**Nexacro platform JSON Adaptor**

**JSON Data Layout 정의서**

1. **JSON Adaptor 정의**

* Nexacro platform의 기본통신 전문 (XML, SSV, Binary)을 사용하지 않고, JSON 전문으로 통신할 수 있도록 구현한 통신 Adaptor

주) 기본통신 전문을 JSON으로 변환하는 과정이 필요하며, 그로 인하여 기본통신 전문에 비하여 성능이 다소 하락 할 수 있습니다.

1. **Nexacro platform data type**

* STRING: 문자열. 최대값은 system에서 한번에 할당 가능한 크기인 2GB로 제한됨

Column의 size와 관계없이 최대치까지 사용될 수 있다.

* INT: 정수형 (-231 ~ 231-1)
* ~~BOOL: Boolean 타입. INT(1) 타입으로 0이면 false, 나머지는 true를 의미한다.~~
* FLOAT, DECIMAL: 실수형 (±2.2X10-308 ~ ±1.7X10308)
* BIGDECIMAL: 실수형(문자열로 저장) (±10-1056 ~ ±10+1056)
* DATE: 날짜형 YYYYMMDD (-8192/01/01 ~ 8191/12/31)
* ~~DATETIME : 날짜와시간 YYYYMMDDHHmmssuuu (msec포함가능)~~
* ~~TIME : 6자리 HHmmssuuu(msec포함가능)~~
* BLOB: 이진 데이터 – base 64 encoding

1. **JSON Layout 정의**

* Client의 수신 JSON String 전문은 JSON.parse() 를 통해 object화 할 수 있어야 합니다.
* Server에 보내는 송신 JSON 전문은 object 를 JSON.stringify() 를 통해 변환한 JSON String 전문 입니다.
  1. 최상위 노드의 구조와 순서는 다음과 같이 구성합니다.

|  |
| --- |
| Header |
| Parameters |
| Datasets |

* 1. Header : 송수신 Header 를 정의 합니다.

Header 노드의 기본 구성은 다음과 같습니다.

|  |  |
| --- | --- |
| “datatype” | “JSON” |
| “codepage” | “utf-8” |

* 1. Parameters : 송수신 parameter를 정의 합니다.

Parameters 노드의 기본 구성은 다음과 같습니다.

|  |
| --- |
| "parameters":  [  ["ErrorCode", "INT", 0], // id , type, value  ["ErrorMsg", "STRING", "JSON 조회 성공"]  ] |

* + - parameters는 Array List 형태로 구성합니다.
    - Parameter type : Array
    - parameter Array 구성

|  |  |  |  |
| --- | --- | --- | --- |
| Index | 1번째 | 2번째 | 3번째 |
| 구성 | ID | Type | Value |
| 예) | ErrorCode | INT | 0 |

* 1. Datasets : 송수신 Dataset을 정의 합니다.

Datasets는 여러개의 Dataset으로 구성되기 때문에 Object Array 형태로 구성합니다.

Datasets 노드의 기본 구성은 다음과 같습니다.

|  |
| --- |
| "datasets":  [  {  "ds\_id":"indata",  "ds\_colinfo":  {  "constcolumn":  [  ["ConstColumn0","10"],  ["ConstColumn1","20"]  ],  "column":  [  ["\_RowType\_","STRING","1"],  ["Column0","STRING","256"],  ["Column1","STRING","256"],  ["Column2","STRING","256"]  ]  },  "ds\_rows":  {  "row":  [  ["U",null,"zzzzzzzzzzzzz",""],  ["N","k","r","g"]  ],  "orgrow":  [  ["O",null,"ppppp",""],  []  ]  }  },  {  "ds\_id":"indata2",  "ds\_colinfo":  {  "constcolumn":  [],  "column":  [  ["\_RowType\_","STRING","1"],  ["col1","STRING","256"],  ["col2","STRING","256"]  ]  },  "ds\_rows":  {  "row":  [  ["N","test1","뷁"]  ],  "orgrow":  [  []  ]  }  }  ] |

* + - 구성

|  |  |  |  |
| --- | --- | --- | --- |
| **datasets** | Dataset Object Array | | |
| **ds\_id** | String Value | |
| **ds\_colinfo** | **constcolumn** | |
|  | Array list |
| **column** | |
|  | Array list |
| **ds\_rows** | **row** | |
|  | Array list |
| **orgrow** | |
|  | Array list |

* Bold 체는 예약어임.
  + - constcolumn 구성

|  |  |  |
| --- | --- | --- |
| Index | 1번째 | 2번째 |
| 구성 | id | Value |
| 예) | ConstColumn0 | 10 |

* + - column 구성

|  |  |  |  |
| --- | --- | --- | --- |
| Index | 1번째 | 2번째 | 3번째 |
| 구성 | Column id | type | size |
| 예) | Column0 | STRING | 256 |

* row의 첫 번째 값은 해당 레코드의 상태를 표시하기 때문에 column 정보 구성 시

첫 번째는 column은 “\_RowType\_” 으로 지정합니다.

* + - row 구성

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | 1번째 | 2번째 | 3번째 | …n |
| 구성 | 상태 값 | Value | value | …n |
| 예) | N | 값1 | 값2 | …n |

* column으로 구성된 데이터를 순서대로 array로 담아 전달

Array의 첫 번째 값은 해당 recode의 상태를 표시하는 값으로 지정됩니다.

**상태 표시 값 : N(변경 없음), I(insert recode), U(update recode), D(delete recode)**

**server 에서 client 로 전송 시 상태 값은 “N”으로 고정합니다.**

* + - orgrow 구성
* column으로 구성된 데이터를 순서대로 array로 담아 전달

Array의 첫 번째 값은 해당 recode의 상태를 표시하는 값으로 지정됩니다.

row Array의 상태 값이 U인 recode의 변경 전 값이 담깁니다.

**상태 표시 값 : O(변경 전 값)**

* **client 에서 server 로 전송 시 사용, server 에서 client 로 전송 시 생략 (adaptor에서 사용하지 않음)**

1. **송수신 샘플 전문**
   1. 송신 (client -> server)

|  |
| --- |
| {  "datatype":"JSON",  "codepage":"utf-8",  "parameters":  [  ["g\_val2","STRING","globalval"],  ["Variable0","STRING",""],  ["test","STRING","123"],  ["aaa","STRING","bbb"]  ],  "datasets":  [  {  "ds\_id":"indata",  "ds\_colinfo":  {  "constcolumn":  [  ["ConstColumn0","10"],  ["ConstColumn1","20"]  ],  "column":  [  ["\_RowType\_","STRING","1"],  ["Column0","STRING","256"],  ["Column1","STRING","256"],  ["Column2","STRING","256"]  ]  },  "ds\_rows":  {  "row":  [  ["U",null,"zzzzzzzzzzzzz",""],  ["N","k","r","g"],  ["D","a","b","c"],  ["I",null,null,null]  ],  "orgrow":  [  ["O",null,"ppppp",""],  [],  [],  []  ]  }  },  {  "ds\_id":"indata2",  "ds\_colinfo":  {  "constcolumn":  [],  "column":  [  ["\_RowType\_","STRING","1"],  ["col1","STRING","256"],  ["col2","STRING","256"]  ]  },  "ds\_rows":  {  "row":  [  ["N","test1","뷁"],  ["N","가나","다라"]  ],  "orgrow":  [  [],  []  ]  }  }  ]  } |

* 1. 수신 (server -> client)

|  |
| --- |
| {  "datatype":"JSON",  "codepage":"utf-8",  "parameters":  [  ["g\_val2","STRING","globalval"],  ["Variable0","STRING",""],  ["test","STRING","123"],  ["aaa","STRING","bbb"]  ],  "datasets":  [  {  "ds\_id":"indata",  "ds\_colinfo":  {  "constcolumn":  [  ["ConstColumn0","10"],  ["ConstColumn1","20"]  ],  "column":  [  ["\_RowType\_","STRING","1"],  ["Column0","STRING","256"],  ["Column1","STRING","256"],  ["Column2","STRING","256"]  ]  },  "ds\_rows":  {  "row":  [  ["U",null,"zzzzzzzzzzzzz",""],  ["N","k","r","g"],  ["D","a","b","c"],  ["I",null,null,null]  ]  }  },  {  "ds\_id":"indata2",  "ds\_colinfo":  {  "constcolumn":  [],  "column":  [  ["\_RowType\_","STRING","1"],  ["col1","STRING","256"],  ["col2","STRING","256"]  ]  },  "ds\_rows":  {  "row":  [  ["N","test1","뷁"],  ["N","가나","다라"]  ]  }  }  ]  } |