내용

[사전 구조 1](#_Toc204328575)

[사전 추가 방법 (DEVICE.json) 4](#_Toc204328571)

[사전 추가 방법 (ACTION.json) 7](#_Toc204328572)

[동사 생성 규칙 (Verb 확장 기준) 9](#_Toc204328573)

[사전 추가 방법 (ATTRIBUTE.json) 14](#_Toc204328574)

**사전구조**

**ACTION.json**

스마트홈 디바이스가 수행할 수 있는 **행위(Action)** 를 정의한 JSON 파일입니다.  
이 파일은 문장 합성 시 **동사**로 활용될 표현 목록을 제공합니다.

각 행위는 다음과 같은 범주로 분류됩니다:

* **시작/실행**
* **정지/중단**
* **재개/계속**
* **변경/전환**
* **설정/조정**
* **정보 요청**

각 행위는 아래와 같은 세 가지 필드를 포함한 구조로 정의되어 있습니다:

* **동사(Verb)**:  
  사용자가 명령 시 사용할 수 있는 주요 **동사 표현** 목록입니다.
* **대상(Tag)**:  
  해당 행위가 적용될 수 있는 **디바이스 또는 장치의 범주**입니다.
* **목적어 형식(Particle)**:  
  명령 문장에서 **목적어**로 사용될 요소의 형식 또는 구조를 정의합니다.  
  일반적으로 중첩 배열 형태로 표현됩니다.

**ATTRIBUTE.json**

스마트홈 디바이스 제어에 사용되는 속성(attribute) 정보를 정의한 JSON 파일입니다.  
이 파일은 문장 합성 시 **목적어**로 활용될 표현 목록을 제공합니다.

각 속성은 다음과 같은 구조로 구성됩니다:

* **타입(type)**: 속성의 데이터 유형 (Numerical 또는 Categorical)
* **옵션(option)**: 해당 속성에 대해 사용자가 입력할 수 있는 **목적어** 또는 **부사 표현**의 목록

**PARTICLE.json**

**ACTION.json**의 Particle 항목에 정의된 **목적어 형태(문법적 요소)** 를 구체화하기 위해 사용되는 JSON 파일입니다.

이 파일은 각 **의미 범주**에 해당하는 **조사/형태소 표현** 목록을 제공하며, 명령 문장에서 목적어의 위치나 의미를 명확히 하기 위한 **후치어(조사)** 를 정의합니다.

각 항목은 다음과 같은 구조로 이루어져 있습니다:

* **키(Key)**: 의미 역할 또는 개념명
* **값(Value)**: 해당 역할에 대응되는 **조사 표현 목록**

**DEVICE.json**

스마트홈 디바이스별로 수행 가능한 **행위(Action)** 를 정의하고, 각 행위에 대한 **문장 구성 구조**를 명시하는 JSON 파일입니다.  
이 파일은 문장 합성 과정에서 **명령을 완성**하는 최종 단계로 사용되며, 각 디바이스별로 어떤 동작을 어떤 형식으로 수행할 수 있는지를 결정합니다.

**구성 방식**

각 디바이스는 다음과 같은 계층 구조로 정의됩니다:

* **디바이스 이름 (예: "공기청정기")**
  + **행위 범주 (예: "시작/실행", "정보 요청")**
    - **의도(ACTION) 키워드 (예: "TurnOnPower", "GetInformation")**
      * **문장 구성 구조 (예: [[["", "PowerType"]]])**

**필드 간 연결 관계**

* **의도 키워드 (예: "GetInformation")**:  
  → ACTION.json에서 정의된 키
* **슬롯 값 (예: "AirPurifierInfo", "PowerType")**:  
  → ATTRIBUTE.json에 정의된 항목
* **슬롯 배열**:  
  → 하나의 명령 문장에서 사용될 **속성(attribute) 간의 순서와 결합 방식**을 지정

**사전 추가 방법 (DEVICE.json)**

**1. Device 선정**

추가하고자 하는 디바이스를 결정합니다. ex) 에어로타워

**2. 기본 구조 설정**

다음 두 가지 방법 중 하나를 선택합니다:

**① 템플릿 복사**: 기존에 존재하는 가장 유사한 디바이스(예: "공기청정기") 구조를 복사합니다.

**② 수동 정의**: 디바이스가 수행할 수 있는 행동 목록을 ACTION 범주에 따라 직접 작성합니다.

**3. 불필요한 기능 제거**

복사해온 구조에서 해당 디바이스가 지원하지 않는 기능은 반드시 삭제합니다.

**4. Tag 기반 ACTION 선정**

ACTION.json의 Tag 항목을 확인하여 해당 디바이스가 어떤 행위(intent) 를 가질 수 있는지 판단합니다.

필요한 경우 새로운 ACTION을 정의하거나, 유사한 디바이스의 ACTION 구조를 참조하여 재사용합니다. → 참고: ACTION.json 작성 방법

**5. ATTRIBUTE 선정**

각 ACTION에서 요구하는 **슬롯 값(= 목적어)** 에 해당하는 **속성(attribute)** 을 ATTRIBUTE.json에서 선택합니다

이때, 해당 Attribute가 ATTRIBUTE.json에 **이미 존재하는지** 또는 **새롭게 정의해야 하는지** 확인합니다. → 참고:ATTRIBUTE.json 작성 방법

**6. 문법 구조 배열 작성**

DEVICE.json에서 각 ACTION은 **문장 합성을 위한 슬롯 배열 구조**로 정의됩니다. 이 구조는 **속성(Attribute)** 들의 **순서와 종속 관계**를 표현하는 역할을 합니다.

**6.1. 문법 구조 작성 방법**

기본구조

[

[attribute 의 종속적 순서 1 ],

[ attribute 의 종속적 순서 2],

… ,

]

각 순서 배열은 ACTION.json의 "Particle"에 정의된 순서와 **1:1 대응**합니다.

각 슬롯에는 **단일 attribute** 또는 **attribute 배열**이 들어올 수 있으며, 배열 내 조합 방식에 따라 문장이 생성됩니다.

Ex) "Particle": [["Object", "Value"]]인 경우

[

[["Attribute1"], ["Attribute2", "Attribute3"]],

[["Attribute4"] , “” ]

]

첫 번째 슬롯 그룹은, "**Object**"에 해당하는 "Attribute1" 와 "**Value**"에 해당하는"Attribute2", "Attribute3" 의 조합으로

"Attribute1" + "Attribute2"

"Attribute1" + "Attribute3" 조합이 생기고,

두 번째 슬롯 그룹은 , "Object"에 해당하는 " Attribute4" 와 "Value"에 해당하는

"” 의 조합으로, 두번째 Value를 생략 가능합니다.

**작동 방식**

1. **슬롯은 순차적으로 해석됨**

Ex) "SetSound": [["SoundType", ["SoundLevel", "SoundMode"]]]

- "SoundType" 뒤에 반드시 "SoundLevel"과 "SoundMode"가 함께 붙은 구조 생성됨

- “SoundType"이 선행 조건인 **부속 구조**임

2. **각 슬롯 안에서는 조합 가능**

Ex) "TurnOnPower": [[["", "PowerType"]]]

- "PowerType"이 붙는 구조 → 예: "전원 켜줘"

- "PowerType"이 없어도 문장이 가능한 경우, 이처럼 ""만 남겨두면 옵션형 구조 생성 가능

3. **다중 조합 생성 방식**

Ex) [["AirConditionerMode", "PowerType"]]

- 이 구조는 "AirConditionerMode" 와 "PowerType" 각각에 대해 **독립적으로 문장이 생성됨**

- 예: "냉방 모드로 켜", "전원을 켜줘" 등

**4. 다중 중첩 조합 생성 방식**

Ex ) [ [ ["SoundType", “SoundLocation”], ["SoundLevel", "SoundMode"] ] ]

- 첫 슬롯 "SoundType"과 "SoundLocation" → 2개

- 두 번째 슬롯 "SoundLevel", "SoundMode" → 2개

- **조합 개수: 2×2 = 4개 문장 생성 가능**

**사전 추가 방법 (ACTION.json)**

커스텀 동작이 필요하거나 기존에 없는 **새로운 동작(Action)** 을 정의해야 할 경우, 다음 절차에 따라 ACTION.json에 항목을 추가합니다.

**1. 동작의 의미 및 범주 결정**

새롭게 정의할 동작이 어떤 범주에 속하는지 판단합니다.  
분류 기준은 다음 중 하나로 선택합니다:

* 시작/실행
* 정지/중단
* 재개/계속
* 변경/전환
* 설정/조정
* 정보 요청

예: "LED를 켜다" → **시작/실행**, "데이터 동기화" → **설정/조정**

**2. ACTION 키 작성**

해당 범주의 키 하위에 새로운 동작 Action 이름을 작성합니다.

"TurnOnLED": {

"Particle": [[]],

"Verb": [],

"Tag": []

}

**3. Particle 필드 작성**

각 Action에서 사용할 수 있는 **문장 구성 조합(slot)** 을 Particle 필드에 정의합니다.

PARTICLE.json 기준으로 다음과 같은 항목이 존재합니다:

{

    "None" : [""],

    "Object" : ["", "을"],

    "Value" : ["", "으로"],

    "ScheduleTime" : ["뒤에","뒤","있다","있다가"],

    "Duration" : ["","동안","간"],

    "Of" : ["","의"],

    "Count" : ["","만", "정도"]

}

**Particle 필드는** 위 항목 중에서 **해당 동작에 필요한 slot 역할을 조합**해 결정합니다.

여러 역할을 함께 사용할 수 있으며, 필요한 만큼 늘릴 수 있습니다.

Ex) "Particle": [["Object", "Value"]],

"온풍기 세기 높여줘", "온풍기 약하게 해줘" 등과 같은 조합을 허용

**4. Verb 필드 작성**

해당 동작과 관련된 **모든 동사 표현을 가능한 한 많이 작성**합니다.

* 중복, 띄어쓰기, 구어체 형태 모두 포함해도 됩니다.
* 후처리 시스템이 자동으로 정제 및 전처리를 수행합니다.

**5. Tag 필드 작성**

이 동작이 **적용 가능한 디바이스 종류**를 Tag 필드에 명시합니다.

* "공기청정기", "스마트글래스", "뷰티디바이스" 등 실제 DEVICE.json에 정의된 디바이스 이름을 작성하여 후에 추가가 쉽게합니다.

**동사 생성 규칙 (Verb 확장 기준)**

ACTION.json의 "Verb" 필드를 작성할 때 참고하면 좋은 **동사 확장 규칙**입니다.  
이 규칙은 사용자의 다양한 발화를 모두 포괄할 수 있도록 설계되었으며,  
**명령문, 청유문, 높임/낮춤, 격식/비격식** 등의 조합을 통해 다양한 형태로 자동 생성됩니다.

**1. 형태소 분석 기반 분기**

입력된 동사가 다음 중 어느 형태인지 판단합니다:

| **유형** | **설명** | **예시** |
| --- | --- | --- |
| **명사 기반 파생 동사** | 명사 + 연결 어미로 구성된 표현 | "건조시켜줘","건조해줘", “빨래해줘” |
| **동사 원형 기반 동사** | 고유한 동사 어간으로 구성된 표현 | "켜줘", “꺼줄래”, “틀어봐” |

**2.1 명사형 기반 동사 파생**

명사형 어절에서 어간을 추출하고, 다음 접미사 규칙을 적용해 원형 동사로 정규화합니다:

| **규칙** | **예시** |
| --- | --- |
| 명사형 + -하- | "건조" → "건조하" |
| 명사형 + -시키- | "건조" → "건조시키" |

정규화된 명사형 동사는 이후 **2.2 동사형 입력 처리 방식**와 동일한 과정으로  
**연결 어미(-아/-어) 적용 → 보조 동사 결합 → 문장 확장** 과정으로 이어집니다.

**2.2 동사형 입력 시 처리 방식**

동사 원형이 입력된 경우, 어간을 분리하여 모음 조화에 따라 연결 어미를 적용합니다:

| **입력** | **연결 어미** | **결과** |
| --- | --- | --- |
| 켜다 | -아- | 켜 + 아 → 켜 |
| 끄다 | -어- | 끄 + 어 → 꺼 |

**3. ‘도록’ / ‘게’ 확장을 통한 어근 생성**

원형 동사나 파생 동사(예: 건조시키다, 켜다)에 의도/목적/방향 표현을 추가하여, 문장 확장의 중간 단계로 사용되는 ‘도록’, ‘게’ 형태를 생성합니다.

| **형태** | **의미 용법** | **생성 방식 예시** |
| --- | --- | --- |
| -도록 하- | “~하게 하다” (목적 + 동작 유도) | 켜다 → 켜도록 하다 → 켜도록 해줘 |
| -게 하- | “~하게 하다” (허용/요구) | 켜다 → 켜게 → 켜게 해줘 |

**4. 보조 동사 결합을 통한 활용형 생성**

원형 동사에 다음 **보조 동사 5종**을 결합하여, 다양한 문장 형태를 생성합니다:  
주요 보조동사 5종: **주다**, **보다**, **놓다**, **두다, 버리다**

Ex) 건조시키, 켜 기준:

| **원형** | **연결 어미** | **보조동사** | **예시** |
| --- | --- | --- | --- |
| 건조시키 | -아/-어 | 주다 | 건조시켜주 |
| 건조시키 | -아/-어 | 보다 | 건조시켜보 |
| 건조시키 | -아/-어 | 놓다 | 건조시켜노 |
| 건조시키 | -아/-어 | 두다 | 건조시켜두 |
| 건조시키 | -아/-어 | 버리다 | 건조시켜버리 |
| 켜 | -아 | 주다 | 켜주 |
| 켜 | -아 | 보다 | 켜보 |
| 켜 | -아 | 놓다 | 켜노 |
| 켜 | -아 | 두다 | 켜두 |
| 켜 | -아 | 버리다 | 켜버리 |

**5. 2단 보조 동사 결합**

동사 어간에 보조 동사 2개를 연속으로 결합하여 더 복잡하고 자연스러운 명령/요청 표현을 생성합니다. 예: 켜 + 놓다 + 주다 → 켜놔줘

| 1단계 보조동사 | 2단계 보조동사 | 예시 어간 | 결합 예시 |
| --- | --- | --- | --- |
| 놓다 | 주다 | 켜 | 켜놓아주다 → 켜놔줘 |
| 놓다 | 보다 | 켜 | 켜놓아보다 → 켜놔봐 |
| 두다 | 보다 | 저장하다 | 저장해두어보다 → 저장해둬봐 |
| 주다 | 보다 | 해 | 해주어보다 → 해줘봐 |
| 버리다 | 주다 | 해 | 해버려주다 → 해버려줘 |

**6. 보조 동사 결합된 어간 + 기본 어간을 통한 예문 생성**

앞서 생성된 보조 동사 결합 어간(켜주, 건조시켜보 등)을 바탕으로, 다양한 문장 형식(청유형/명령형 등) 과 높임/낮춤, 격식/비격식의 조합에 따라 실제 문장 형태를 생성합니다.

| **문장 형식** | **설명** | **예시 (켜주)** |
| --- | --- | --- |
| 명령형 | 동작을 직접 요구 | 켜줘, 켜주세요, 켜주십시오 |
| 청유형 | 함께 하자고 제안 | 켜주자, 켜줍시다 |
| 평서형 | 상태나 동작을 진술 | 켜줘, 켜줘요 |
| 관형사형 | 미래 동작 의지/요청 표현 | 켜줄래 |

**선택 가능한 조합**

| **단계** | **분류** | **선택지** | **선택 수** |
| --- | --- | --- | --- |
| 1 | **어간 유형 선택** | - 명사 + 하  - 명사 + 시키  - 명사 + 하, 시키 둘 다 | 3 |
| 2 | **의도/목적 확장** | - 없음  - 도록  - 게  - 도록, 게 둘 다 | 4 |
| 3 | **1단 보조동사 선택** | - 없음  - 주다  - 보다  - 놓다  - 두다  - 버리다 | 6 (5 조합 + 없음) |
| 4 | **2단 보조동사 조합** | - 없음  - 놓다 → 주다  - 놓다 → 보다  - 두다 → 보다  - 주다 → 보다  - 버리다 → 주다 | 6 (5 조합 + 없음) |
| 5 | **문장 형식** | - 평서문  - 명령문  - 청유문  - 관형사형 | 4 |
| 6 | **높임/말투** | - 낮춤말  - 높임말 | 2 |

**Verb 입력 Tip**

**1. 구어체 그대로 입력해도 됩니다**

* **입력 예시**: "빨래 건조해줘", "불 켜봐", "꺼줄래"
* 시스템은 입력된 문장을 형태소 분석하여 **핵심 동사 표현만 추출**해 처리합니다.
* 예를 들어 "빨래 건조해줘"가 입력되면, 동사를 찾아,   
  "건조"를 중심으로 → "건조시키다" → "건조시켜줘", "건조시켜봐" 등으로 확장 후 빨래가 후에 건조 앞에 붙는 형식입니다.

**2. 띄어쓰기나 조사가 있어도 자동 정규화됩니다**

* "빨래를 건조해줘" → "빨래"는 객체로 인식, "건조"는 동사화
* "불 좀 켜줘" → "불"은 목적어, "켜줘"는 동사로 인식됨
* 따라서 **"켜봐"**, **"불 켜줘"**, **"불을 켜줘"** 모두 동사 활용형이 나오고, 앞에 목적어가 추가된 형태가 나옵니다.

Ex)

| **입력 문장** | **처리 방식 예시** |
| --- | --- |
| 빨래 건조해줘 | 건조 → 건조시키 → 건조시켜줘, 건조시켜봐 등 + "빨래" 결합 |
| 불 켜봐 | 켜 → 켜봐, 켜줘, 켜두자 등 + "불" 결합 |
| 전원을 꺼줄래요 | 끄 → 꺼주다 → 꺼줘, 꺼주세요 등 + "전원" 결합 |

**사전 추가 방법 (ATTRIBUTE.json)**

ATTRIBUTE.json은 문장에서 속성(=목적어)을 정의하는 JSON 사전입니다.  
속성은 기기의 상태를 조정하거나 정보를 조회하는 데 사용되는 요소로, 주로 명령문 내 목적어 자리에 등장합니다.

속성은 **수치 기반(Numerical)** 또는 **범주 기반(Categorical)** 중 하나로 분류되며, 이에 따라 "type"과 "option" 형식이 달라집니다.

**1. 속성명(Key) 정의**

각 항목의 최상단 키는 해당 속성의 고유 이름입니다. 속성은 **의미가 명확한 영문 CamelCase** 형태로 작성합니다.

예: BrightnessLevel, TimeSecond, LEDType, Color, Temperature

**2. type 지정**

속성이 **수치 정보**를 표현한다면 "Numerical", **고정된 값 중 하나**를 고른다면 "Categorical"로 지정합니다.

| **조건** | **type 값** | **예시 속성** |
| --- | --- | --- |
| 범주형 값 (모드, 색, 종류 등) | "Categorical" | Color, LEDType, PowerType |
| 수치 범위 값 (시간, 단계 등) | "Numerical" | TimeSecond, Temperature, BrightnessLevel |

**3.** **"type": "Categorical" 작성법**

사용자가 말할 수 있는 고정된 **자연어 표현 목록**을 "option" 배열로 정의합니다. 띄어쓰기, 동의어, 구어체, 음역어 등 다양한 표현을 포함하는 것이 좋습니다.

"속성명": {

"type": "Categorical",

"option": ["표현1", "표현2", ...]

}

| **필드** | **설명** |
| --- | --- |
| type | 반드시 "Categorical"로 고정 |
| option | 사용자가 말할 수 있는 모든 명사(자연어) 배열 |

Ex)

"Color": {

"type": "Categorical",

"option": ["빨간색", "파란색", "초록색", "노란색", "푸른색"]

}

**4. "type": "Numerical" 작성법**

**Numerical 타입은 수치 기반의 속성**을 정의할 때 사용됩니다.

“3단계로 설정해줘”, “20도 맞춰줘”, “10초 뒤에 꺼줘”와 같이 **숫자 + 단위**로 표현되는 목적어(속성)를 처리합니다.

Ex)

"속성명": {

"type": "Numerical",

"option": {

"min": 정수, "max": 정수, "step": 정수, "unit": ["단위1", ...],

"numeralType": "Cardinal" 또는 "Ordinal"

} }

| **필드** | **설명** |
| --- | --- |
| min | 최소값 (정수) |
| max | 최대값 (정수) |
| step | 값 간격 단위 |
| unit | 허용 단위 리스트 (예: "초", "도", "단계") |
| numeralType | "Cardinal" (기수: 한, 둘, 셋… ) 또는 "Ordinal" (서수:일, 이, 삼 .. ) |

**단위가 말로 생략될 수 있는 경우**: 반드시 unit에 ""를 포함하세요.  
예: "밝기 22퍼센트로" 같은 표현도 "밝기 22퍼"로 처리 가능해짐.

**numeralType에 따라 생성되는 언어에 직접 영향을 미칩니다.**  
"한 단계" 같은 표현이 필요한 경우 " Cardinal " 필수.

min, max, step 값은 실제 디바이스 기능 또는 비즈니스 로직 기준에 맞춰 설정합니다.