

The image features the Hybris Software logo, which consists of a stylized 'h' inside a circle, followed by the words 'hybris software' in a bold, sans-serif font. Below this, the text 'An SAP Company' is written in a smaller, sans-serif font. The background is a dark blue gradient with a large, abstract, low-poly geometric shape in a lighter blue color on the right side.

(h) hybris software
An SAP Company

hybris Developer Training Part I - Core Platform

Validation Framework

Data Validation Framework

Validation service

Administration cockpit

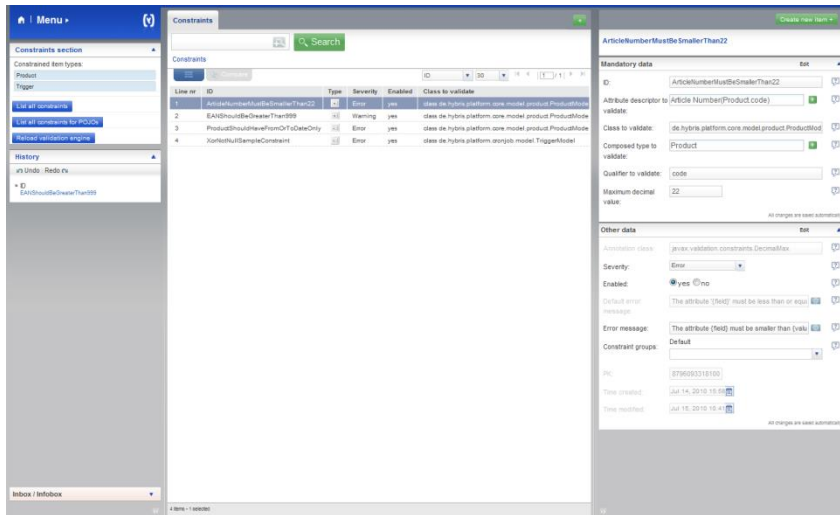
Cockpit integration

- The hybris Data Validation Framework is based on the JSR 303 Java validation specification
- It offers an easy and extensible way to validate data
- You can provide user-friendly and meaningful notifications when allowing users to provide or re-enter valid input
- Only validated data is passed on to the persistence layer
- In contrast to the JSR 303 specification, hybris constraints can be created and modified at runtime

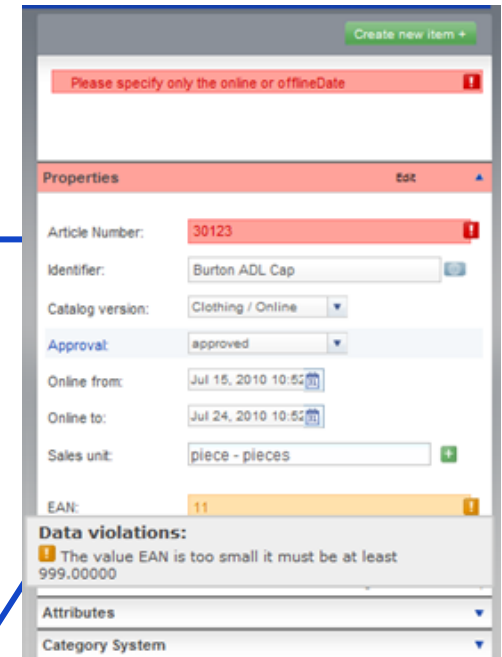


- ➔ Constraint definitions are written in Java or XML
- ➔ Constraint instances are persisted in the hybris database as items
- ➔ You may define your own constraint types
- ➔ hybris Data Validation consists of the three following areas:
 - ➔ The Validation Service is a service in the ServiceLayer which defines constraints and performs data validation
 - ➔ The Administration Cockpit allows you to create and manage constraints
 - ➔ Cockpit integration provides validation feedback to the user

Administration Cockpit



Cockpit Integration



Validation

ValidationService

Validation Engine

Data Validation Framework

Validation Service

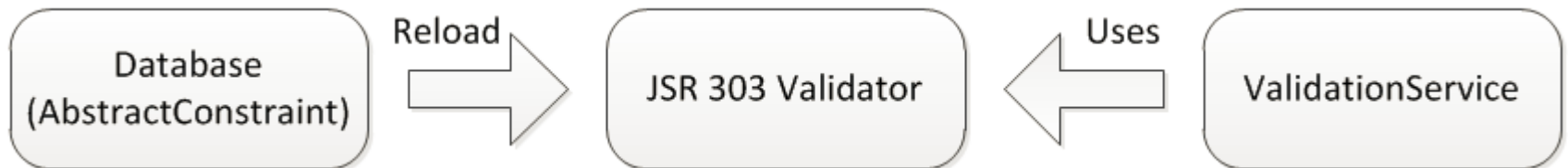
Administration Cockpit

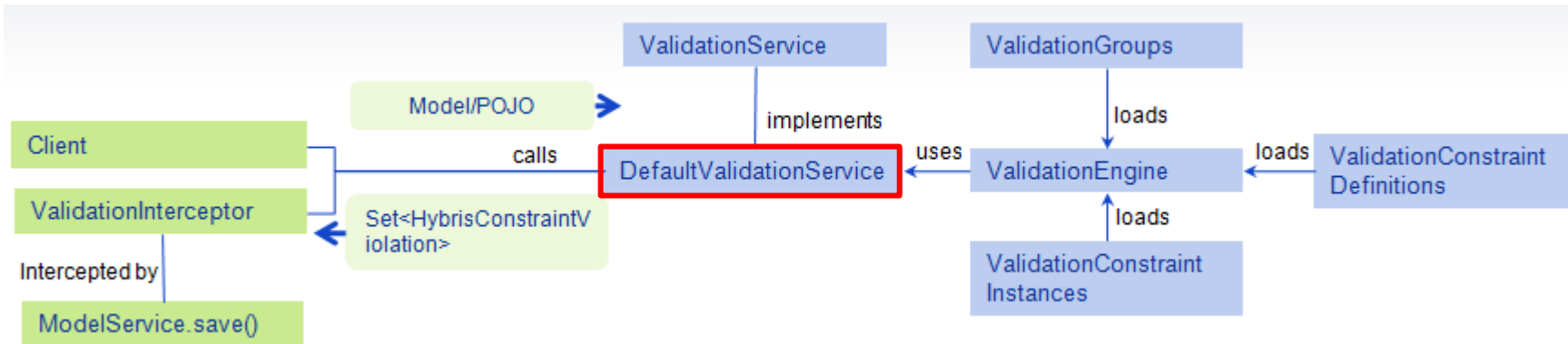
Cockpit Integration

Key Features



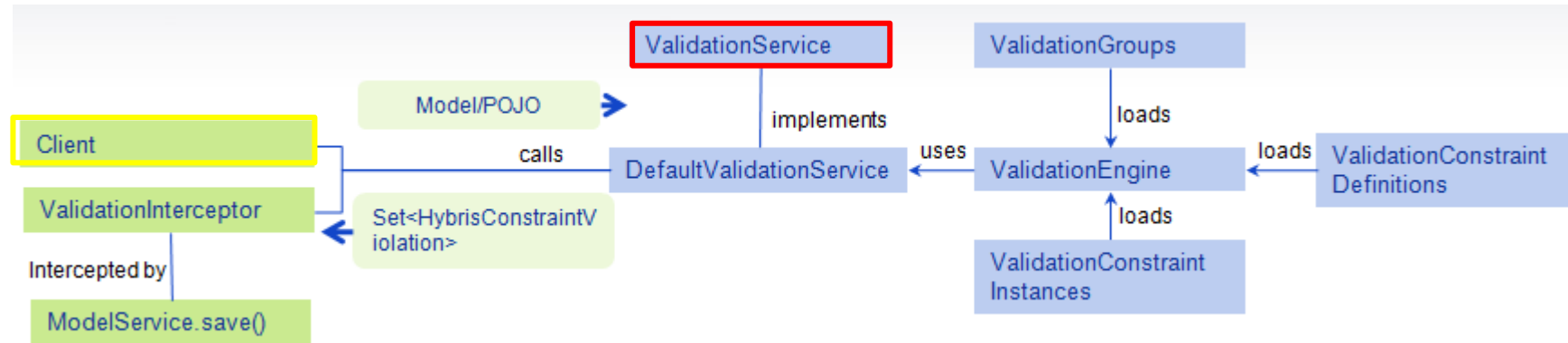
- ➔ The Validation Service loads the validation engine with constraints
- ➔ Data is validated based on a set of validation constraints
- ➔ All `modelService.save` calls are intercepted aborting save operations where necessary
- ➔ Validation methods may be called explicitly





- ➔ Validation is performed by the **DefaultValidationService**
- ➔ This service uses a JSR 303-compliant validation engine
- ➔ The Validation Service contains two groups of methods:
 - ➔ Validation methods to validate items, properties or values
 - ➔ Control methods to alter the behavior of the validation engine

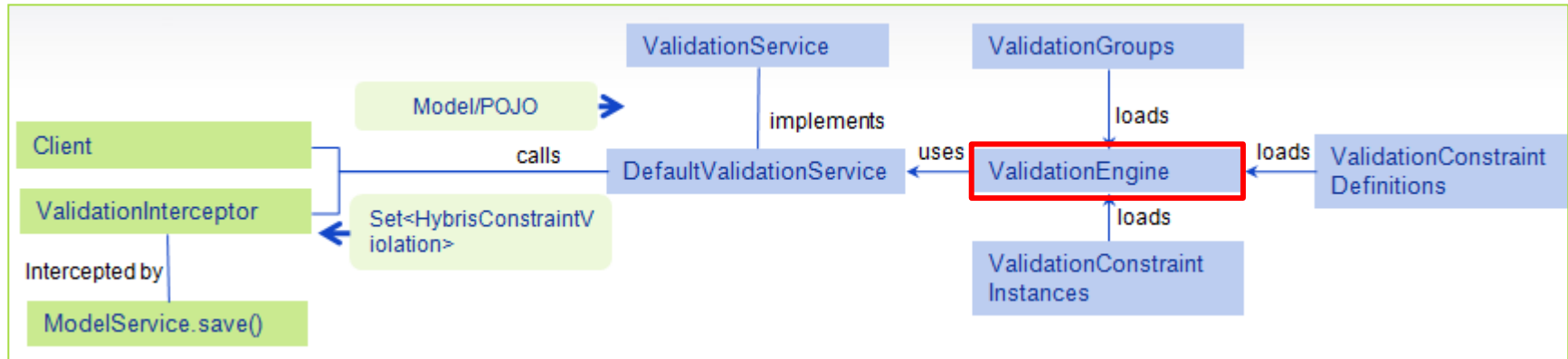
Example



```
interface ValidationService {
void reloadValidationEngine();
<T> Set<HybrisConstraintViolation> validate(T object,
    Collection<ConstraintGroupModel> g);
<T> Set<""> validateProperty(T object, String propertyName,
    Collection<ConstraintGroupModel> g);
<T> Set<""> validateValue(Class<T> beanType, String propertyName, Object o,
    Collection<ConstraintGroupModel> g);
void setActiveConstraintGroups(Collection<ConstraintGroupModel> groups);
...
}
```

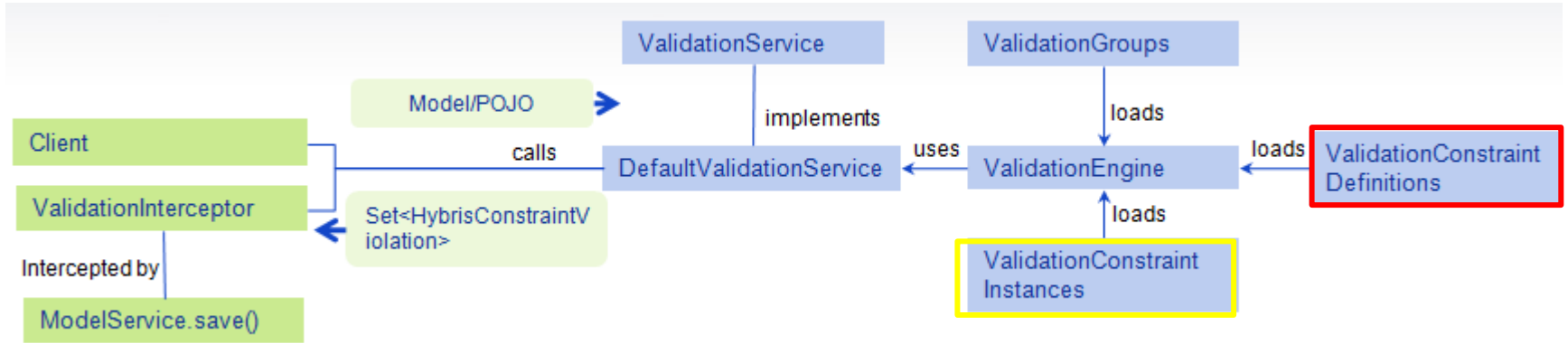
Example Client usage:

```
ProductModel model = new ProductModel();
Set<HybrisConstraintViolation> constraintViolations = validationService.validate(model);
```



- ➔ The Validation Engine is loaded with JSR 303-compliant constraint instances at run-time
- ➔ It can be reloaded with new or modified constraint instances
- ➔ All JSR 303 constraint types are supported
- ➔ New constraint types may be created but require new files, recompilation and a system initialization or update.

Constraints



All JSR 303-compliant validation constraints can be used:

Java annotation:

NotNullConstraint



```
public class User {  
    @NotNull  
    private String firstname;  
}
```

- ➔ The Data Validation framework provides implementations of all constraint types defined in the JSR 303 specification and few additional ones:
 - ➔ `PatternConstraint`
 - ➔ `PastConstraint`, `FutureConstraint`
 - ➔ `MinConstraint`, `MaxConstraint`
 - ➔ `DecimalMinConstraint`, `DecimalMaxConstraint`
 - ➔ `SizeConstraint`
 - ➔ `AssertTrueConstraint`, `AssertFalseConstraint`
 - ➔ `IsNullConstraint`, `NotNullConstraint`
 - ➔ `NotEmpty`, `NotBlank`
 - ➔ `XorNotNull` Annotation
 - ➔ `Dynamic` Annotation

→ Severity

- Each constraint can be assigned an Error/Warning/Info severity level
- Error constraints cause a save operation to be aborted
- Users may choose to ignore Warning constraints and save anyway

→ Groups

- Constraints can be assigned to zero or more groups (zero = default group)
- The engine can be told which constraint group(s) to use for validation

→ There are attribute, type and dynamic constraints

Data Validation Framework

Validation Service

Administration Cockpit

Cockpit Integration

Administration Cockpit overview



The screenshot shows the Administration Cockpit interface. The left sidebar contains a 'Menu' with 'Admin' and 'Validation' options. Under 'Validation', there are sections for 'Constraints section' and 'Constraint groups section'. The 'Constraints section' has links for 'List all constraints', 'List all constraints for POJOs', and 'Reload validation engine'. The 'Constraint groups section' has a link for 'List all constraint groups'. The 'History' section shows 'Undo' and 'Redo' actions. The central area displays a table of constraints with columns: Line nr, ID, Type, Severity, Enabled, and Needs reload. The table lists 18 items. The right sidebar shows the 'Size constraint (ProductNameMinSizeCoverageConstraint)' configuration, including fields for ID, Attribute descriptor to validate, Enabled status, Severity, Minimum range value, Maximum range value, Default error message, and Error message.

Line nr	ID	Type	Severity	Enabled	Needs reload
1	ProductCodeMinSizeCoverageConstraint		Error	yes	no
2	ProductCodeSizeSampleConstraint		Error	yes	no
3	ProductDescriptionMinSizeCoverageConstraint		Error	yes	no
4	ProductDescriptionNotNullConstraint		Error	yes	no
5	ProductEanNotEmptyConstraint		Error	yes	no
6	ProductNameMinSizeCoverageConstraint		Error	yes	no
7	ProductNameNotNullConstraint		Error	yes	no
8	ProductPictureNotNullConstraint		Error	yes	no
9	ProductPricesMinSizeCoverageConstraint		Error	yes	no
10	VariantProductCodeMinSizeCoverageConstraint		Error	yes	no
11	VariantProductDescriptionMinSizeCoverageConstraint		Error	yes	no
12	VariantProductDescriptionNotNullConstraint		Error	yes	no
13	VariantProductEanNotEmptyConstraint		Error	yes	no
14	VariantProductNameMinSizeCoverageConstraint		Error	yes	no
15	VariantProductNameNotNullConstraint		Error	yes	no
16	VariantProductPictureNotNullConstraint		Error	yes	no
17	VariantProductPricesMinSizeCoverageConstraint		Error	yes	no
18	XorNotNullSampleConstraint		Error	yes	no

List constraints

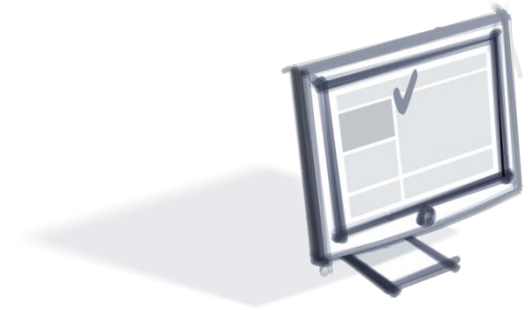
Reload the validation engine

Create and manage constraints

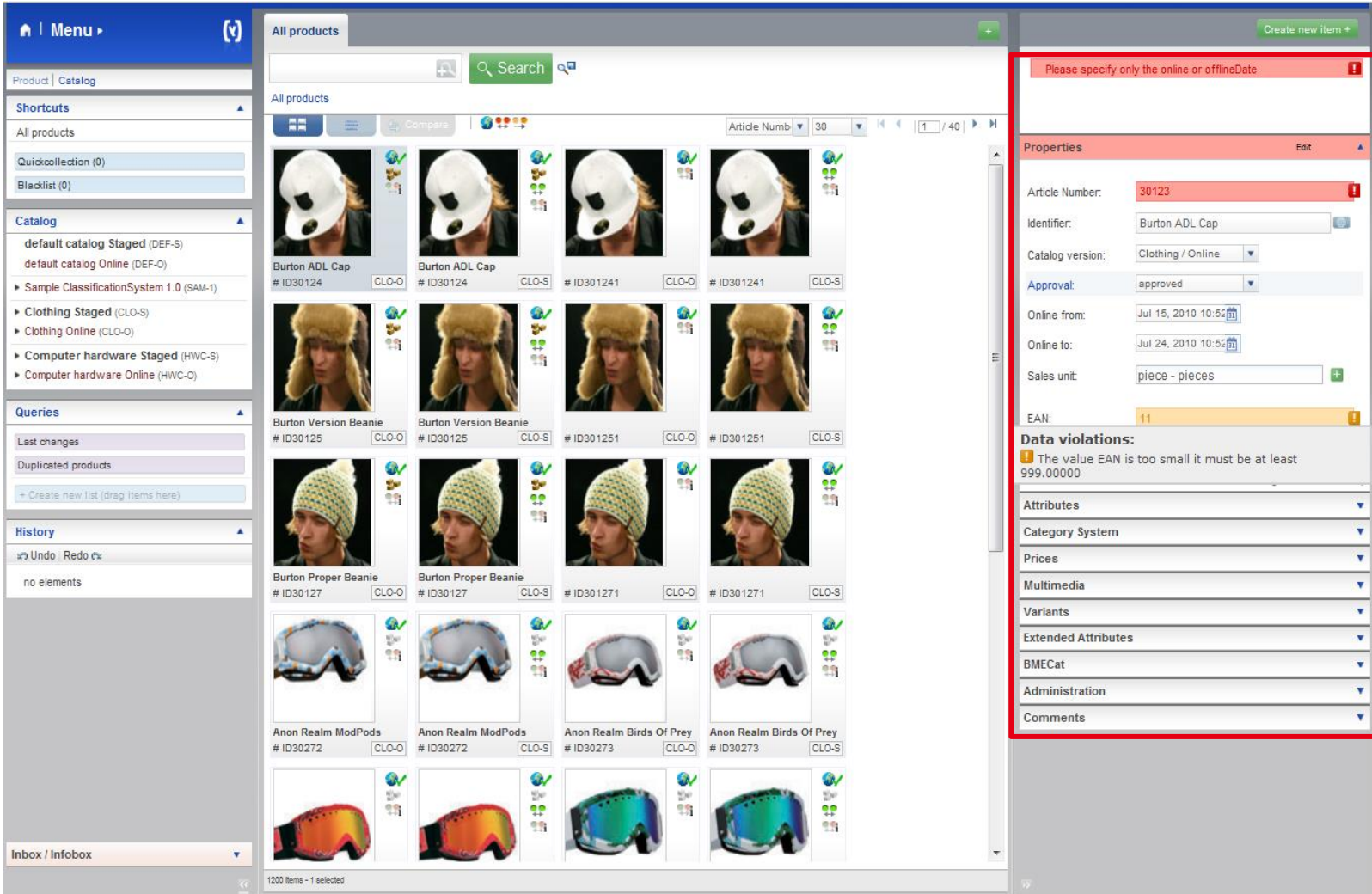
With the administration cockpit you can:

- Create and manage constraints and constraint groups
- Filter constraints
- List constraint declarations
- Reload the validation engine

Although this can also be done with the hMC, we recommend using the administration cockpit which is more user-friendly and intuitive.



Data Validation Framework Validation Service Administration Cockpit **Cockpit Integration**



The screenshot displays the Cockpit Integration interface. On the left is a sidebar with navigation options: Menu, Product | Catalog, Shortcuts, Catalog, Queries, and History. The main area shows a grid of product images, including Burton ADL Caps, Burton Version Beanies, Burton Proper Beanies, Anon Realm ModPods, and Anon Realm Birds Of Prey. On the right, the 'Properties' panel for a selected item (Article Number: 30123) is visible. It contains fields for Identifier, Catalog version, Approval, Online from/to dates, Sales unit, and EAN. A red box highlights the 'Data violations' section, which states: 'The value EAN is too small it must be at least 999.00000'. Below this, a list of attributes is shown, including Category System, Prices, Multimedia, Variants, Extended Attributes, BMECat, Administration, and Comments.

Data validation is integrated into the editing area of cockpits

Users are notified of all validation issues

1. Explain the main features of the hybris Validation Framework and the three areas of which it consists.
2. Name the specification on which the hybris Validation Framework is based.
3. Give a short description of the Validation Service architecture.
4. Is it possible to create your own constraint types? If so, how do you declare a custom constraint type?
5. What happens to a save operation if a constraint violation occurs?
6. What is the Administration Cockpit? What are its primary uses?

- ➔ wiki.hybris.com/display/release5/Data+Validation+Framework
- ➔ jcp.org/aboutJava/communityprocess/final/jsr303/index.html
- ➔ [wiki.hybris.com/display/release5/
Administration+Cockpit+-+Business+Guide](http://wiki.hybris.com/display/release5/Administration+Cockpit+-+Business+Guide)

(x)