User guide

JS Complete API Reference

```
const InputTable: IDataSource;      // Source of the data from the first
port
const InputTables: IDataSource[];   // Array of the input data sources
const OutputTable: IOutputTable;    // Output data source
const InputVariables: IVariables;   // Input variables

// Enumerations describing metadata of fields and variables
enum DataType { None, Boolean, DateTime, Float, Integer, String, Variant }
// Data type
enum DataKind { Undefined, Continuous, Discrete }
// Data kind
enum UsageType { Unspecified, Excluded, Useless, Active,
// Usage type of fields
                 Predicted, Key, Group, Value, Transaction, Item }

// View of the data set columns
interface IColumn extends Iterable<boolean | number | string | Date |
undefined> {
    readonly Index: number;
// Index
    readonly Name: string;
// Name
    readonly DisplayName: string;
// Caption
    readonly DataType: DataType;
// Data type
    readonly DataKind: DataKind;
// Data kind
    readonly DefaultUsageType: UsageType;
// Field usage type
    readonly RowCount: number;
// Number of values

    // Get method returns the column value by index
    Get(row: number): boolean | number | string | Date | undefined;
    // IsNull method checks for the Null column value
    IsNull(row: number): boolean;
}
```

```
// Access to iterated list of columns by name and index
interface IColumns extends Iterable<IColumn> {
    [name: string]: IColumn;
    [index: number]: IColumn;
}

// Properties and methods of data set
interface IDataSource {
    readonly Columns: IColumns;
// List of columns
    readonly ColumnCount: number;
// List of columns
    readonly RowCount: number;
// Row count

    // Get method returns the value of the set row in the set column
    Get(row: number, col: number | string): boolean | number | string |
Date | undefined;
    // IsNull method checks for the Null value of the set row in the set
column
    IsNull(row: number, col: number | string): boolean;
}

// Additional methods are defined for the output data set
interface IOutputTable extends IDataSource {
    // Append method enables to append a record to data set
    Append(): void;
    // Set method enables to set the value of the set field to the records
created by Append method
    Set(col: number | string, value: boolean | number | string | Date |
null | undefined): void;
}

// View of input variable
interface IVariable {
    readonly Index: number;
// Index
    readonly Name: string;
// Name
    readonly DisplayName: string;
// Caption
```

```
    readonly DataType: DataType;
// Data type
    readonly Value: boolean | number | string | Date | undefined;
// Value
    readonly IsNull: boolean;
// Check for Null
}

// Access to iterated list of input variables by name and index
interface IVariableItems extends Iterable<IVariable> {
    [name: string]: IVariable;
    [index: number]: IVariable;
}
View of input port variables
interface IVariables {
    readonly Items: IVariableItems;
    readonly Count: number;
}

// require is used for import of CommonJS modules
const require: IRequire;

// View of module
interface IModule {
    readonly id: string;              // identifier
    parent?: this;                    // calling module
    filename?: string;                // full path to module
    loaded: boolean;                  // whether the module is completely
loaded
    exports: any;                     // exported object
}

// require function accepts the module identifier and returns the exports
module property
interface IRequireFunction {
    (id: string): any;
}

interface IRequire extends IRequireFunction  {
    // resolve function returns the full path to the module
    resolve: (id: string) => string;
    // cache of modules
```

```
    cache: { [resolvedId: string]: IModule]; };
}

// Global console object
var console: Console;

interface Console {
     // assert method enables to direct the message to console if the
first argument is false
    assert(condition?: boolean, ...data: any[]): void;
    // error method shows error message
    error(...data: any[]): void;
    // warn methods show warning message
    warn(...data: any[]): void;
    // info and log methods show information message
    info(...data: any[]): void;
    log(...data: any[]): void;
    // clear method enables to clean the message console in the preview
window
    clear(): void;
}
```