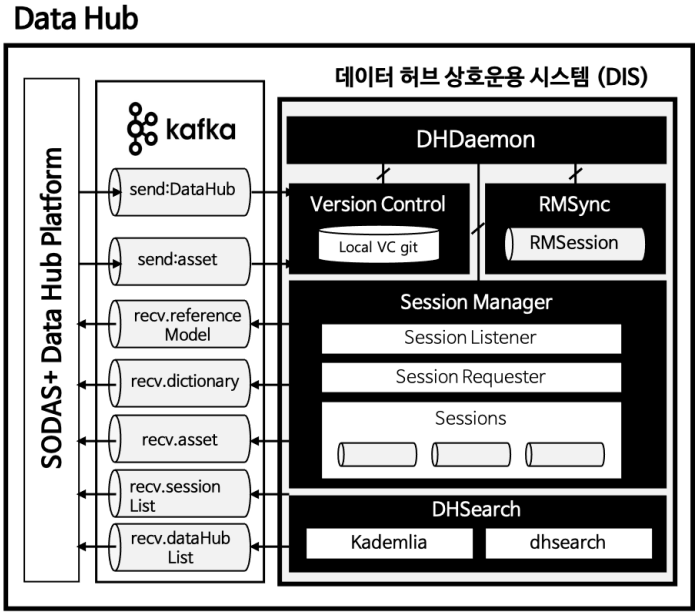


표 1: 데이터 허브 상호 운용 시스템 (DIS) 모듈 개요

Module	Sub-Modules	Description
Daemon	*DHDaemon* *ctrlKafka*	데이터 허브 상호운용 시스템이 실행될 때 자동으로 백그라운드에서 실행되어 DHSearch, RMSync, SessionManager 스레드를 실행하는 최상위 스레드에 해당하는 데몬으로, SODAS+ 플랫폼에서 내부적으로 연동되는 ETRI 시스템과의 KAFKA를 통한 데이터 허브 Configuration, 동기화 옵션 등을 주고받아, 하위 스레드로 전달하는 기능을 수행
DHSearch	*DHSearch* *Kademlia*	데이터 허브 상호운용 시스템 데몬에 의해 실행되는 스레드로, 실행 초기에는 참조 허브(RH)의 Bootstrap 서버로부터 SODAS+ 플랫폼에 최근접속한 Seed Node의 접속 정보와 시스템 데몬으로부터 사용자 관심 토픽 레벨 정보를 받아, 관심 토픽의 데이터맵을 소유한 다른 데이터 허브를 분산 탐색하는 기능을 수행
RMSync	*RMSync*	데이터 허브 상호운용 시스템 데몬에 의해 실행되는 스레드로, 실행 초기에는 참조 허브(RH)와 세션 연동을 통한 참조 모델(Reference-Model)를 수신 받아 관리(표준 허브와 동기화)하며, 추후 참조 허브(RH)에서 참조 모델이 업데이트될 경우, 동기화하는 기능을 수행
SessionManager	*SessionManager* *Session*	데이터 허브 상호운용 시스템 데몬에 의해 실행되는 스레드로, 시스템 데몬으로부터 동기화 옵션을 전달받아 관리하며, DHSearch 모듈에서 탐색한 관심 토픽의 데이터맵을 소유한 다른 데이터 허브와의 데이터맵 동기화를 위한 세션 연동 기능을 수행하며, VersionControl 모듈과 연동하여 다른 데이터 허브와의 세션을 통한 데이터맵 전송 기능을 수행
VersionControl	*VCModule* *VCConsumer* *VersionController*	데이터 허브 시스템에서 발생한 데이터맵 변화 이벤트에 따른 증분 데이터 기반 데이터맵 버전 관리 기능을 수행

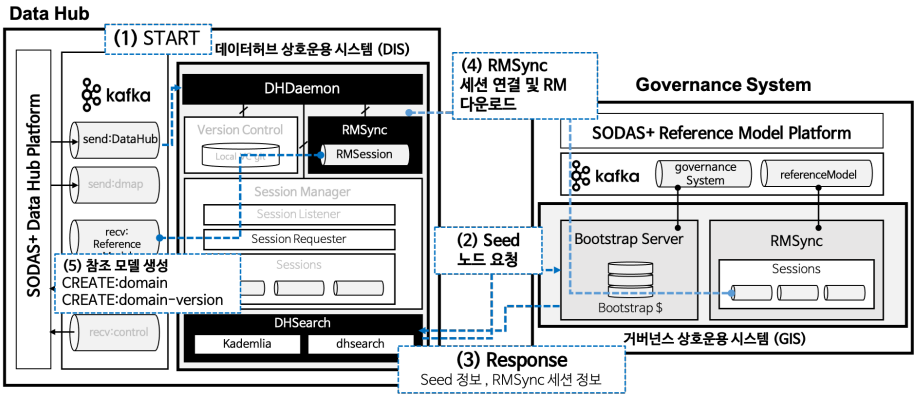
2.3 동작 개요

위의 세 가지 기능을 지원하기 위하여 DIS의 시간에 따른 동작은 다음과 같다.

1. 데이터 허브 시작
2. 관심 정보 업데이트
3. 데이터맵 동기화

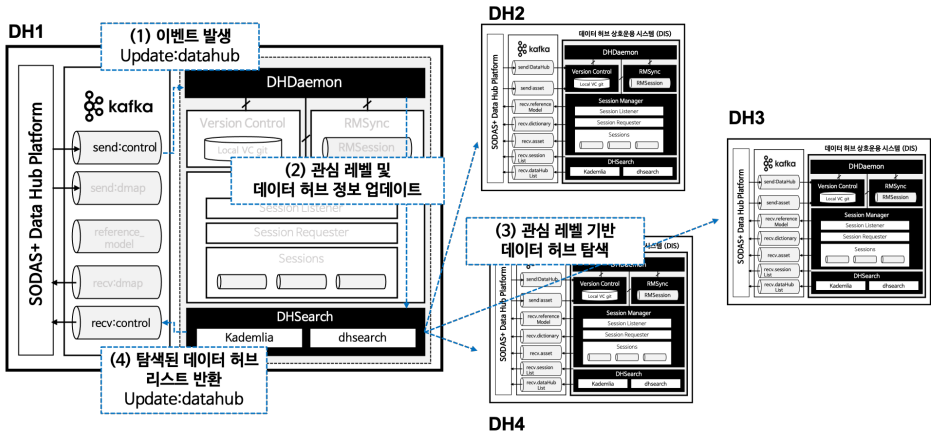
2.3.1 데이터 허브 시작

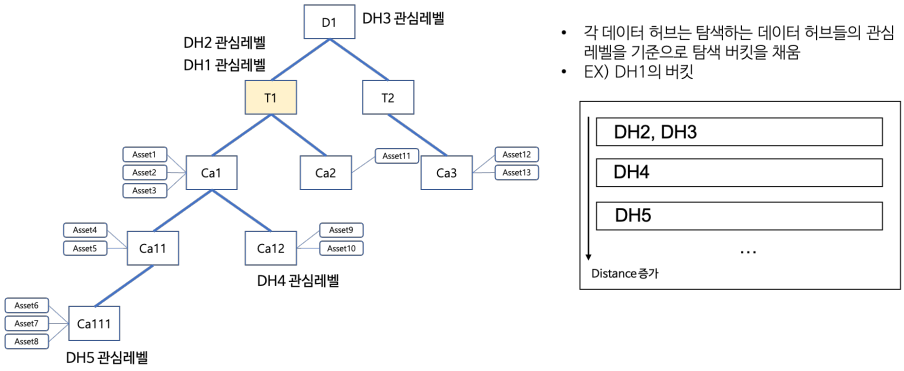
데이터 허브가 시작하면 데이터 허브 플랫폼‘은 ‘*START* 이벤트를 발생하고, DIS는 *START* 이벤트를 감지하여 GIS와의 통신을 시작한다. GIS와의 통신을 통해 현재 SODAS+ 플랫폼 내의 분산 데이터 허브 탐색을 위한 시드 노드 정보를 받아오고 이와 동시에 현재 ‘거버넌스 시스템‘에서 관리중인 표준 참조 모델을 가져와서 ‘데이터허브 플랫폼‘으로 전달한다.



2.3.2 관심 정보 업데이트

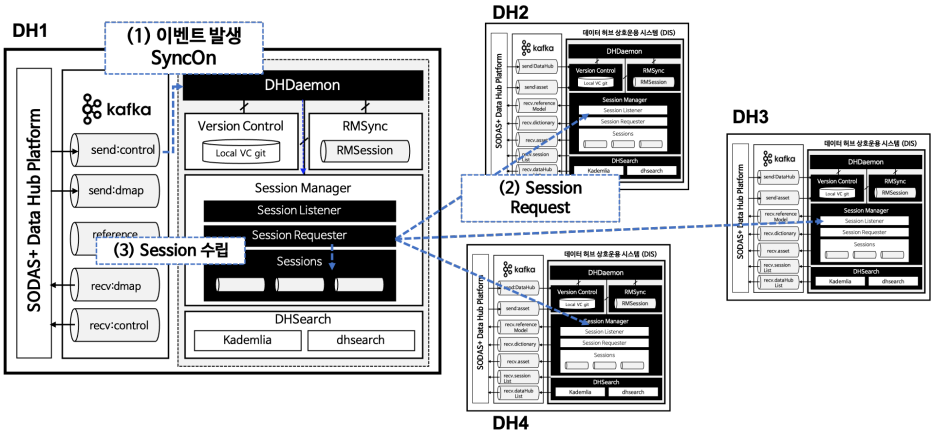
관심 정보 업데이트는 ETRI에서 개발한 SODAS+ Portal 웹페이지에서 발생된다. 이는 사용자에 의해 결정되거나 혹은 데이터 허브가 보유한 데이터 셋의 속성에 의해 결정된다. 데이터 허브의 관심 정보 (혹은 관심 레벨)는 SODAS+ 시스템 내에서 데이터 허브가 위치할 오버레이 네트워크를 결정한다. 다시 말해, 데이터 허브 탐색 시 타 데이터 허브와의 거리를 측정하기 위한 기준이 된다. 설정된 관심 정보를 바탕으로 데이터 허브의 고유 값이 수정되며, 이를 바탕으로 SODAS+ 네트워크를 재 탐색하여 거리 기반의 데이터 허브 버켓을 채운다. 관심 정보 업데이트 이벤트 (*UPDATE*) 가 발생한 경우 상호 운용 시스템의 동작은 다음 그림과 같다.



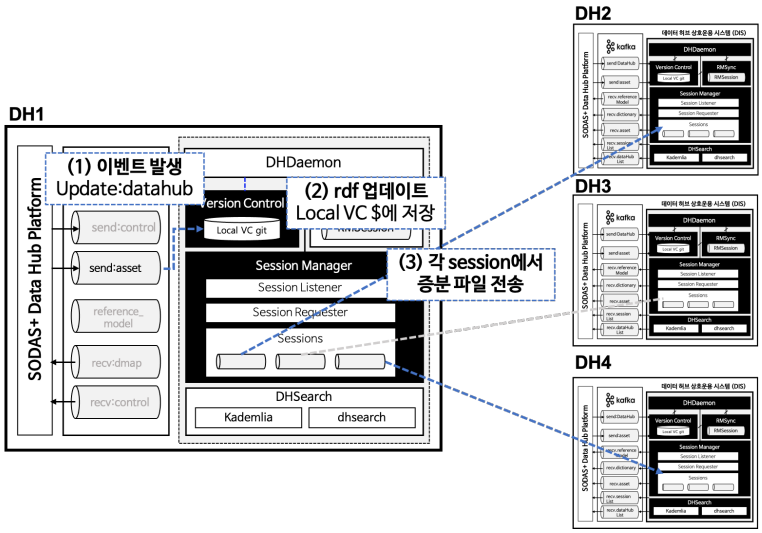


2.3.3 데이터맵 동기화

데이터 맵 동기화는 관심 레벨 설정이 수행된 이후 수행되어야 하며, 데이터맵 동기화는 사용자 포탈 혹은 CLI 명령어를 통해 트리거된다. SyncON 이벤트가 분산 동기화 시스템으로 전달되면 데이터 허브는 기존에 탐색 단계에서 가지고 있는 데이터 허브 버킷을 가까운 거리 순으로 순회하며 세션을 맺는다. 아래 그림은 `SYNC_ON` 이벤트를 감지하고 세션을 요청하는 프로세스를 보여준다.



세션이 연결된 이후에는 각 세션은 세션에서 협의된 내용을 바탕으로 파일 전송이 이루어진다. 다음 그림는 데이터맵 수정/생성/삭제 이벤트가 전달되었을 때 대응되는 세션에서의 능동적 동기화 시나리오를 보여준다.



2.4 데이터 맵 관리 시스템과의 연동 구조

DIS는 데이터맵 관리 시스템 과 연동되어 작동하며, 시스템으로부터 들어오는 이벤트에 따라 동작을 수행한다. 다음은 데이터맵 관리 시스템 과 DIS 간 통신을 위해 정의된 이벤트 종류를 보여준다.

표 2: 데이터 허브 상호 운용 시스템 (DIS) 및 데이터맵 관리 시스템 연동 기능에 따른 이벤트

구분	토픽	TX	RX	Operation
데이터 허브 제어	send.dataHub	DHPlatform	DIS	START UPDATE SYNC_ON
데이터 허브 에셋 동기화 관리	send.asset	DHPlatform	DIS	CREATE UPDATE
데이터 허브 에셋 동기화 관리	recv.asset	DIS	DHPlatform	CREATE UPDATE
데이터 허브 탐색	recv.dataHubList	DIS	DHPlatform	UPDATE
참조 모델 관리	recv.referenceModel	DIS	DHPlatform	CREATE UPDATE
딕셔너리 관리	recv.dictionary	DIS	DHPlatform	CREATE UPDATE
세션 정보 관리	recv.sessionList	DIS	DHPlatform	UPDATE

CHAPTER 3

Daemon

Daemon 모듈은 SODAS+ DIS 시스템으로 들어오는 이벤트 입력을 처리하고 DIS 시스템의 모듈을 가동하고 이벤트를 전달하는 역할을 수행한다. Daemon 모듈은 데이터 허브로부터 들어오는 Kafka 메시지를 관리하고 타 모듈들과 연동을 수행하는 *DHDaemon* 과 Kafka 와의 통신을 위한 *ctrlKafka* 모듈 그리고 DIS의 CLI 통신을 위한 *Daemon Server* 로 구성된다.

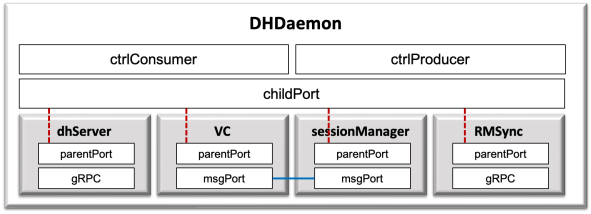
3.1 DHDaemon

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

DHDaemon 은 DIS의 데몬 모듈로, 데이터 허브와의 통신을 위한 이벤트를 처리하고 하위 모듈(*DHSearch*)로 전달하는 역할을 수행한다.



class DHDaemon()

DHDaemon 은 DIS 시스템의 데몬으로 생성자에서는 'setting.cfg'의 설정 내용을 바탕으로 DIS 프로그램에서 필요한 모듈들을 실행한다.

DHDaemon._dhSearchListener(*message*)

dhSearch 모듈로부터 전달되는 메시지를 처리하기위한 Listener로 UPDATE_BUCKET_LIST 이벤트가 들어오는 경우 탐색된 버킷 리스트를 recv.dataHubList 토픽으로 전송함.

인수

- **message** (dictionary(event, message)()) – 이벤트 종류와 메시지를 저장한 딕셔너리 구조

더 보기:

- *DHSearch._dmUpdateBucketList*

DHDaemon._dhSearchUpdateInterestTopic(*interestTopic*)

데이터 허브의 관심 주제 (interest topic)이 업데이트 되는 경우 dhSearch 모듈로 해당 이벤트를 전달하는 메서드

더 보기:

- *DHSearch._dhDaemonListener*

DHDaemon._dmServerListener(*message*)

데몬용 CLI API를 처리하기위한 daemon server listener

인수

- **message** (dictionary(event, message)()) – event와 message key 값을 가진 메시지

DHDaemon._dmServerSetBucketList(*bucket_list*)

CLI 데이터 업데이트를 위해 데몬이 가진 정보를 dmServer 모듈로 전달 수정된 버킷 리스트 정보를 dmServer 모듈로 전달함. [postMessage] to daemonServer

DHDaemon._dmServerSetRM(*referenceModel*)

CLI 데이터 업데이트를 위해 데몬이 가진 정보를 dmServer 모듈로 전달 수정된 참조 모델 (referenceModel) 정보를 dmServer 모듈로 전달함. [postMessage] to daemonServer

DHDaemon._dmServerSetSessionList(*sessionList*)

CLI 데이터 업데이트를 위해 데몬이 가진 정보를 dmServer 모듈로 전달 수정된 세션 리스트 정보를 dmServer 모듈로 전달함. [postMessage] to daemonServer

인수

- **sessionList** (list(string)()) –

DHDaemon._raiseError(*errorCode*)

에러 메시지 전달

DHDaemon._rmSyncInit()

rmSync 모듈로 INIT 메시지를 전달하는 메서드

더 보기:

- *RMSync._dhDaemonListener*

DHDaemon.**_rmSyncListener**(*message*)

rmSync 모듈 (참조 모델 동기화 모듈)로부터 전달되는 메시지를 처리하기 위한 Listener로 UPDATE_REFERENCE_MODEL 이벤트가 들어오는 경우 데이터 허브로 들어온 오픈 참조 모델을 데이터 허브로 전달

더 보기:

- *RMSync._dmUpdateReferenceModel*

DHDaemon.**_smInit**()

sessionManager 모듈로 INIT 메시지를 전달하는 메서드

더 보기:

- *SessionManager._dhDaemonListener*

DHDaemon.**_smListener**(*message*)

sessionManager 모듈로부터 전달되는 메시지를 처리하기 위한 Listener로 GET_SESSION_LIST_INFO 메시지가 들어오는 경우 획득된 session list 정보를 recv.sessionList 토픽으로 전달

더 보기:

- *SessionManager._dmGetSessionListInfo*

DHDaemon.**_smSyncOn**(*datahubs*)

sessionManager로 SYNC_ON 메시지를 전달하는 메서드로 datahub 아이디 리스트가 주어지면 해당 리스트의 노드 ID 정보로부터 버킷 정보를 추출하여 sessionManager 모듈에 SYNC_ON 메시지와 함께 전달 [postMessage] to sessionManager {'event':'SYNC_ON'}

인수

- **datahubs** (list(string)()) –

반환

number –

더 보기:

- *SessionManager._dhDaemonListener*

DHDaemon.**_smUpdateInterestTopic**(*interestTopic*)

sessionManager로 관심 토픽 정보를 전달하는 메서드

인수

- **interestTopic** (list(string)()) – 관심 토픽 정보 리스트

더 보기:

- *SessionManager._dhDaemonListener*

DHDaemon.**_smUpdateNegotiation**(*session_negotiation_option*)

sessionManager로 협상 옵션 정보를 전달하는 메서드

인수

- **session_negotiation_option** (dictionary()) –

더 보기:

- *SessionManager._dhDaemonListener*

`DHDaemon._vcInit()`

initialize Version Control module [postMessage] to versionControl module {'event':'INIT'}

`DHDaemon._vcListener(`*message*`)`

VersionControl 모듈로부터 전달되는 메시지를 처리하기 위한 Listener

`DHDaemon._vcUpdateReferenceModel(`*referenceModel*`)`

참조 모델 (reference model)을 governance system으로부터 전달받은 경우 (CREATE/UPDATE) 수정된 참조 모델을 바탕으로 파일 트리를 형성하도록 해당 정보를 version control 모듈로 전달함

 [postMessage] to versionControl module `{'event': 'UPDATE_REFERENCE_MODEL', 'data': referenceModel}`

`DHDaemon.init()`

DHDaemon 초기화 함수로, 처음 DIS가 시작할 때 호출됨. DIS와 DataHub 통신에 필요한 Kafka Topic 생성.

반환
Promise.<void> –

더 보기:

- *ctrlProducer.createCtrlTopics*

`DHDaemon.run()`

데몬 실행 함수로 각 서브 모듈들을 worker thread로 실행함. 모든 worker thread 실행 후, `recv.dataHub``로 들어오는 모든 이벤트를 처리하는 쓰레드 시작 (``ctrlConsumer.onMessage()`)

더 보기:

- *Daemon Server*
- *DHSearch*
- *VCModule*
- *SessionManager*
- *RMSync*
- *ctrlConsumer.onMessage*

`DHDaemon.stop()`

DIS 데몬 종료

3.2 ctrlKafka

Authors

Eunju Yang (yejyang@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

ctrlKafka 모듈은 DISDaemon과 data map 관리 시스템 사이의 통신을 위한 Kafka 라이브러리를 포함한다. 약속된 DISDaemon이 통신하기위한 *ctrlConsumer* 와 *ctrlProducer* 모듈을 포함한다.

3.2.1 ctrlConsumer

class ctrlConsumer(*kafkaHost*, *options*, *dhDaemon*, *conf*)

ctrlConsumer 타겟이 되는 kafka 정보를 받아들여 주어진 조건을 만족하는 kafka 로부터 'send.datahub' 토픽의 정보를 지속적으로 listening 하는 ctrlConsumer 객체 생성

인수

- **kafkaHost** (string()) – kafka Host 정보
- **options** (dictionary()) – options for kafka
- **dhDaemon** (*DHDaemon()*) – dhDaemon object
- **conf** (dictionary()) – configuration

ctrlConsumer.eventSwitch(*event*, *msg*)

send.datahub 로 들어오는 메시지의 event 형태에 따른 대응 START : reference model 동기화 시작 DHDaemon._rmSyncInit   STOP : DIS 동작 종료 (not yet implemented)   UPDATE : 관심 허브 정보 등록 DHDaemon._dhSearchUpdateInterestTopic , DHDaemon._smUpdateInterestTopic   SYNC_ON : 특정 데이터 허브와 동기화 시작 DHDaemon._smSyncOn

인수

- **event** (event()) – event {START, STOP, UPDATE, SYNC_ON}
- **msg** (string()) – detailed message

더 보기:

- *DHDaemon._rmSyncInit*
- *DHDaemon._dhSearchUpdateInterestTopic*
- *DHDaemon._smUpdateInterestTopic*
- *DHDaemon._smSyncOn*

ctrlConsumer.onMessage()

ctrlConsumer 의 onMessage 함수 해당 토픽으로 들어오는 메시지를 이벤트와 메시지로 파싱한 후 이벤트 종류에 따른 처리를 위해 ctrlConsumer.eventSwitch 로 전달

예외

error() – 메시지가 send.dataHub 의 규약을 따르지 않는 경우 에러 반환

반환

eventSwitch(event,msg) – eventSwitch

더 보기:

- *ctrlConsumer.eventSwitch*

3.2.2 ctrlProducer

class ctrlProducer(*kafkaHost*)

Kafka producer

인수

- **kafkaHost** (**string()**) – 카프카 정보

ctrlProducer._produce(*topic*, *msg*)

지정한 토픽으로 메시지를 전송하는 메서드

인수

- **topic** (**string()**) – 이벤트 토픽
- **msg** (**string()**) – 전송할 이벤트 스트링

ctrlProducer.createCtrlTopics()

카프카 토픽 생성하는 메서드로 DIS에서 사용하는 모든 토픽을 생성함. 해당 토픽이 이미 생성되어 있는 경우 생성하지 않으며, 토픽이 없는 경우 시스템이 동작할 수 없으므로 모든 토픽이 생성된 후 반환 됨.

반환

Promise.<void> – createTopics()

ctrlProducer.sendError(*errorCode*)

에러 메시지 전달

더 보기:

- *ctrlProducer._produce*

3.3 Daemon Server

Authors

Eunju Yang (yejyang@kaist.ac.kr)

Version

3.0.0 of 2022.11.30

DaemonServer 는 DIS의 CLI 를 지원하기위한 CLI Server로 kafka event기반의 컨트롤 통신 외에 CLI Client를 사용한 제어 및 조회를 위한 API를 제공한다

class dServer()

daemonServer 클래스로, DIS 시스템의 client CLI를 지원하기위한 서버 모듈

dServer._dmSetInterest(*interestList*)

setInterest 동작

dServer._dmStart()

daemonStart 동작

dServer._dmSyncOn()

daemonSyncOn 동작

dServer._parentSwitch(*message*)

메시지 스위칭을 위한 private method

`dServer.getDaemonServer()`
daemonServer (gRPC server) 정보 반환

반환
* –

`dServer.getDhList(call, callback)`
데이터 허브 리스트 조회 API

`dServer.getSessionList(call, callback)`
세션 리스트 조회 API

`dServer.setInterest(call, callback)`
관심 토픽 설정 API

`dServer.start()`
start API

`dServer.startSignal(call, callback)`
startSignal API

`dServer.syncOnSignal(call, callback)`
syncOn API

CHAPTER 4

DHSearch

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

DHSearch 모듈은 SODAS+ DIS *DHDaemon* 에 의해 실행되는 모듈로, 거버넌스 시스템(GS)의 Bootstrap 서버로부터 최근 SODAS+ 플랫폼에 접속한 SeedNode의 접속 정보를 조회한 뒤, SeedNode 정보를 통해 관심도 기반 데이터 허브 분산 탐색하는 역할을 수행한다.

DHSearch 모듈은 *DHDaemon* 와의 연동을 통한 Control Event 메시지를 처리하는 *DHSearch* 와 거버넌스 시스템(GS)의 Bootstrap 서버로부터 gRPC 기반 SeedNode 정보를 조회하기 위한 gRPC 프로토버퍼 *Bootstrap.proto*, 그리고 관심도 기반 데이터 허브 분산 탐색 프로토콜이 구현된 *Kademlia* 로 구성된다.

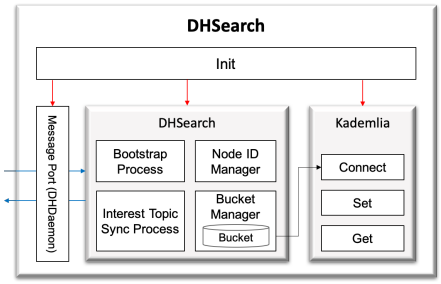
4.1 DHSearch

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

DHSearch 는 GIS BootstrapServer 에 SeedNode 리스트를 조회하는 기능과 분산 탐색 프로토콜을 구현한 *Kademlia* 라이브러리 기능을 활용하여 오픈 데이터 생태계의 데이터 허브를 탐색하면서 end point, 관심 동기화 수준, 메타데이터 정보를 수집/관리하여, *DHDaemon* 으로 전달한다.



`class DHSearch()`

DHSearch

`DHSearch._bootstrapProcess()`

GIS BootstrapServer 로 SeedNode 리스트를 조회한 결과를 내부 변수 `seedNodeList` 에 저장함.

반환

Promise - null

더 보기:

- *DHSearch.run*
- *DHSearch.getSeedNode*

`DHSearch._deleteMyInfoFromKademlia()`

분산 탐색 네트워크에서 연결을 해제하는 *Kademlia* `KNode.delete` 함수를 통해, 분산 탐색 네트워크의 다른 데이터 허브의 Bucket 에서 해당 데이터 허브의 노드 정보를 삭제 요청함.

반환

Promise - null

더 보기:

- *DHSearch._dhDaemonListener*
- *KNode.delete*

`DHSearch._dhDaemonListener(`*message*, *message:event*`)`

DHDaemon 에서 전달되는 스레드 메시지를 수신하는 이벤트 리스너.

인수

- **`message`** – dictionary(event, message) 구조의 스레드 메시지
- **`message:event`** – `UPDATE_INTEREST_TOPIC`, `DIS_STOP`

더 보기:

- *DHDaemon._dhSearchUpdateInterestTopic*
- *DHSearch._deleteMyInfoFromKademlia*
- *DHSearch._updateInterestInfo*
- *KNode.delete*

`DHSearch._discoverProcess()`

GIS BootstrapServer 로부터 조회한 SeedNode 리스트의 DataHub end point 를 통해, 분산 탐색 네트워크에 연결하는 *Kademlia* `KNode.connect` 함수를 통해 분산 탐색 네트워크 참여함.

반환

Promise - null

더 보기:

- *DHSearch.run*
- *KNode.connect*

DHSearch.**_dmUpdateBucketList**()

분산 탐색 네트워크에 연결한 뒤 Bucket 정보가 업데이트될 때마다, Bucket 정보를 *DHDaemon* 로 UPDATE_BUCKET_LIST 스레드 메시지를 전송함.

더 보기:

- *DHDaemon._dhSearchListener*

DHSearch.**_updateInterestInfo**(*messageData*)

UPDATE_INTEREST_TOPIC 이벤트 스레드 메시지와 함께 전달된 데이터(관심 동기화 수준, DataHub 메타데이터)를 내부 변수에 업데이트한 뒤, Bootstrap-DistributeSearch 로직을 수행하는 내부 함수 호출함.

인수

- **messageData** – UPDATE_INTEREST_TOPIC 이벤트 스레드 메시지와 함께 전달된 데이터(관심 동기화 수준, DataHub 메타데이터)

반환

Promise - null

더 보기:

- *DHSearch._dhDaemonListener*
- *DHSearch.run*

DHSearch.**getSeedNode**(*seedNode*)

GIS BootstrapServer 로 SeedNode 리스트를 조회하는 gRPC client.

인수

- **seedNode** – GIS BootstrapServer 의 SeedNode 리스트에 등록할 자신의 노드 정보

반환

Promise - GIS BootstrapServer 로부터 조회한 SeedNode 리스트

더 보기:

- *DHSearch._bootstrapProcess*

DHSearch.**run**()

DHDaemon 으로부터 UPDATE_INTEREST_TOPIC 이벤트 수신 후, GIS BootstrapServer 로부터 SeedNode 리스트 조회 및 조회한 SeedNode end point 를 통해, 분산 탐색 네트워크에 연결하는 로직을 수행함.

더 보기:

- *DHSearch._updateInterestInfo*
- *DHSearch._bootstrapProcess*
- *DHSearch._discoverProcess*

4.2 Kademlia

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

Kademlia 는 *DHSearch* 의 데이터 허브 분산 탐색 프로토콜이 구현된 라이브러리 모듈로, 데이터 허브와의 통신을 위한 이벤트를 처리하고 하위 모듈(*DHSearch*)로 전달하는 역할을 수행한다.

4.2.1 KNode

`class KNode(`*desc*`)`
KNode

인수

- **desc** – UDP 통신 수신자 end point

`KNode._MSG(`*type*, *params*`)`
분산 탐색 네트워크에서 통신하는 UDP 프로토콜 메시지 규격으로 변환해주는 함수.

인수

- **type** – UDP 메시지 type
- **params** – 전송할 메시지

반환
UDP_Message

`KNode._findClosestNodes(`*key*, *howMany*, *exclude*`)`
Bucket 내 XOR 거리를 비교할 key 값과의 거리가 가까운 노드들의 정보(contacts)를 반환함.

인수

- **key** – XOR 거리를 비교할 key 값
- **howMany** – contact 를 저장할 수 있는 contacts 의 최대 크기
- **exclude** – 비교 대상에서 제외할 nodeID(노드 자기 자신에 해당할 경우 제외.)

반환
contacts

`KNode._iterativeFind(`*key*, *mode*, *cb*`)`
분산 탐색 네트워크를 순회하면서 key 값에 해당하는 contact 객체를 탐색하면서, mode(NODE, VALUE)에 따라 UDP 통신을 전송하는 함수.

인수

- **key** – 조회할 key or 노드 ID 값
- **mode** – NODE, VALUE
- **cb** – callback 함수

KNode.**_iterativeFindNode**(*nodeID*, *cb*)

분산 탐색 네트워크를 순회하면서 노드 ID 값에 해당하는 contact 객체를 탐색하는 함수.

인수

- **nodeID** – 조회할 contact 노드 ID
- **cb** – callback 함수

더 보기:

- *KNode.set*

KNode.**_iterativeFindValue**(*key*, *cb*)

분산 탐색 네트워크를 순회하면서 key 값에 해당하는 value(contact 객체 정보)를 탐색하는 함수.

인수

- **key** – 조회할 contact 객체 정보에 해당하는 key 값
- **cb** – callback 함수

더 보기:

- *KNode.get*

KNode.**_onDelete**(*message*)

수신한 UDP 메시지의 type 이 DELETE 에 해당할 때 호출되는 함수로, 메시지 내 contact 객체 정보를 Bucket 내에서 삭제하는 통신에 해당함.

인수

- **message** – 수신한 UDP 메시지

KNode.**_onFindNode**(*message*)

수신한 UDP 메시지의 type 이 FIND_NODE 에 해당할 때 호출되는 함수로, message.key 와 XOR 거리 기반 가까운 노드들의 정보(contacts)를 반환하는 통신에 해당함.

인수

- **message** – 수신한 UDP 메시지

KNode.**_onFindValue**(*message*)

수신한 UDP 메시지의 type 이 FIND_VALUE 에 해당할 때 호출되는 함수로, message.key 값에 해당하는 value(contact 객체 정보)를 반환하는 통신에 해당함.

인수

- **message** – 수신한 UDP 메시지

KNode.**_onMessage**(*message*)

분산 탐색 네트워크에서 통신하는 노드 중 UDP 메시지를 수신하는 노드에서 UDP 메시지의 type 에 따라, 내부 처리 함수를 호출하는 브로커 함수.

인수

- **message** – 수신한 UDP 메시지

더 보기:

- *KNode._onPing*
- *KNode._onStore*
- *KNode._onDelete*
- *KNode._onFindValue*

KNode.**_onPing**(*message*)

수신한 UDP 메시지의 type 이 PING 에 해당할 때 호출되는 함수로, 분산 탐색 네트워크의 노드 간 네트워크 연결 여부를 확인하는 통신에 해당함.

인수

- **message** – 수신한 UDP 메시지

KNode.**_onStore**(*message*)

수신한 UDP 메시지의 type 이 STORE 에 해당할 때 호출되는 함수로, 특정 contact 정보를 key, value 로 저장하는 통신에 해당함.

인수

- **message** – 수신한 UDP 메시지

더 보기:

- *KNode.set*

KNode.**_onStoreReply**()

수신한 UDP 메시지의 type 이 STORE_REPLY 에 해당할 때 호출되는 함수로, STORE 통신의 응답에 해당함.

더 보기:

- *KNode._onStore*

KNode.**_refreshBucket**(*bucketIndex*, *callback*)

새로운 contact 가 Bucket 에 추가되면서, Bucket 내 contact 간 XOR 거리 기반 가까운 순으로 재정렬하는 함수.

인수

- **callback** – callback 함수

KNode.**_updateContact**(*contact*, *cb*)

인자로 주어진 contact 객체 정보를 Bucket 에 업데이트하는 함수로, contact 객체 정보가 이미 Bucket 내에 있을 경우 최신 정보로 업데이트하며, contact 객체 정보가 신규 추가에 해당하고 동시에 Bucket Size 가 여유로울 때는 Bucket 에 신규 추가하며, Bucket Size 가 꽉 찼을 때는 Bucket 내 노드들에게 PING 메시지를 전송해서 응답하지 않는 contact 는 제거한 뒤, 신규 contact 를 추가함.

인수

- **contact** – Bucket 에 업데이트할 contact 객체 정보
- **cb** – callback 함수

KNode.**connect**(*address*, *port*, *sl_portNum*, *sync_interest_list*, *metadata*, *cb*)

분산 탐색 네트워크에 연결하는 주요 API 로, 인자로 주어진 SeedNode 리스트 정보로 contact 객체를 생성한 뒤, 해당 contact 객체의 end point 정보를 통한 UDP 통신 기반 분산 탐색 네트워크에 연결함.

인수

- **address** – 분산 탐색 네트워크에 신규 참여할 DataHub 의 IP 주소
- **port** – 분산 탐색 네트워크에 신규 참여할 DataHub 의 Port 번호
- **sl_portNum** – 분산 탐색 네트워크에 신규 참여할 DataHub 의 *SessionListener* gRPC 서버 Port 번호
- **sync_interest_list** – 분산 탐색 네트워크에 신규 참여할 DataHub 의 관심 동기화 수준 리스트
- **metadata** – 분산 탐색 네트워크에 신규 참여할 DataHub 의 메타데이터

- **cb** – callback 함수

더 보기:

- *DHSearch._discoverProcess*

KNode.**debug**()

디버그용 Bucket 내 contact 정보를 콘솔에 출력하는 함수.

KNode.**delete**(*address*, *port*, *sl_portNum*, *sync_interest_list*, *metadata*, *isDisStop*, *cb*)

주어진 인자로 contact 객체를 생성한 뒤, 분산 탐색 네트워크에 접속한 모든 데이터 허브들의 정보(Bucket 객체)를 조회하면서, 생성한 contact 정보를 삭제 요청하는 UDP 통신을 전송함.

인수

- **address** – 분산 탐색 네트워크에 신규 참여할 DataHub 의 IP 주소
- **port** – 분산 탐색 네트워크에 신규 참여할 DataHub 의 Port 번호
- **sl_portNum** – 분산 탐색 네트워크에 신규 참여할 DataHub 의 *SessionListener* gRPC 서버 Port 번호
- **sync_interest_list** – 분산 탐색 네트워크에 신규 참여할 DataHub 의 관심 동기화 수준 리스트
- **metadata** – 분산 탐색 네트워크에 신규 참여할 DataHub 의 메타데이터
- **isDisStop** – Bucket 정보 업데이트 Notification 간결화를 위한 Boolean
- **cb** – callback 함수

더 보기:

- *DHSearch._dhDaemonListener*
- *DHSearch._deleteMyInfoFromKademlia*

KNode.**get**(*key*, *cb*)

KNode.set 함수로 저장한 contact 정보를 key 값을 통해 조회하는 함수.

인수

- **key** – 조회할 contact 객체 정보에 해당하는 key 값
- **cb** – callback 함수

더 보기:

- *KNode._iterativeFindValue*
- *KNode.set*

KNode.**set**(*key*, *value*, *cb*)

분산 탐색 네트워크에서 key, value 값으로 특정 contact 정보를 저장하는 함수.

인수

- **key** – contact 객체 정보에 해당하는 key 값
- **value** – 저장할 contact 객체 정보
- **cb** – callback 함수

더 보기:

- *KNode._iterativeFindNode*
- *KNode.get*

KNode.**toString**()

분산 탐색 네트워크에 등록될 자신의 노드 정보를 문자열로 변환한 뒤 반환함.

반환

string

4.2.2 Bucket

class Bucket()

Bucket

Bucket.**add**(*contact*)

Bucket 내 관리하고 있는 Contact List 에 새로 추가될 contact 를 lastSeen 기준으로 정렬한 위치에 추가함.

인수

- **contact** – 새로 추가할 contact 객체

반환

Bucket - 새로 추가된 contact 정보를 포함한 Bucket 객체

Bucket.**contacts**()

Bucket 내 관리하고 있는 Contact List 객체의 clone 을 반환.

반환

contacts - Contact List 객체의 clone

Bucket.**contains**(*contact*)

contact 의 nodeID 를 이용하여 Bucket 내 관리하고 있는 Contact List 에 해당 contact 객체가 있는지 유무를 반환.

인수

- **contact** – Contact List 내 포함 유무를 확인할 contact 객체

반환

boolean - Contact List 내 contact 객체의 포함 유무

Bucket.**findContact**(*id*)

Bucket 내 관리하고 있는 Contact List 에서 인자로 주어진 id 와 일치하는 contact

인수

- **id** – 조회할 nodeID

반환

contact - 주어진 id 와 일치하는 contact 객체

Bucket.**get**(*index*)

Bucket 내 관리하고 있는 Contact List 의 index 번째에 해당하는 contact 객체 반환.

인수

- **index** – 조회할 contact 객체의 순번

반환

contact - index 번째의 contact 객체

Bucket.**indexOf**(*contact*)

Bucket 내 관리하고 있는 Contact List 에서 인자로 주어진 contact 의 nodeID 와 일치하는 contact 객체의 순번을 반환함.

인수

- **contact** – 순번을 조회할 contact 객체

반환

(number)index - Contact List 내 인자로 주어진 contact 의 nodeID 와 일치하는 contact 객체의 순번

Bucket.**remove**(*contact*)

Bucket 내 관리하고 있는 Contact List 에서 인자로 주어진 contact 의 nodeID 와 일치하는 contact 정보를 삭제함.

인수

- **contact** – 삭제할 contact 객체

반환

Bucket - 인자로 주어진 contact 정보를 삭제한 Bucket 객체

Bucket.**removeIndex**(*index*)

Bucket 내 관리하고 있는 Contact List 에서 인자로 주어진 index 순번에 해당하는 contact 정보를 삭제함.

인수

- **index** – 삭제할 contact 순번

반환

Bucket - index 순번에 해당하는 contact 정보를 삭제한 Bucket 객체

Bucket.**size**()

Bucket 내 관리하고 있는 Contact List 내 contact 개수를 반환.

반환

contacts.length - Contact List 내 contact 개수

Bucket.**toString**()

Bucket 내 관리하고 있는 Contact List 를 문자열로 변환한 뒤, 반환함.

반환

JSON.stringify(contacts) - Contact List 의 문자열 변환값

4.2.3 RPC

class RPC(*bindAddress*, *callback*)

RPC

RPC.**_expireRPCs**()

UDP 통신 중 RESPONSE TIMEOUT 을 초과할 경우, 해당 통신을 종료함.

RPC.**_onMessage**(*data*)

UDP 통신을 전송하는 함수.

인수

- **data** – UDF 메시지

RPC.**close**()

기 연결되어 있는 UDP 통신 소켓을 종료하는 함수.

RPC.**send**(*node*, *message*)

분산 탐색 네트워크 내 데이터 허브 간 UDP 통신을 전송함.

인수

- **node** – UDP 통신을 전송할 대상 노드 end point

- **message** – (buffer) UDP 통신을 통해 전송할 데이터

4.3 Bootstrap.proto

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

CHAPTER 5

RMSync

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

RMSync 모듈은 SODAS+ DIS *DHDaemon* 에 의해 실행되는 모듈로, SODAS+ 생태계 내 오픈 참조 모델을 관리하는 Governance System 과의 오픈 참조 모델 동기화를 위한 세션 연동 및 해당 세션을 통한 오픈 참조 모델 동기화를 수행하는 모듈이다.

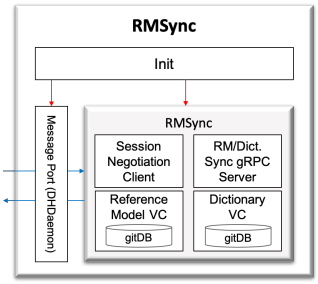
5.1 RMSync

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

RMSync 는 GIS RMSessionManager 와 오픈 참조 모델(Reference Model)과 딕셔너리(Dictionary) 동기화를 위한 세션 협상을 수행한 뒤, GIS RMSession 과 세션 연동하고, 오픈 참조 모델과 딕셔너리 동기화 수신을 위한 gRPC 서버를 구동함으로써 추후 GIS 에서 업데이트되는 오픈 참조 모델과 딕셔너리를 동기화한다.



---

class RMSync()

RMSync

RMSync.Subscribe(*self*)

오픈 참조 모델의 변경점만 추출한 git patch 파일을 전달받아 VC 기반 오픈 참조 모델 동기화를 수행 및 *DHDaemon* 로 `UPDATE_REFERENCE_MODEL` 스레드 메시지를 전송하는 내부 함수 호출.

인수

- **self** – RMSync 객체 (내부 변수 접근용)

더 보기:

- *RMSync._setRMSyncServer*
- *RMSync.gitPatch*
- *RMSync._dmUpdateReferenceModel*

RMSync._dhDaemonListener(*message*, *message:event*)

DHDaemon 에서 전달되는 스레드 메시지를 수신하는 이벤트 리스너

인수

- **message** – dictionary(event, message) 구조의 스레드 메시지
- **message:event** – `INIT`

더 보기:

- *DHDaemon._rmSyncInit*

RMSync._dmUpdateReferenceModel(*path_list*, *operation*)

GIS 로부터 오픈 참조 모델 동기화 전송을 받은 후, 업데이트된 오픈 참조 모델의 파일 경로와 KAFKA 이벤트 메시지 생성을 위한 업데이트된 오픈 참조 모델의 변경 operation(CREATE, UPDATE)를 *DHDaemon* 로 `UPDATE_REFERENCE_MODEL` 스레드 메시지를 전송함.

인수

- **path_list** – 업데이트된 오픈 참조 모델의 파일 경로
- **operation** – 업데이트된 오픈 참조 모델의 변경 operation(CREATE, UPDATE)

더 보기:

- *DHDaemon._rmSyncListener*
- *RMSync.Subscribe*

RMSync._setRMSyncServer()

GIS 로부터의 오픈 참조 모델 동기화 수신을 위한 gRPC 기반 세션 서버를 구동함. 이후 GIS 에서 오픈 참조 모델이 업데이트될 경우, RMSync gRPC 기반 세션 서버의 Subscribe 함수를 호출하여, 오픈 참조 모델의 변경점만 추출한 git patch 파일을 전달받아 VC 기반 오픈 참조 모델 동기화를 수행함.

더 보기:

- *RMSync.run*
- *RMSync.Subscribe*

RMSync.gitPatch(*git_patch*, *self*)

오픈 참조 모델의 변경점만 추출한 git patch 파일을 전달받아, VC 모듈의 git apply 함수를 통한 로컬 gitDB 에 오픈 참조 모델 동기화(파일 저장)를 수행함.

인수

- **git_patch** – 오픈 참조 모델의 변경점만 추출한 git patch 파일
- **self** – RMSync 객체 (내부 변수 접근용)

더 보기:

- *RMSync.Subscribe*

RMSync.requestRMSession()

GIS RMSessionManager 로 오픈 참조 모델 동기화를 위한 세션 연동을 요청하는 gRPC 통신을 전송하며, 이때, DataHub 의 ID 와 기 구동한 gRPC 기반 세션 서버의 IP, Port 를 전송함.

더 보기:

- *RMSync.run*

RMSync.run()

DHDaemon 으로부터 INIT 이벤트 수신 후, GIS 로부터의 오픈 참조 모델 동기화 수신을 위한 gRPC 기반 세션 서버 구동 및 GIS RMSessionManager 로 세션 연동을 요청하는 RMSync 주요 로직을 수행

더 보기:

- *RMSync._setRMSyncServer*
- *RMSync.requestRMSession*

5.2 RMSession.proto

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version

3.0.0 of 2022.11.30

5.3 RMSessionSync.proto

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

CHAPTER 6

SessionManager

Authors

Eunju Yang (yejyang@kaist.ac.kr)
Ji-Hwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

SessionManager 모듈은 SODAS+ DIS *DHDaemon* 에 의해 실행되는 모듈로, *DHSearch* 에서 탐색한 관심 동기화 수준의 데이터맵을 소유한 다른 데이터 허브와의 데이터맵 동기화를 위한 세션 연동 기능을 수행하며, VersionControl 모듈과 연동하여 다른 데이터 허브와의 세션을 통한 데이터맵 전송 기능을 수행한다.

SessionManager 모듈은 다른 데이터 허브의 *SessionListener* 로 데이터맵 동기화 세션 연동을 위한 세션 협상 요청을 보내는 *SessionRequester* 와 다른 데이터 허브의 세션 협상 요청을 처리하는 *SessionListener* 로 구성된다.

데이터 허브 간 세션 협상 성공이 될 경우 *Session* 이 구동되며, 세션 협상 과정에서 합의한 동기화 수준에 해당하는 데이터맵이 변경될 경우 상대 데이터 허브의 *Session* 으로 변경된 부분만 증분 추출하여 동기화 전송한다.

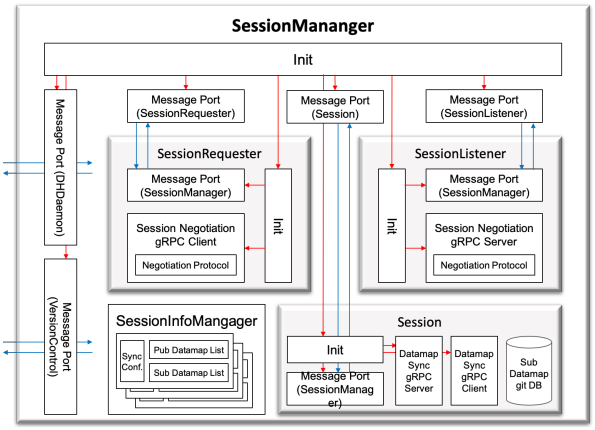
6.1 SessionManager

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

SessionManager 모듈은 *SessionRequester*, *SessionListener*, *Session* 모듈을 생성 및 관리하는 모듈로, 데이터 허브 간 세션 협상이 체결될 경우, 연동된 세션 리스트 정보를 관리를 주요 기능으로 한다.



---

class SessionManager()

SessionManager

SessionManager._createSession()

Session 모듈을 worker thread 로 실행하고, 미사용 Port 번호를 할당하는 함수.

SessionManager._dhDaemonListener(*message*, *message:event*)

DHDaemon 에서 전달되는 스레드 메시지를 수신하는 이벤트 리스너.

인수

- **message** – dictionary(event, message) 구조의 스레드 메시지
- **message:event** – UPDATE_INTEREST_TOPIC, UPDATE_NEGOTIATION_OPTIONS, SYNC_ON

더 보기:

- *DHDaemon._smInit*
- *DHDaemon._smUpdateInterestTopic*
- *DHDaemon._smUpdateNegotiation*
- *DHDaemon._smSyncOn*

SessionManager._dmGetSessionListInfo()

SessionRequester 와 *SessionListener* 간 세션 연동이 되어 세션 리스트가 업데이트된 경우, *DHDaemon* 으로 업데이트된 세션 리스트를 GET_SESSION_LIST_INFO 이벤트 스레드 메시지로 전달함.

더 보기:

- *DHDaemon._smListener*

SessionManager._isEmptyObj(*obj*)

인자로 주어진 dictionary 가 빈 객체인지를 체크하는 함수.

인수

- **obj** – 빈 객체인지를 확인하기 위한 dictionary

SessionManager.**_refactoringSessionInfo**(*tempSession*, *otherNodeId*)

SessionRequester 와 *SessionListener* 간 세션 연동이 되어 세션 리스트가 업데이트된 경우, 업데이트된 내부 세션 관리용 dictionary 에서 *DHDaemon* 으로 전달하는 세션 리스트로 자료구조를 리팩토링하는 함수.

인수

- **tempSession** – 신규 연동된 임시 세션 객체
- **otherNodeId** – 세션 연동을 할 상대 데이터 허브의 ID

SessionManager.**_sessionInit**(*sessionWorker*)

Session worker thread 의 초기화 실행을 위한 함수로 INIT 이벤트 스레드 메시지를 전달함.

인수

- **sessionWorker** – 신규 생성한 세션 worker thread 객체

더 보기:

- *Session._smListener*

SessionManager.**_sessionListener**(*message*, *message:event*)

Session 에서 전달되는 스레드 메시지를 수신하는 이벤트 리스너.

인수

- **message** – dictionary(event, message) 구조의 스레드 메시지
- **message:event** – RECONFIGURATION_NEGOTIATION_OPTIONS

SessionManager.**_sessionTransmitNegotiationResult**(*sessionWorker*, *end_point*, *session_desc*, *sn_options*)

SessionRequester 와 *SessionListener* 간 세션 협상 체결이 된 경우, *Session* 로 상대 세션 모듈의 end point 정보와 세션 협상 결과를 TRANSMIT_NEGOTIATION_RESULT 이벤트 스레드 메시지와 함께 전달함.

인수

- **sessionWorker** – 세션 협상 체결 이후, 다른 데이터 허브의 세션 모듈과 연동할 내부 세션 객체
- **end_point** – 다른 데이터 허브의 세션 모듈의 접속 정보(IP, Port)
- **session_desc** – 세션 객체의 메타데이터(세션 생성자 정보, 세션 ID)
- **sn_options** – 세션 협상 결과

더 보기:

- *Session._smListener*

SessionManager.**_sessionUpdatePubAsset**(*sessionWorker*, *commit_number*)

versionControl 에서 send.asset 토픽을 통해 데이터맵 변화 이벤트 감지 및 git commit 실행한 뒤, SessionManager 모듈로 UPDATE_PUB_ASSET 이벤트 스레드 메시지로 전달하고, SessionManager 모듈은 업데이트된 데이터맵 수준을 포함하는 관심 동기화 수준으로 협상된 *Session* 모듈로 commit number 를 UPDATE_PUB_ASSET 이벤트 스레드 메시지와 함께 전달함.

인수

- **sessionWorker** – 변경된 Pub 데이터맵이 동기화 수준에 해당하는 세션 모듈 객체
- **commit_number** – VC 모듈에서 최근 커밋한 커밋 번호

더 보기:

- *Session._smListener*

SessionManager._setSessionPort()

신규 *Session* 모듈 생성을 위한, 미사용 Port 번호를 조회하는 함수.

SessionManager._slGetNewSessionInfo()

SessionListener 로 세션 협상 과정에서 필요한 *Session* worker thread 의 end point 정보를 GET_NEW_SESSION_INFO 이벤트 스레드 메시지와 함께 전달함.

더 보기:

- *SessionListener._smListener*

SessionManager._slInit()

SessionListener worker thread 의 초기화 실행을 위한 함수로 INIT 이벤트 스레드 메시지를 전달함.

더 보기:

- *SessionListener._smListener*

SessionManager._slListener(*message*, *message:event*)

SessionListener 에서 전달되는 스레드 메시지를 수신하는 이벤트 리스너.

인수

- **message** – dictionary(event, message) 구조의 스레드 메시지
- **message:event** – TRANSMIT_NEGOTIATION_RESULT

더 보기:

- *SessionListener._smTransmitNegotiationResult*

SessionManager._slUpdateInterestList()

DHDaemon 으로부터 UPDATE_INTEREST_TOPIC 이벤트 스레드 메시지를 받으면 실행되는 함수로, *SessionListener* 로 업데이트된 관심 동기화 수준 정보를 UPDATE_INTEREST_LIST 이벤트 스레드 메시지와 함께 전달함.

더 보기:

- *SessionListener._smListener*

SessionManager._slUpdateNegotiationOptions()

DHDaemon 으로부터 UPDATE_NEGOTIATION_OPTIONS 이벤트 스레드 메시지를 받으면 실행되는 함수로, *SessionListener* 로 업데이트된 세션 협상 옵션 정보를 UPDATE_NEGOTIATION_OPTIONS 이벤트 스레드 메시지와 함께 전달함.

더 보기:

- *SessionListener._smListener*

SessionManager._srGetNewSessionInfo()

SessionRequester 로 세션 협상 과정에서 필요한 *Session* worker thread 의 end point 정보를 GET_NEW_SESSION_INFO 이벤트 스레드 메시지와 함께 전달함.

더 보기:

- *SessionRequester._smListener*

SessionManager._srInit()

SessionRequester worker thread 의 초기화 실행을 위한 함수로 INIT 이벤트 스레드 메시지를 전달함.

더 보기:

- *SessionRequester._smListener*

SessionManager._srListener(*message*, *message:event*)

SessionRequester 에서 전달되는 스레드 메시지를 수신하는 이벤트 리스너.

인수

- **message** – dictionary(event, message) 구조의 스레드 메시지
- **message:event** – TRANSMIT_NEGOTIATION_RESULT

더 보기:

- *SessionRequester._smTransmitNegotiationResult*

SessionManager._srStartSessionConnection(*bucketList*)

DHDaemon 으로부터 SYNC_ON 이벤트 스레드 메시지를 받으면 실행되는 함수로, *SessionRequester* 로 bucketList 내 데이터 허브 리스트와의 세션 협상을 시작하는 START_SESSION_CONNECTION 이벤트 스레드 메시지를 전달함.

인수

- **bucketList** – 데이터맵 동기화 세션 연동을 위한 세션 협상 대상에 해당하는 데이터 허브 리스트

더 보기:

- *SessionRequester._smListener*

SessionManager._srUpdateInterestList()

DHDaemon 으로부터 UPDATE_INTEREST_TOPIC 이벤트 스레드 메시지를 받으면 실행되는 함수로, *SessionRequester* 로 업데이트된 관심 동기화 수준 정보를 UPDATE_INTEREST_LIST 이벤트 스레드 메시지와 함께 전달함.

더 보기:

- *SessionRequester._smListener*

SessionManager._srUpdateNegotiationOptions()

DHDaemon 으로부터 UPDATE_NEGOTIATION_OPTIONS 이벤트 스레드 메시지를 받으면 실행되는 함수로, *SessionRequester* 로 업데이트된 세션 협상 옵션 정보를 UPDATE_NEGOTIATION_OPTIONS 이벤트 스레드 메시지와 함께 전달함.

더 보기:

- *SessionRequester._smListener*

SessionManager._vcListener(*message*, *message:event*)

versionControl 에서 전달되는 스레드 메시지를 수신하는 이벤트 리스너.

인수

- **message** – dictionary(event, message) 구조의 스레드 메시지
- **message:event** – UPDATE_PUB_ASSET

`SessionManager.run()`

SessionManager 실행 함수로 *SessionRequester*, *SessionListener* 모듈들을 worker thread 로 실행하고, 각 모듈들의 초기화 실행을 위해 `INIT` 메시지를 전달함.

더 보기:

- *SessionRequester*
- *SessionListener*

6.2 SessionRequester

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

SessionRequester 모듈은 *SessionManager* 모듈에 의해 실행되는 모듈로, *SessionManager* 으로부터 Control Event(*SYNC_ON*) 메시지를 처리한다.

주요 기능으로는 *SYNC_ON* 메세지와 함께 전달받은 세션 협상을 요청할 다른 데이터 허브 정보(*Bucket*)를 조회하며, 다른 데이터 허브의 *SessionListener* 로 세션 협상을 gRPC 통신(*SessionNegotiation.proto*)을 통해 요청한다.

`class SessionRequester()`

SessionRequester

`SessionRequester._closeConnection()`

다른 데이터 허브의 *SessionListener* gRPC 서버와의 통신 종료.

`SessionRequester._initConnection(sl_ip)`

세션 협상 요청을 보낼 다른 데이터 허브의 *SessionListener* gRPC 서버와 통신할 gRPC client 객체 생성.

인수

- **sl_ip** – 세션 협상 요청을 보낼 다른 데이터 허브의 *SessionListener* gRPC 서버 end point

`SessionRequester._smListener(message, message:event)`

SessionManager 에서 전달되는 스레드 메시지를 수신하는 이벤트 리스너.

인수

- **message** – dictionary(event, message) 구조의 스레드 메시지
- **message:event** – `INIT`, `START_SESSION_CONNECTION`, `GET_NEW_SESSION_INFO`, `UPDATE_INTEREST_LIST`, `UPDATE_NEGOTIATION_OPTIONS`

더 보기:

- *SessionManager._srInit*
- *SessionManager._srStartSessionConnection*
- *SessionManager._srGetNewSessionInfo*

- *SessionManager._srUpdateInterestList*
- *SessionManager._srUpdateNegotiationOptions*

SessionRequester.**_smTransmitNegotiationResult**(*end_point*, *session_desc*, *sn_result*)

SessionRequester 모듈과 다른 데이터 허브의 *SessionListener* 간 세션 협상 체결이 된 경우, *SessionManager* 로 상대 세션 모듈의 end point 정보와 세션 협상 결과를 TRANSMIT_NEGOTIATION_RESULT 이벤트 스레드 메시지와 함께 전달함.

인수

- **end_point** – 다른 데이터 허브의 세션 모듈의 접속 정보(IP, Port)
- **session_desc** – 세션 객체의 메타데이터(세션 생성자 정보, 세션 ID)
- **sn_result** – 세션 협상 결과

더 보기:

- *SessionManager._srListener*

SessionRequester.**_snProcess**(*bucketList*)

SessionManager 로부터 SYNC_ON 이벤트를 전달받은 뒤, bucketList 내 세션 협상 요청을 보낼 데이터 허브 리브스틀 순회하면서, 세션 협상 요청을 처리하는 로직을 수행함.

인수

- **bucketList** – 세션 협상 요청을 보낼 데이터 허브 리스트

더 보기:

- *SessionListener._requestSN*
- *SessionListener._ackSN*

SessionRequester.**run**()

6.3 SessionListener

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

---

class SessionListener()

SessionListener

SessionListener.**_ackSN**(*call*, *callback*)

다른 데이터 허브의 *SessionRequester* 와의 세션 협상 체결이 결정된 이후, *SessionRequester* 로부터 다른 데이터 허브의 연동할 세션 end point 를 전달받는 gRPC 함수.

더 보기:

- *SessionRequester._snProcess*

SessionListener.**_requestSN**(*call*, *callback*)

다른 데이터 허브의 *SessionRequester* 로부터 세션 협상 요청에 대응하는 gRPC 함수로, 다른 데이터 허브의 관심 동기화 수준 및 세션 협상 옵션을 비교하여, 세션 협상 체결 여부를 결정 및 세션 협상 결과를 *SessionRequester* 로 전달함. (세션 협상 체결시, 연동할 세션 end point 도 함께 전달함.)

더 보기:

- *SessionRequester._snProcess*

SessionListener.**_setListenerServer**()

세션 협상 요청을 응답할 gRPC 서버를 구동하는 함수.

SessionListener.**_smListener**(*message*, *message:event*)

SessionManager 에서 전달되는 스레드 메시지를 수신하는 이벤트 리스너.

인수

- **message** – dictionary(event, message) 구조의 스레드 메시지
- **message:event** – INIT, GET_NEW_SESSION_INFO, UPDATE_INTEREST_LIST, UPDATE_NEGOTIATION_OPTIONS

더 보기:

- *SessionManager._slInit*
- *SessionManager._slGetNewSessionInfo*
- *SessionManager._slUpdateInterestList*
- *SessionManager._slUpdateNegotiationOptions*

SessionListener.**_smTransmitNegotiationResult**(*end_point*, *session_desc*, *sn_result*)

SessionListener 모듈과 다른 데이터 허브의 *SessionRequester* 간 세션 협상 체결이 된 경우, *SessionManager* 로 상대 세션 모듈의 end point 정보와 세션 협상 결과를 TRANSMIT_NEGOTIATION_RESULT 이벤트 스레드 메시지와 함께 전달함.

인수

- **end_point** – 다른 데이터 허브의 세션 모듈의 접속 정보(IP, Port)
- **session_desc** – 세션 객체의 메타데이터(세션 생성자 정보, 세션 ID)
- **sn_result** – 세션 협상 결과

더 보기:

- *SessionManager._slListener*

SessionListener.**run**()

SessionListener 실행 함수로 다른 데이터 허브의 *SessionRequester* 로부터의 세션 협상 요청에 응답하는 gRPC 서버를 구동함.

6.4 Session

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

`class Session()`

Session

`Session.Publish(git_patch)`

추출한 git Diff를 상대 session에게 publish하는 함수

인수

- **git_patch** (string()) – git Diff 추출물

`Session.Subscribe(self, call, callback)`

상대 Session으로부터 gRPC를 통해 git Diff 내용을 받아와 적용하고 Kafka로 해당 내용을 외부에 전달하는 함수

인수

- **self** (*Session()*) –
- **call** (dictionary()) – Kafka로 전달받은 내용
- **callback** – Kafka response를 처리하는 콜백 함수

더 보기:

- *Session.gitPatch*
- *Session.kafkaProducer*

`Session.__read_dict()`

JSON으로 저장된 session 내부 변수를 불러오는 함수

`Session.__save_dict(content)`

session 내 변수를 파일로 저장하는 함수

인수

- **content** (dictionary()) –

`Session._init(self)`

gRPC 모듈 활성화 함수

`Session._reset_count(last_commit)`

sync_count를 초기화하는 함수

`Session._smListener(message, message:event)`

SessionManager 에서 전달되는 스레드 메시지를 수신하는 이벤트 리스너.

인수

- **message** (dictionary(event,data)()) – 스레드 메시지
- **message:event** (string()) – INIT, TRANSMIT_NEGOTIATION_RESULT, UPDATE_PUB_ASSET

더 보기:

- *SessionManager._sessionInit*
- *SessionManager._sessionTransmitNegotiationResult*
- *SessionManager._sessionUpdatePubAsset*

Session.**extractGitDiff**(*self*, *topublish*, *topublish:previousLastCommit*, *topublish:commitNumber*)

git Diff를 추출하는 함수

인수

- **self** (*Session()*) –
- **topublish** (dictionary(previousLastCommit, commitNumber)()) – diff 추출에 필요한 정보가 담긴 함수
- **topublish:previousLastCommit** (string()) – 이전 publish 때 사용한 마지막 commit 번호
- **topublish:commitNumber** (Array()) – 이전 publish 이후 들어온 commit 번호의 배열

더 보기:

- *Session.Publish*

Session.**gitPatch**(*git_patch*, *self*)

상대 Session으로부터 전달받은 git Diff를 자신의 gitDB에 적용하는 함수

인수

- **git_patch** (string()) – git Diff 추출물
- **self** (session()) –

반환

- 1인 경우 에러, 0인 경우 정상 동작

Session.**kafkaProducer**(*git_patch*, *self*)

외부에 Kafka 메시지를 발행하는 함수

인수

- **git_patch** (string()) – git Diff 추출물
- **self** (session()) –

Session.**onMaxCount**(*self*)

만약 sync_count에 도달하면 count를 초기화하고 diff를 추출한다

더 보기:

- *Session._reset_count*
- *Session.extractGitDiff*

Session.**prePublish**(*self*, *message*)

git Diff를 Publish하기 전에 내부 변수를 업데이트, 저장하고 publish 조건이 충족되었는 지 확인하는 함수

Session.**run**(*self*)

Session 동작 함수 sync_count 혹은 sync_time 도달 시 Publish하도록 한다

인수

- **self** (*Session()*) –

6.5 SessionNegotiation.proto

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version

3.0.0 of 2022.11.30

6.6 SessionSync.proto

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version

3.0.0 of 2022.11.30

CHAPTER 7

VersionControl

Authors

Eunju Yang (yejyang@kaist.ac.kr)
Ji-Hwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

VersionControl 모듈은 SODAS+ DIS :ref:'dhDaemon'에 의해 실행되는 모듈로, :ref:'vcConsumer'를 통해 수신한 send.asset 토픽의 Kafka 메시지를 처리하고 asset의 버전을 관리하며 이를 통해 asset의 증분 데이터를 추출하는 역할을 수행한다.

VersionControl 모듈은 send.asset 토픽의 Kafka 메시지를 수신하는 *VCConsumer*, 저장소 및 asset 버전을 관리하는 *VersionController*, 그리고 이를 총괄하는 :ref:'vcModule'로 구성된다.

7.1 VCModule

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

class vcModule()

VersionControl 프로세스를 관리하는 모듈

vcModule.commit(*self*, *message*)

vcConsumer 에서 전달된 내용을 기반으로 *versionController* 의 git commit 함수 호출

인수

- **self** (*vcModule()*) – vcModule 객체
- **message** (dictionary()) – *send.asset* 메시지

더 보기:

- *publishVC.commit*

vcModule.editFile(*option*, *filepath*, *content*)

kafka로 전달받은 asset 내용을 파일로 저장/수정/삭제하는 함수

인수

- **option** (string()) – 처리 방법: 저장/수정/삭제
- **filepath** (string()) – 파일 경로
- **content** (string()) – 저장할 내용

vcModule.init()

publishVC 모듈의 초기화 함수를 호출함으로써, pubvc gitDB 의 초기 commit 을 수행함.

vcModule.lockMutex(*self*)

GitDB 사용 시 Mutex를 잠그는 함수

vcModule.reportCommit(*self*, *commitNumber*)

versionController 에서 git commit 후 전달한 commitNumber 를 *SessionManager* 에게 전달

인수

- **self** (*vcModule()*) – vcModule 객체
- **commitNumber** (string()) – git commit 번호

더 보기:

- *SessionManager._vcListener*

vcModule.run()

vcConsumer 모듈의 실행 함수를 호출함으로써, Kafka Consumer 를 구동함.

vcModule.unlockMutex(*self*)

GitDB 사용 완료 후 Mutex 잠금을 해제하는 함수

7.2 VCConsumer

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

`class vcConsumer(`*kafkaHost*, *options*, *VC*`)`
send.referenceModel 과 send.dictionary 를 수신하여 처리하는 Kafka Consumer 객체

인수

- **kafkaHost** (`string()`) – kafka Host 정보
- **options** (`dictionary()`) – options for kafka
- **VC** (*vcModule()*) – vcModule 객체

`vcConsumer.handler(`*message*, *message:topic*, *message:value*, *self*`)`
지정된 메시지 수신 시, 메시지 파싱 후 파일 생성 및 업데이트 `send.asset` 으로 들어오는 모든 이벤트를 vcModule로 파싱하여 전달

인수

- **message** (`dictionary(topic,value)()`) – Kafka 메시지
- **message:topic** (`string()`) – Kafka 토픽 `asset`
- **message:value** (`dictionary()`) – Kafka 내용물
- **self** (*vcConsumer()*) – vcConsumer 객체

더 보기:

- *vcModule.editFile*
- *vcModule.commit*

`vcConsumer.run()`
Kafka Consumer handler 등록을 통한 Kafka Consumer 를 구동하는 함수.

7.3 VersionController

Authors

Eunju Yang (yejyang@kaist.ac.kr)
JiHwan Kim (j.h_kim@kaist.ac.kr)
Jeongwon Lee (korjw1@kaist.ac.kr)

Version
3.0.0 of 2022.11.30

7.3.1 versionController

class VC(*gitDir*, *refRootdir*)

versionController

인수

- **gitDir** (string()) – asset을 저장할 gitDB 의 최상위 경로
- **refRootdir** (string()) – 분류 구조를 참조하기 위한 reference model이 저장된 경로

VC.addReferenceModel(*self*, *ReferenceModel*)

참조할 reference model 목록 추가 함수

인수

- **self** (*VC()*) –
- **ReferenceModel** (Array()) – reference model 목록

VC.returnFirstCommit(*self*, *dir*)

지정된 경로의 gitDB로부터 최초 commit 번호를 추출하는 함수

인수

- **self** (*VC()*) –
- **dir** (string()) – gitDB 경로

반환

commitNumber - 최초 coommit 번호

7.3.2 publishVC

class publishVC(*gitDir*, *refRootdir*)

vcModule 에서 관리하는 gitDB 와 연동된 VC 상속 클래스

인수

- **gitDir** (string()) – asset을 저장할 gitDB 의 최상위 경로
- **refRootdir** (string()) – 분류 구조를 참조하기 위한 reference model이 저장된 경로

publishVC.commit(*filepath*, *message*, *vm*)

gitDB의 변동사항을 git에 add하고 commit한 다음 :ref:'rmSessionManager에 commit 번호를 전달하는 함수

인수

- **filepath** (string()) – 파일 경로
- **message** (string()) – commit message
- **vm** (*vcModule()*) – vcModule 객체

더 보기:

- *vcModule.reportCommit*
- Git.commit

7.3.3 subscribeVC

class subscribeVC(*gitDir*, *refRootdir*)

subscribeVC session에서 관리하는 gitDB와 연동된 VC 상속 클래스

인수

- **gitDir** (**string()**) – asset을 저장할 gitDB 의 최상위 경로
- **refRootdir** (**string()**) – 분류 구조를 참조하기 위한 reference model이 저장된 경로

subscribeVC.apply(*gitPatch*)

상대 session에서 보내온 git Diff 기반 patch 파일을 자신의 gitDB에 적용하는 함수

인수

- **gitPatch** (**string()**) – git Diff 추출물

subscribeVC.commit(*filepath*, *message*)

gitDB의 변동사항을 git에 add하고 commit하는 함수

인수

- **filepath** (**string()**) – 파일 경로
- **message** (**string()**) – commit message

색인

K

P

R

S

V