***KOREN SDI OpenPlatform API***

*Task 3-1 Smart Safety Campus 서비스*

* **Document Information**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Project Name: | KOREN SDI OpenPlatform |  |  |  |
| Project Manager: | 전남대/김경백 |  | Version Number: | V0.1 |
|  |  |  | Version Date: | 2018.10.16 |
| Author: | 전남대/권태용  전남대/염성웅 |  | Written Date: | 2018.10.16 |
| Reviewer: |  |  | Reviewed Date: | 2018.10.16 |

* **Document Distribution**

|  |  |  |  |
| --- | --- | --- | --- |
| Copy Number | Name(Role, Title) | Date | Contact |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Version Date | Author | Description |
| 0.1 | 2018.10.16 | 권태용, 염성웅 | 초안작성 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* **Table of Contents**

[1. 개요 4](#_Toc527580530)

[1.1. 목적 4](#_Toc527580531)

[1.2. Container Control Server 4](#_Toc527580532)

[1.3. Restful API함수 개요 4](#_Toc527580533)

[2. RESTful API 정의서 소개 5](#_Toc527580534)

[3. Container Control API 6](#_Toc527580535)

[3.1. Overview: Container Control API 6](#_Toc527580536)

[3.1.1. Methods 6](#_Toc527580537)

[3.1.2. 리소스 표현 6](#_Toc527580538)

[3.1.3. 속성 7](#_Toc527580539)

[3.1.4. Return Data형식 8](#_Toc527580540)

[3.1.5. 요청 9](#_Toc527580541)

[3.1.6. API 함수 9](#_Toc527580542)

1. 개요
   1. 목적

본 문서의 목적은 Task 3-1 Smart Safety Campus 서비스중 컨테이너 기반의 OpenCCTV Software를 제어하는 Container Control Server의 Restful API 함수의 설명을 위해 작성 되었다

* 1. Container Control Server

Container Control Server는 컨테이너 기반의 OpenCCTV Software를 관리 하기 위한 서버이다. 위 서버는 Restful API 서버를 통하여 사용자가 지정한 OpenCCTV Software의 컨테이너를 생성 제거 관리한다. Kerberos, Zoneminder 와 같은 2가지 OpenCCTV Software를 제공하고 위 프로그램을 통하여 Container Control Server는 코렌망 단말에 연결된 IPCamera의 영상을 HTTP 프로토콜 방식으로 스트리밍 하게 된다. 각각의 소프트웨어는 영상분석 및 스트리밍 가용 카메라 수가 다르고 이 제약 사항은 아래 API 함수 사용 부분에서 설명한다.

* 1. Restful API함수 개요

Container Control API는 Restful 형식의 API이다. 각 함수는 GET, POST, DELET방식을 사용하여 컨테이너의 확인, 생성 및 제거가 가능하다. 현재 API서버의 Authentication은 요구사항이 없다.

1. RESTful API 정의서 소개

본 문서는 Task 3-1, 2개발자에게 필요한 영상 정보의 전달을 위한 HTTP Streaming URL정보를 제공과 컨테이너 추가 제거를 위한 API함수를 설명한다.

RESTful API 정의서에 포함된 함수는 다음과 같은 정보를 가지고 있다.

|  |  |
| --- | --- |
| 항목 | 상세 |
| 데이터 구조 | 사용자 인증 (기본타입) API 에서 사용되는 데이터 구조를 사전 순으로 나열한다. |
| 프로토타입 | 호출 구조에 대한 예제이다. |
| 매개 변수 | 입력되어야 할 항목과 반환되는 항목이다. |
| 설명 | 함수 사용에 대한 설명이다. |
| 반환 값 | 함수의 호출에서 반환되는 다양한 유형, 메시지, 값, 구조 및 설명을 포함하고 있는 항목이다. |
| 설명 | 함수의 사용, 한계 및 영역에 대한 이해를 돕기 위한 특별 설명과 부가정보가 있다. |

1. Container Control API
   1. Overview: Container Control API

위 API는 Container Control Server내에서 실행 되는 컨테이너와 해당 컨테이너가 스트리밍 하는 HTTP Streaming URL을 제공한다. 그리고 컨테이너의 정보는 카메라가 설치된 Obox를 기준으로 저장한다.

* + 1. Methods

Container Control API는 각 Obox, Container, Camera의 메타 데이터를 Json파일에 저장하고 관리한다.

|  |
| --- |
| get  요청 기준과 일치하는 0개 이상의 camera리소스 집합을 반환합니다.  create  카메라 메타 데이터를 업데이트 합니다. 이 메소드는 사용자가 추가한 OpenCCTV Software 도커 컨테이너 정보를 Obox, Container, Camera의 구조로 업데이트 합니다.  delete  사용자가 선택한 카메라의 스트리밍을 중지하고 카메라 메타 데이터 및 컨테이너 메타 데이터를 삭제한다. |

* + 1. 리소스 표현

아래의 JSON 구조는 obox, container, camera리소스의 형식을 보인다.

|  |
| --- |
| {  "**Obox**": [  {  "**name**": "JNU",  "**container**": [  {  "**name**": "containerName",  "**type**": "type",  "**webport**": "80",  "**camera**": [  {  "**name**": "cameraName",  "**url**": "http proxy url"  }  ]  }  ]  }  ]  } |

* + 1. 속성

다음 표는 이 리소스에 표시되는 속성을 정의합니다.

|  |
| --- |
| **Json** |
| **Obox**  **Array**  코렌망에 설치된 Obox들의 데이터를 가지는 배열 값입니다. |
| **Obox name**  **String**  카메라가 연결된 Obox의 위치를 설정한 값이다. Topology를 그리게 될 경우 node를 지정 하기 위한 값이다. |
| **Obox container**  **Array**  해당 obox의 ipcamera를 스트리밍 하는 컨테이너 데이터를 담는 배열 값이다. |
| **Container name**  **String**  해당 컨테이너의 이름을 저장하는 키 값이다. |
| **Container type**  **String**  해당 컨테이너의 실행 이미지 값이다. kerberos와 zoneminer 값 중 하나를 가진다. |
| **Container webport**  **String**  해당 컨테이너가 컨테이너 내부 web server 포트와 바인딩된 포트의 값이다. |
| **Container camera**  **Array**  해당 컨테이너가 스트리밍중인 IPcamera의 배열 값이다. |

* + 1. Return Data형식

API함수의 결과는 JSON형식으로 리턴된다. 이때 JSON 데이터의 형식은 아래와 같다.

{

“success”: 0,

“result”: []

}

|  |
| --- |
| **Return Data 속성** |
| **success**  **int**  함수의 실행 여부를 확인 하는 값이다.  요청한 함수의 성공여부를 1, 0으로 표현하고 1일 경우 성공 0일 경우 실행 오류가 발생한 경우이다. |
| **result**  **Array, String**  각 함수들의 결과 값을 가지는 데이터 이다. 이때 성공 할 경우 해당 결과가 배열 형식으로 저장되고 실패 할 경우 String형식으로 실패 이유를 리턴한다. |

Container Control API: list

* + 1. 요청
* 0.1v의 API는 Authentication을 사용하고 있지 않기 때문에 아래 와 같은 방식으로 HTTP request를 통한 사용이 가능 합니다.

|  |
| --- |
| GET http://api\_server\_ip/method\_path |

* + 1. API 함수
* GET /Data

저장된 JSON파일의 Obox 배열을 리턴 합니다.

* Return Example

{

"success": 1,

"result": [

{

"name": "JNU",

"container": [

{

"name": "containerName",

"type": "type",

"webport": "80",

"camera": [

{

"name": "cameraName",

"url": "httpproxyurl"

}

]

}

]

}

]

}

|  |
| --- |
| **Return Data 속성** |
| success  int  함수의 실행 여부를 확인 하는 값이다.  요청한 함수의 성공여부를 1, 0으로 표현하고 1일 경우 성공 0일 경우 실행 오류가 발생한 경우이다. |
| result  Array  OpenCCTV Control Server에 실행 중인 컨테이너 및 카메라의 정보를 Obox Json 배열 값을 리턴한다. |

* GET /getOboxList

저장된 JSON파일의 Obox 이름을 배열로 리턴 합니다.

* Return Example

{

"success": 1,

"result": [

{

"name": "JNU",

"camera": [

{

"name": "cameraName",

"url": "http proxy url",

"type": "type"

}

]

}

]

}

|  |
| --- |
| **Return Data 속성** |
| success  int  함수의 실행 여부를 확인 하는 값이다.  요청한 함수의 성공여부를 1, 0으로 표현하고 1일 경우 성공 0일 경우 실행 오류가 발생한 경우이다. |
| result  Array  각 obox별 이름과 카메라 베열값을 포함한 json 데이터를 배열로 리턴한다. |
| camera name, url, type  Array (string, string, string)  설치된 카메라 배열 값으로 카메라의 이름, 스트리밍 url, 컨테이너 타입을 리턴한다. |

* GET /getCameraByObox/:oboxName

Path에 oboxName부분에 원하는 obox이름을 입력 하므로써 해당 Obox에 연결된 camera의 데이터를 배열 형식으로 리턴합니다.

* Return Example

{

"success": 1,

"result": [

{

"name": "cameraName",

"url": "http proxy url",

"type": "type"

}

]

}

|  |
| --- |
| **Return Data 속성** |
| **success**  int  함수의 실행 여부를 확인 하는 값이다.  요청한 함수의 성공여부를 1, 0으로 표현하고 1일 경우 성공 0일 경우 실행 오류가 발생한 경우이다. |
| **result**  Array  해당 obox의 설치된 카메라정보를 배열로 리턴한다. |
| **name, url, type**  string, string, string  설치된 카메라 배열 값으로 카메라의 이름, 스트리밍 url, 컨테이너 타입을 리턴한다. |

* GET /getCameraByOboxAndCamera/:oboxName/:cameraName

Path에 oboxName과 cameraName부분에 원하는 obox내의 카메라 이름을 입력 함으로써 해당 Obox내의 camerName을 가지고 있는 카메라의 데이터를 리턴한다.

* Return Example

{

"success": 1,

"result": {

"name": "cameraName",

"url": "http proxy url",

"type": "type"

}

}

|  |
| --- |
| **Return Data 속성** |
| success  int  함수의 실행 여부를 확인 하는 값이다.  요청한 함수의 성공여부를 1, 0으로 표현하고 1일 경우 성공 0일 경우 실행 오류가 발생한 경우이다. |
| result  JSON, String  해당 카메라가 있을 경우 각 카메라의 정보를 json형식으로 리턴하고 해당 카메라가 없을 경우 String으로 “no camera”를 리턴한다. |
| name, url, type  string, string, string  해당 카메라의 이름, 스트리밍 url, 컨테이너 타입을 리턴한다. |

* POST /createContainer

컨테이너 생성을 하고 해당 컨테이너에 ipcamera의 rtspurl을 등록시키는 함수입니다.

|  |
| --- |
| **Parameter 정의** |
| **image**  string  생성할 컨테이너의 이미지값으로 kerberos와 zoneminder를 사용 할 수 있다. |
| **cameras**  Array[string]  사용자가 추가할 카메라의 이름을 배열 형식으로 사용한다. 이때 kerberos는 카메라 이름을 하나만 사용 할 수 있다. |
| **rtsp**  Array[string]  IPcamera의 rtsp url을 적고 모든 rtsp://url/path형식의string 값을 배열 형식으로 보내야 한다. 이때 배열의 크기는 cameras 배열 크기와 같아야 한다. |
| **obox**  string  토폴로지에 그려질 Obox노드의 이름을 사용한다. |

* Return Example

{

"success": 1,

"result": "making container"

}

|  |
| --- |
| **Return Data 속성** |
| **success**  int  함수의 실행 여부를 확인 하는 값이다.  요청한 함수의 성공여부를 1, 0으로 표현하고 1일 경우 성공 0일 경우 실행 오류가 발생한 경우이다. |
| **result**  String  api 함수 실행 결과로 성공적으로 실행시 ‘making container’를 리턴하고  parameter를 잘 못 입력한 경우 “wrong parameter”를 리턴한다. |

* DELET /deleteCamera/:oboxName/:cameraName

스트리밍중인 카메라를 삭제하는 API함수이다. 이때 path에 카메라가 설치된 obox이름과 해당 카메라의 이름을 입력하여 ipcamera영상 스트리밍 중지하고 json파일에서 삭제한다.

|  |
| --- |
| **Parameter 정의** |
| **oboxName**  string  삭제할 카메라가 위치한 obox이름 |
| **cameraName**  string  삭제할 카메라의 이름 |

* -Return Example

{

"success": 1,

"result": “delete camera”

}

|  |
| --- |
| Return Data 속성 |
| success  int  함수의 실행 여부를 확인 하는 값이다.  요청한 함수의 성공여부를 1, 0으로 표현하고 1일 경우 성공 0일 경우 실행 오류가 발생한 경우이다. |
| result  string  카메라를 삭제 한 경우 “delete camera”라는 string 메시지를 리턴한다.  해당 카메라가 없을 경우 ‘there is no camera’라는 string 메시지를 리턴한다. |