Se utilizará el IDE NetBeans para la creación de los proyectos y ejecutables (https://netbeans.org/downloads/). IMPORTANTE: el JDK usado es el 1.8.

1. Descargamos como ZIP el repositorio mithra de Github y lo descomprimimos.
    - Las carpetas ConfigFile, DiskLogger, MyAgent, onMessage, SubscribedAgent, DevicesCID y ServerCID de la carpeta descargada, son las de los proyectos que tendremos que crear, y contienen tanto los nombres de los paquetes del proyecto, como las respectivas clases que tiene cada paquete.
    - Así, por ejemplo, Como la carpeta DevicesCID, dentro de su carpeta src, contiene OccurrencesSearch, Occurrences, IPLogger y deviceAgents, significa que el proyecto DevicesCID tendrá los paquetes IccurrencesSearch, Occurrences, IPLogger y deviceAgents.
    - De la misma forma, por ejemplo, la carpeta Occurrences contiene dos archivos: OccurrencesList.java y OccurrencesCounter.java. Esto significa que el paquete Occurrences del proyecto DevicesCID contendrá las clases OccurrencesList y OccurrencesCounter.
2. Creamos primero todos los proyectos. Para ello:
    - Creamos nuevos proyectos (File->New Project...) de tipo "Java application".
    - En la pestaña "Projects" (a mí me sale a la izquierda en la aplicación), nos encontraremos los proyectos creados.
    - Eliminamos el paquete que crea por defecto en cada proyecto (despliegas con doble click el nombre del proyecto, doble click en "Source Packages", click derecho en el único que se despliega, delete y aceptar).
3. Estructuramos los proyectos como están en cada apartado del ZIP descompreso (indicado en el apartado 1):
    - Para crear paquetes, click derecho en "Source Packages" de cada proyecto, New->Java Package, y le pones el nombre del paquete.
    - Para añadir las clases a cada paquete, desde el explorador de archivos arrastras el archivo hacia NetBeans justo encima del paquete que te interese y lo sueltas (CUIDADO: si no se pone

justamente encima del paquete te crea un paquete por defecto donde mete el archivo).

4. Configuramos los proyectos secundarios (todos excepto DevicesCID y ServerCID):
    - Click derecho encima del nombre del proyecto, Properties
    - En el apartado Build->Packaging, deben estar activos "Compress JAR File" y "Build JAR after Compiling", y desactivado "Copy Dependent Libraries".
    - En el apartado Libraries vamos a añadir las dependencias de cada proyecto presionando el botón "Add JAR/Folder" (CONSEJO: en la ventana que se nos abre, al elegir el .jar, seleccionar "Relative Path"). Las dependencias son las siguientes:

      (NOTA: los archivos magentix2-2.01-jar-with-dependencies.zip, commons-lang3-3.6.jar, commons-io-2.5.jar y com.eclipsesource.json.jar se encuentran en la carpeta "ExternalLibraries" del repositorio)

        - ConfigFile: com.ecplipsesource.json.jar magentix2-2.01-jar-with-dependencies.zip
        - DiskLogger: nada
        - MyAgent: magentix2-2.01-jar-with-dependencies.zip
        - onMessage: magentix2-2.01-jar-with-dependencies.zip
        - SubscribedAgent: magentix2-2.01-jar-with-dependencies.zip
5. Creamos los .jar:
    - Click derecho en el nombre del proyecto, Clean and Build.
    - El archivo se encontrará en la carpeta del proyecto, en la carpeta "dist".
6. Configuramos ahora los proyectos DevicesCID y ServerCID:
    - Click derecho encima del nombre del proyecto, Properties

- En el apartado Build->Packaging, deben estar activos "Copy Dependent Libraries" y "Build JAR after Compiling", y desactivado "Compress JAR File".
- En el apartado Libraries vamos a añadir las dependencias de cada proyecto presionando el botón "Add JAR/Folder" (CONSEJO: en la ventana que se nos abre, al elegir el .jar, seleccionar "Relative Path"). Las dependencias son las siguientes:
  - DevicesCID: com.eclipsesource.json.jar magentix2-2.01-jar-with-dependencies.zip ConfigFile.jar DiskLogger.jar SubscribedAgent.jar MyAgent.jar onMessage.jar commons-lang3-3.6.jar commons-io-2.5.jar
  - ServerCID: SubscribedAgent.jar com.eclipsesource.json.jar magentix2-2.01-jar-with-dependencies.zip DiskLogger.jar MyAgent.jar ConfigFile.jar onMessage.jar

7. Creamos los -jar (igual que en el apartado 5).
8. Ejecutar el proyecto.
   - Se puede tanto ejecutar desde NetBeans, como irse a la carpeta "dist" del proyecto y ejecutar "java -jar aplicacion.jar".

NOTAS: - Se necesita de los archivos "config.json" en los proyectos para poder ejecutar DevicesCID y ServerCID (hay ejemplos de ambos archivos en las carpetas de los proyectos en los repositorios llamados "myconfig.json"). - El programa DevicesCID necesita que primero esté ejecutándose ServerCID para que el agente SSH se pueda subscribir al agente SERV.