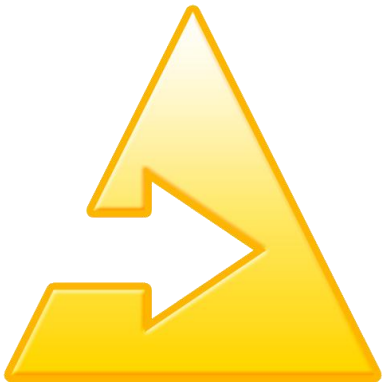
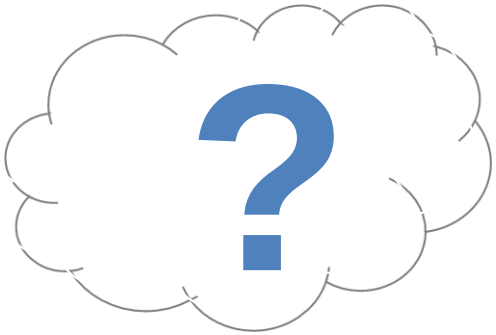
ImageJ2 Integration (Beta)



KNIME Nodes



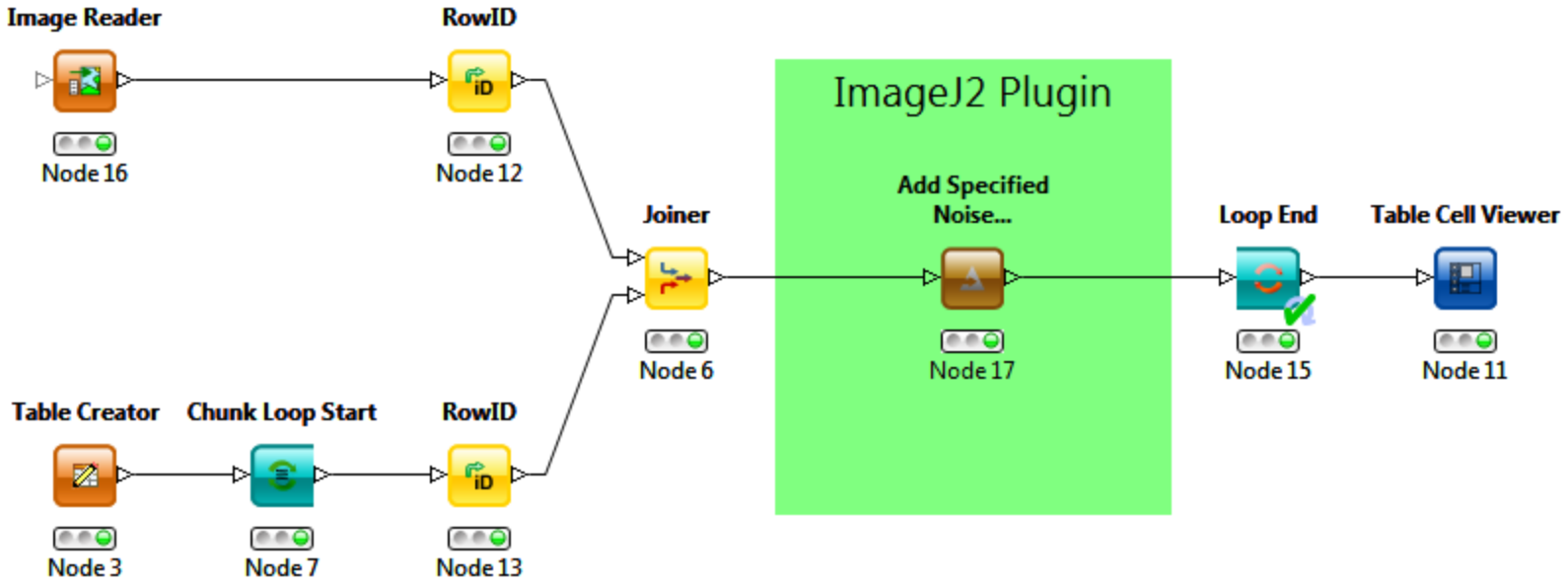
ImageJ2 Commands

Automatic Node Generation from ImageJ2 Plugins

- Requirements
  - Command interface implemented
  - tagged as headless
  - no user interaction

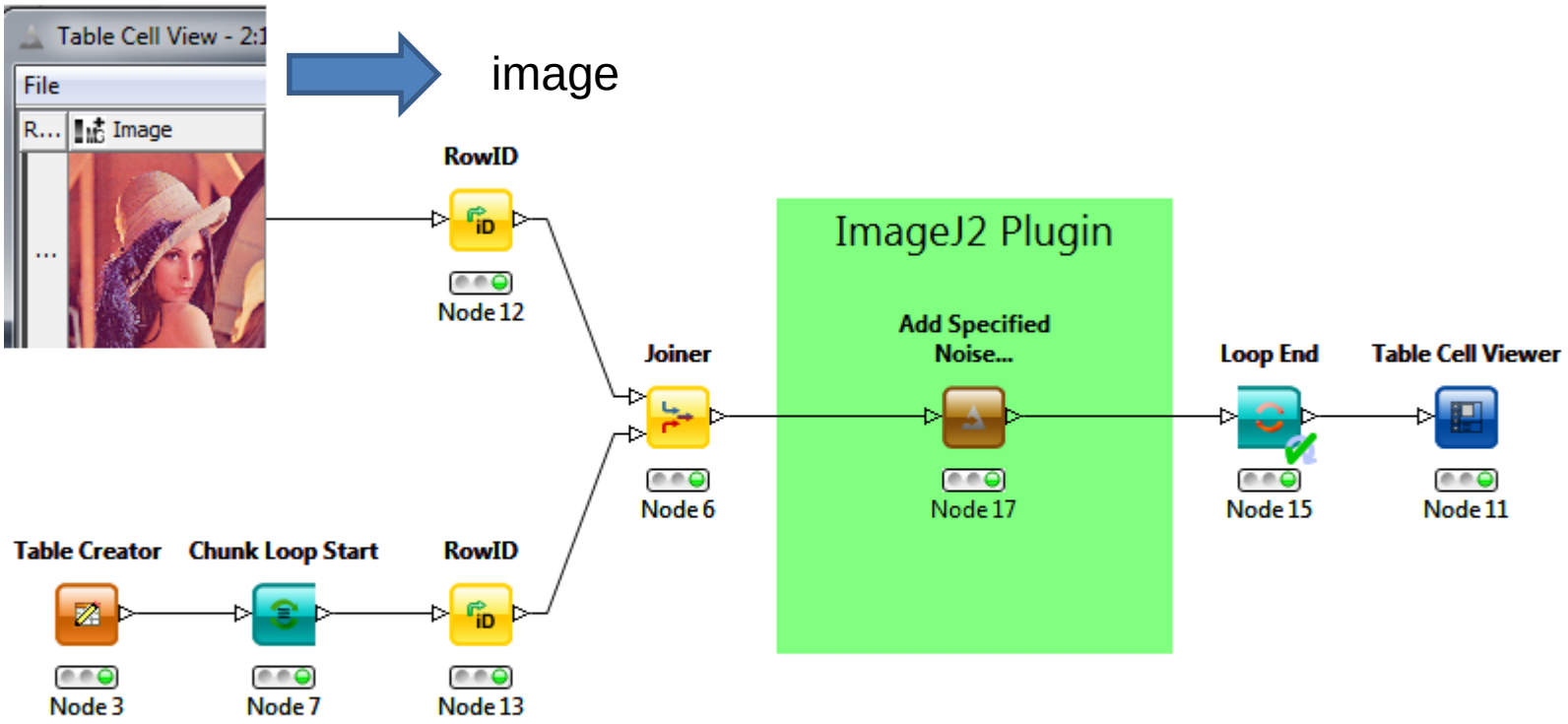
- Result
  - automatic discovery by ImageJ2 services
  - working ImageJ2 plugin
  - converted to fully flagged KNIME node

Example workflow



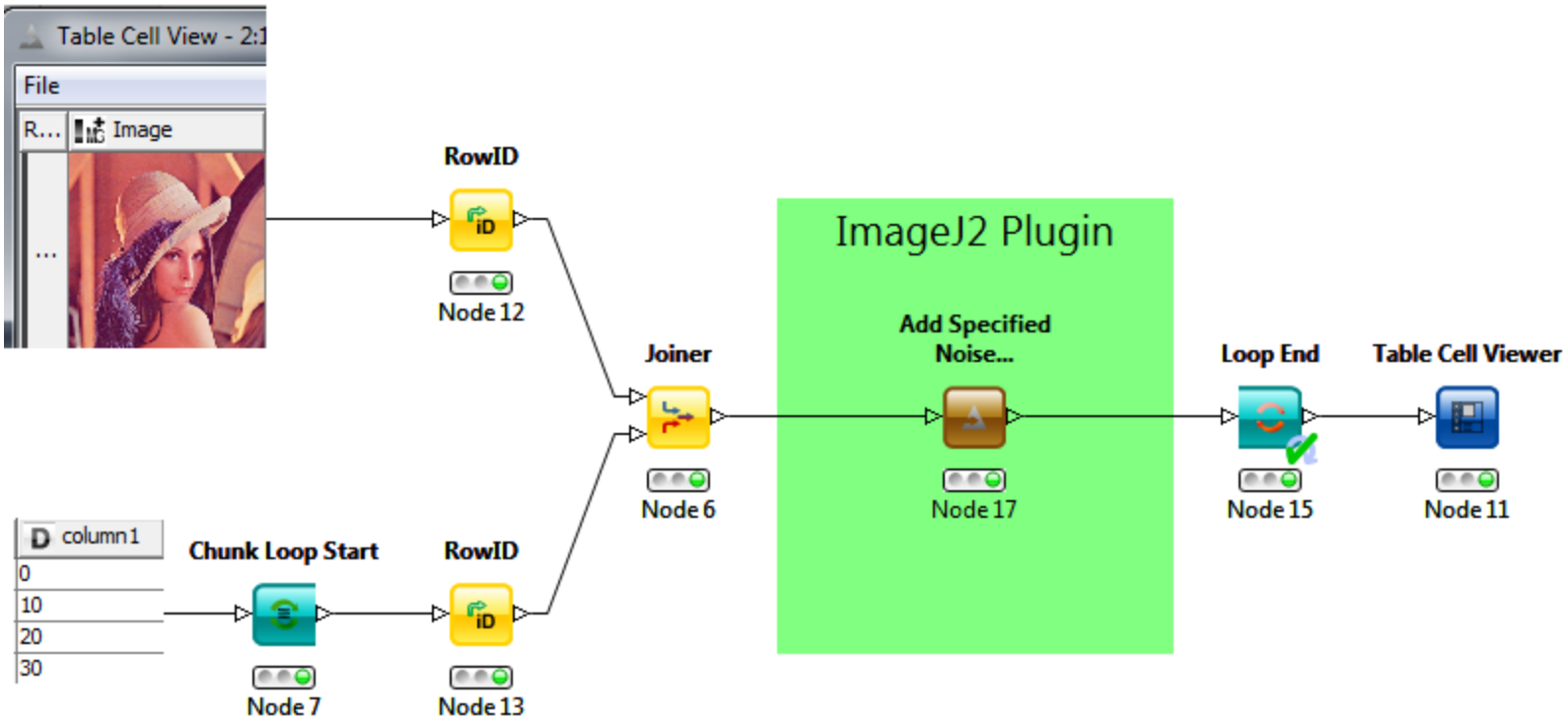
ImageJ2 plugin in a workflow that uses loops to vary a plugin parameter.

Example workflow



ImageJ2 plugin in a workflow that uses loops to vary a plugin parameter.

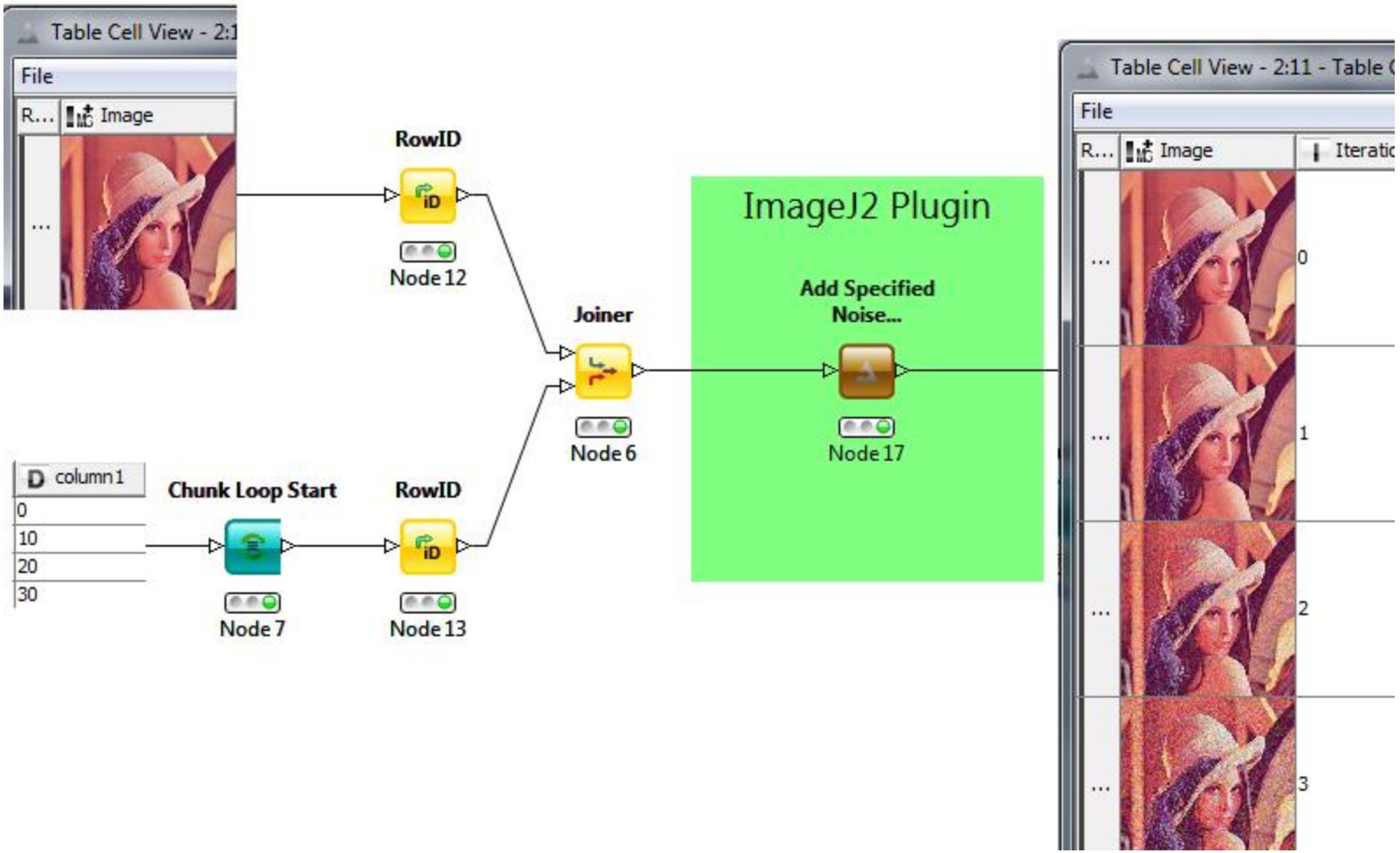
Example workflow



standard deviations

ImageJ2 plugin in a workflow that uses loops to vary a plugin parameter.

Example workflow



Example Code

```
@Plugin(headless = true)
public class MyMiniPlugin implements Command {

        @Parameter(type = ItemIO.OUTPUT)
        private String output;

        public void run() {
            output = "a working plugin ready for KNIME and ImageJ2";
        }
}
```

Example Code

implement the ImageJ2 Command interface

```
@Plugin(headless = true)
public class MyMiniPlugin implements Command {

        @Parameter(type = ItemIO.OUTPUT)
        private String output;

        public void run() {
            output = "a working plugin ready for KNIME and ImageJ2";
        }
}
```

Example Code

mark it as headless

```
@Plugin(headless = true)
public class MyMiniPlugin implements Command {

        @Parameter(type = ItemIO.OUTPUT)
        private String output;

        public void run() {
            output = "a working plugin ready for KNIME and ImageJ2";
        }
}
```

Example Code

```
@Plugin(headless = true)
public class MyMiniPlugin implements Command {

        @Parameter(type = ItemIO.OUTPUT)
        private String output;

        public void run() {
            output = "a working plugin ready for KNIME and ImageJ2";
        }
}
```

add your code

Extended Example Code

add some annotation sugar

```
@Plugin(menu = {@Menu(label = "Plugins"), @Menu(label = "MyMiniPlugin") },
        iconPath = "KnimeIcon.png",
        headless = true)
public class MyMiniPlugin implements Command {

        @Parameter(type = ItemIO.OUTPUT)
        private String output;

        public void run() {
            output = "a working plugin ready for KNIME and ImageJ2";
        }
}
```

Additional things to know

- Packaging
  - extract as jar
  - with Sezpoz annotations (discovery)

- Installation
  - place in ImageJ2 plugin directory

  - as fragment of the KNIME ImageJ2 Plugin
  - using the test installer
    (Preferences/KNIME/Image Processing Plugin)

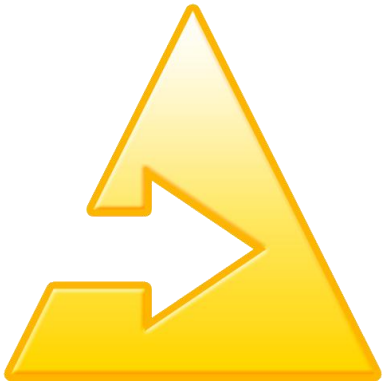
Additional things to know

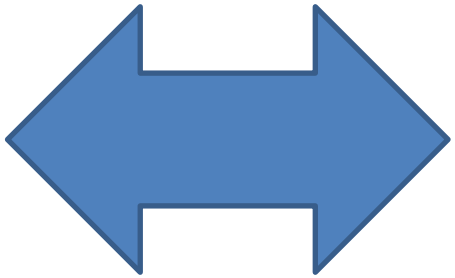
- ImageJ2 is still beta
  - the ImageJ2 plugin is beta
  - intended for developers not for production
  - especially the installation of ImageJ2 plugins is not final (*)

(*) KNIME workflows should not depend on local KNIME nodes that require manual installation. In order to facilitate sharing of workflows it is best to depend on nodes that are accessible via public repositories. The test installer is intended for local testing by developers.
At the moment no final ImageJ2 plugin installation mechanism exists. The plugin should be used as developer preview.

ImageJ2 Integration (Beta)

code once





use twice