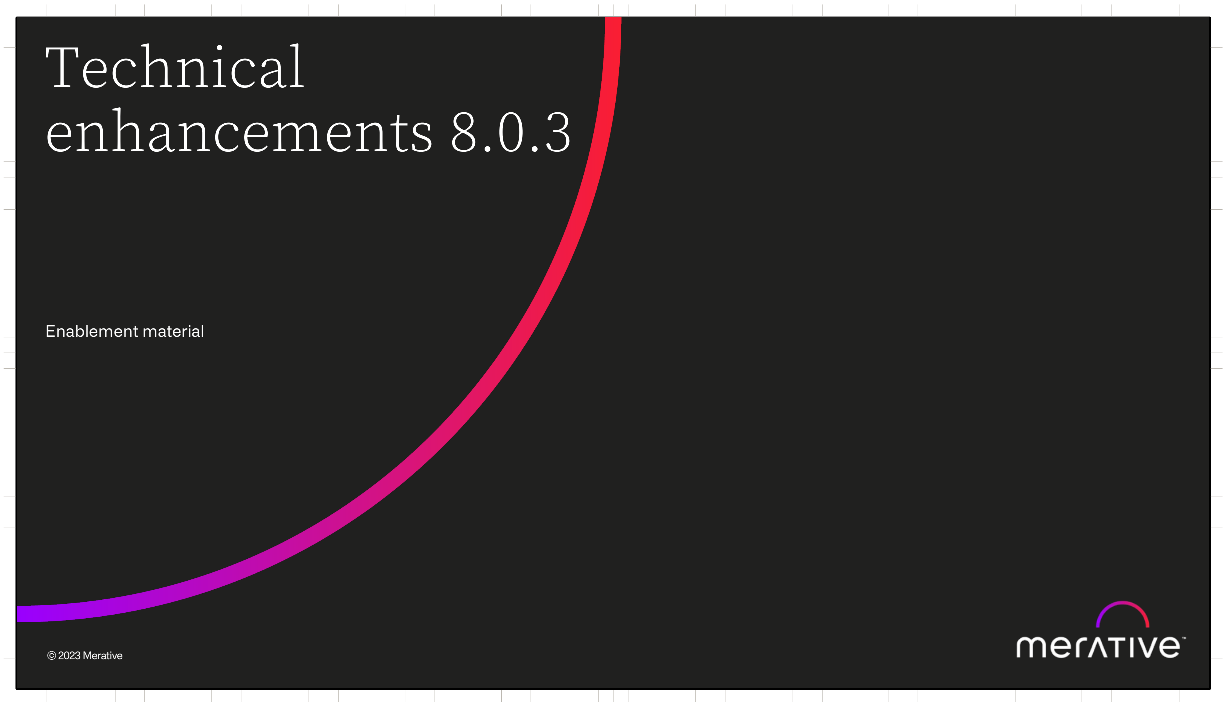


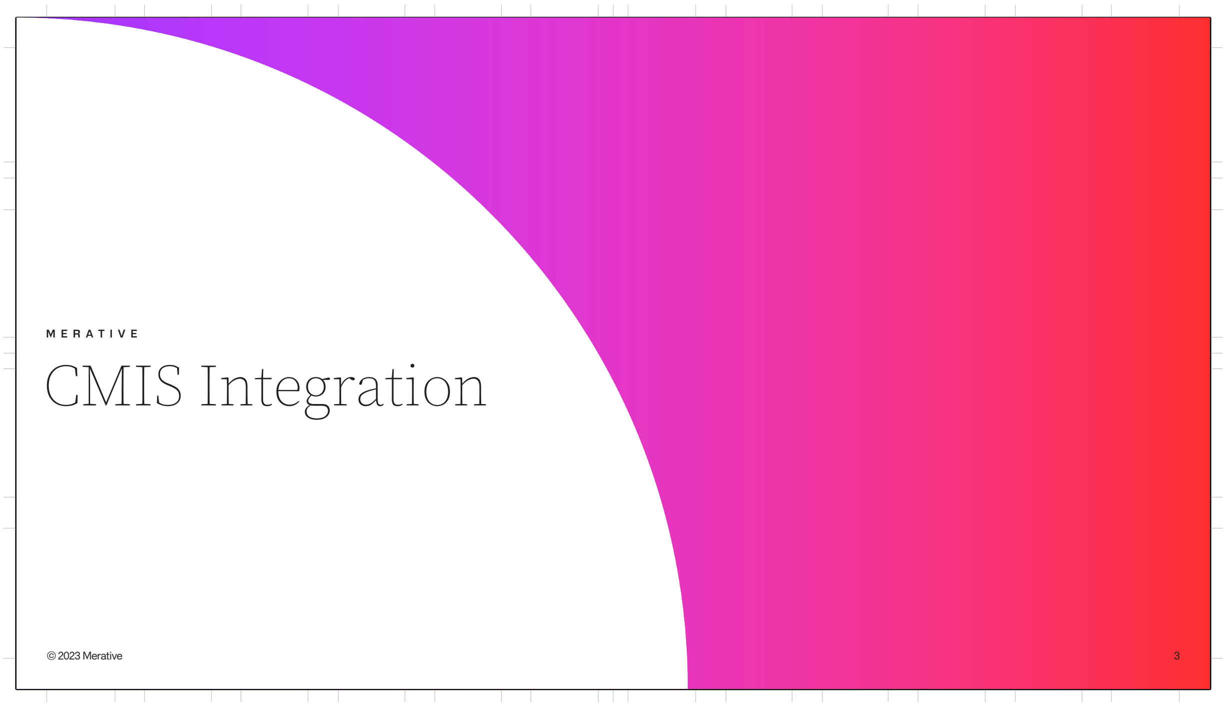
Technical enhancements 8.0.3

Enablement material

Hello and welcome to this deep dive presentation that introduces the technical enhancements that are available in Merative Social Program Management (SPM) V8.0.3

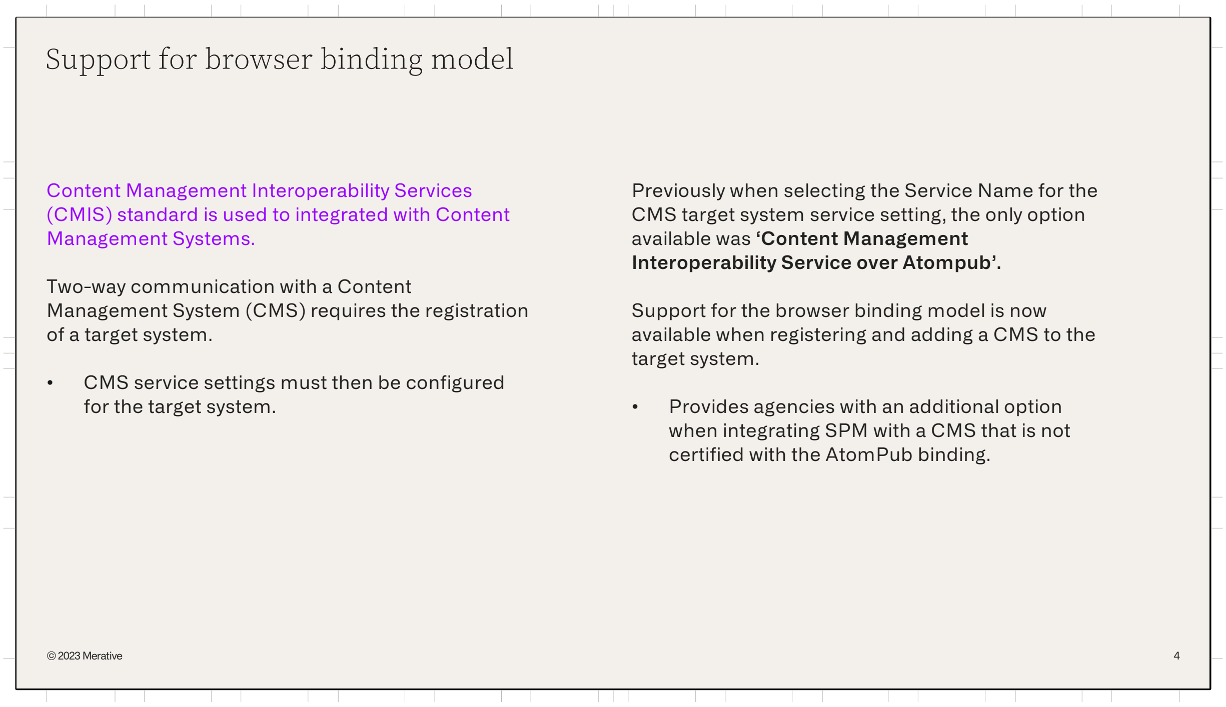
Agenda

1. CMIS integration
2. Applying deductions to underpayments
3. Currency symbol placement and spacing
4. Add on Development Environment v1.1



MERATIVE

CMIS Integration



Support for browser binding model

Content Management Interoperability Services (CMIS) standard is used to integrated with Content Management Systems.

Two-way communication with a Content Management System (CMS) requires the registration of a target system.

- CMS service settings must then be configured for the target system.

Previously when selecting the Service Name for the CMS target system service setting, the only option available was **'Content Management Interoperability Service over Atompub'.**

Support for the browser binding model is now available when registering and adding a CMS to the target system.

- Provides agencies with an additional option when integrating SPM with a CMS that is not certified with the AtomPub binding.

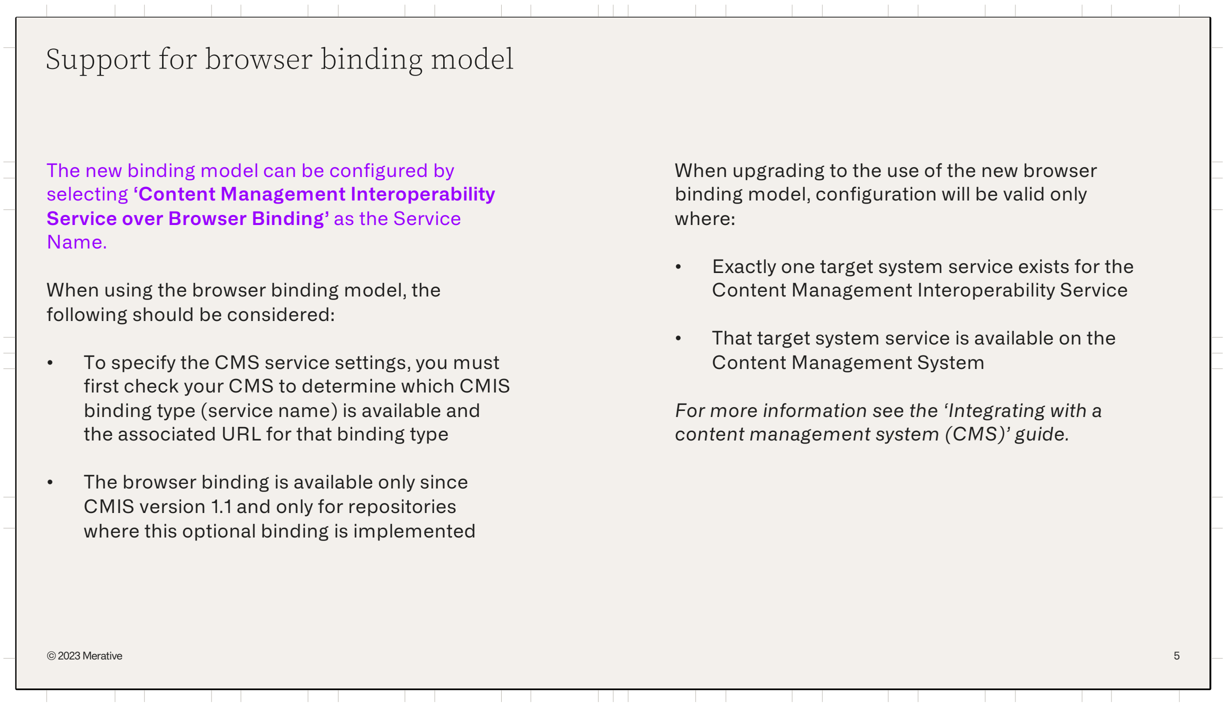
Merative Social Program Management uses the Content Management Interoperability Services (CMIS) standard to integrate with Content Management Systems.

To permit two-way communication between Merative Social Program Management and a Content Management System (CMS), a target system must be registered for the CMS. CMS service settings must then be configured for the target system.

Previously when selecting the Service Name for the CMS target system service setting, the only option available was 'Content Management Interoperability Service over Atompub'.

Support for the browser binding model is now available when registering and adding a CMS to the target system.

- The introduction of support for this additional binding model now provides agencies with an additional option when integrating SPM with a CMS that is not certified with the AtomPub binding.



Support for browser binding model

The new binding model can be configured by selecting **'Content Management Interoperability Service over Browser Binding'** as the Service Name.

When using the browser binding model, the following should be considered:

- To specify the CMS service settings, you must first check your CMS to determine which CMIS binding type (service name) is available and the associated URL for that binding type
- The browser binding is available only since CMIS version 1.1 and only for repositories where this optional binding is implemented

When upgrading to the use of the new browser binding model, configuration will be valid only where:

- Exactly one target system service exists for the Content Management Interoperability Service
- That target system service is available on the Content Management System

For more information see the 'Integrating with a content management system (CMS)' guide.

5

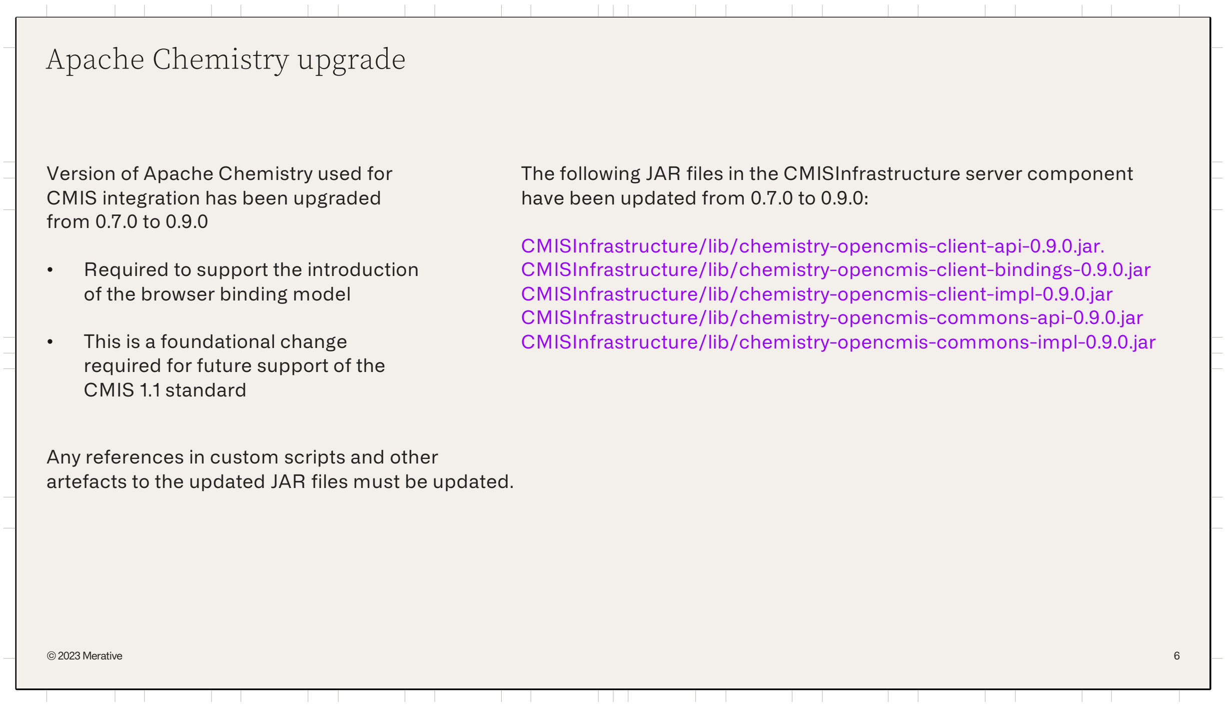
The new binding model can be configured by selecting 'Content Management Interoperability Service over Browser Binding' as the Service Name.

When using the browser binding model, the following should be considered:

- To specify the CMS service settings, you must first check your CMS to determine which CMIS binding type (service name) is available and the associated URL for that binding type.
- The Atompub binding is available for all repositories that support CMIS version 1.0 and 1.1. The browser binding is available only since CMIS version 1.1 and only for repositories where this optional binding is implemented.

When upgrading to the use of the new browser binding model, configuration will be valid only where exactly one target system service exists for the Content Management Interoperability Service and that target system service is available on the Content Management System.

For more information see the ‘Integrating with a content management system (CMS)’ guide.


Apache Chemistry upgrade

Version of Apache Chemistry used for CMIS integration has been upgraded from 0.7.0 to 0.9.0

- Required to support the introduction of the browser binding model
- This is a foundational change required for future support of the CMIS 1.1 standard

Any references in custom scripts and other artefacts to the updated JAR files must be updated.

The following JAR files in the CMISInfrastructure server component have been updated from 0.7.0 to 0.9.0:

CMISInfrastructure/lib/chemistry-opencmis-client-api-0.9.0.jar.
CMISInfrastructure/lib/chemistry-opencmis-client-bindings-0.9.0.jar
CMISInfrastructure/lib/chemistry-opencmis-client-impl-0.9.0.jar
CMISInfrastructure/lib/chemistry-opencmis-commons-api-0.9.0.jar
CMISInfrastructure/lib/chemistry-opencmis-commons-impl-0.9.0.jar

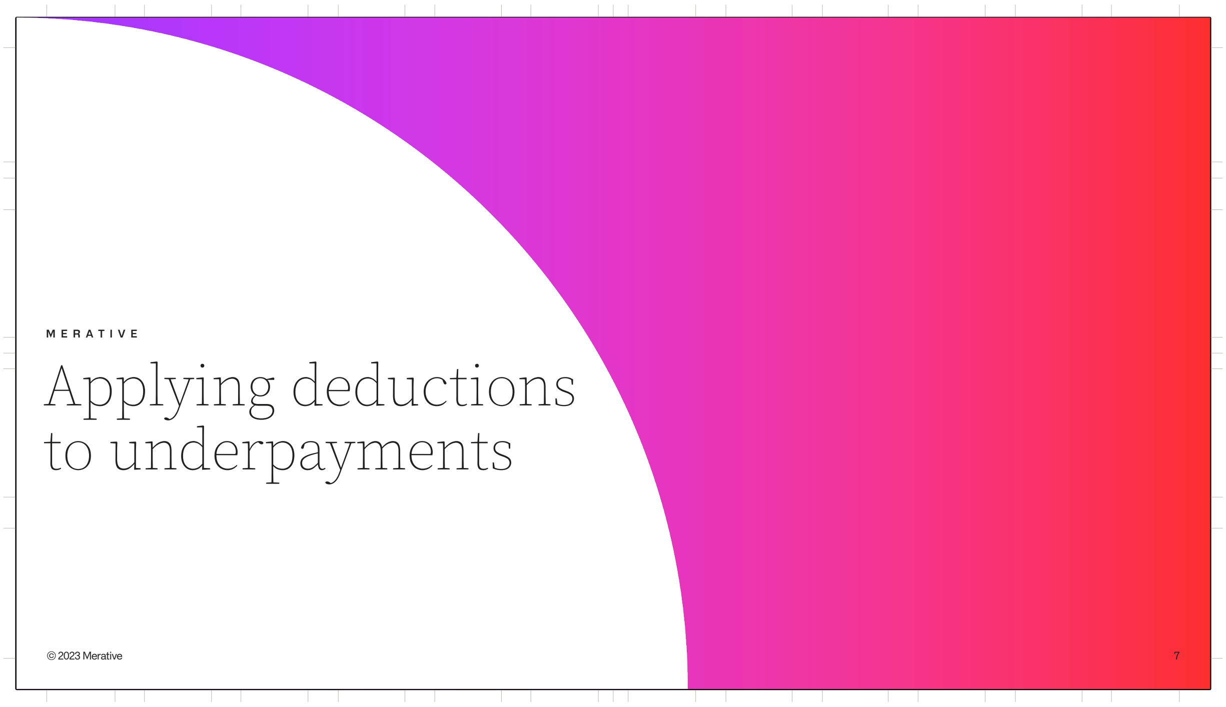
The version of Apache Chemistry used for CMIS integration has been upgraded from 0.7.0 to 0.9.0 in initial support of the CMIS 1.1 standard.

- This was required in order to support the introduction of the browser binding model and is a foundational change required for future support of the CMIS 1.1 standard

As a result of these updates, the following changes have been made in the CMISInfrastructure server component:
•CMISInfrastructure/lib/chemistry-opencmis-client-api-0.9.0.jar - version updated from 0.7.0 to 0.9.0.
•CMISInfrastructure/lib/chemistry-opencmis-client-bindings-0.9.0.jar - version updated from 0.7.0 to 0.9.0.
•CMISInfrastructure/lib/chemistry-opencmis-client-impl-0.9.0.jar - version updated from 0.7.0 to 0.9.0.
•CMISInfrastructure/lib/chemistry-opencmis-commons-api-0.9.0.jar - version updated from 0.7.0 to 0.9.0.

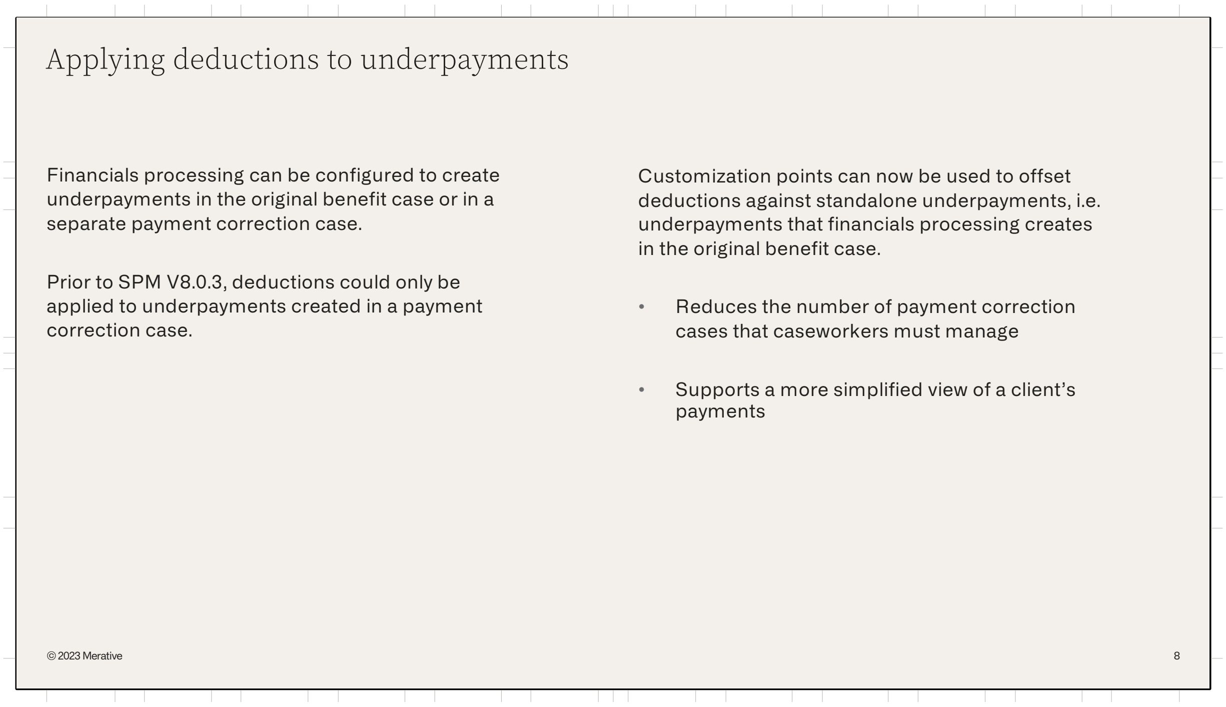
•CMISInfrastructure/lib/chemistry-opencmis-commons-impl-0.9.0.jar - version updated from 0.7.0 to 0.9.0.

Note that any references in custom scripts and other artefacts to the updated JAR files listed above must be updated.



MERATIVE

Applying deductions to underpayments



Applying deductions to underpayments

Financials processing can be configured to create underpayments in the original benefit case or in a separate payment correction case.

Prior to SPM V8.0.3, deductions could only be applied to underpayments created in a payment correction case.

Customization points can now be used to offset deductions against standalone underpayments, i.e. underpayments that financials processing creates in the original benefit case.

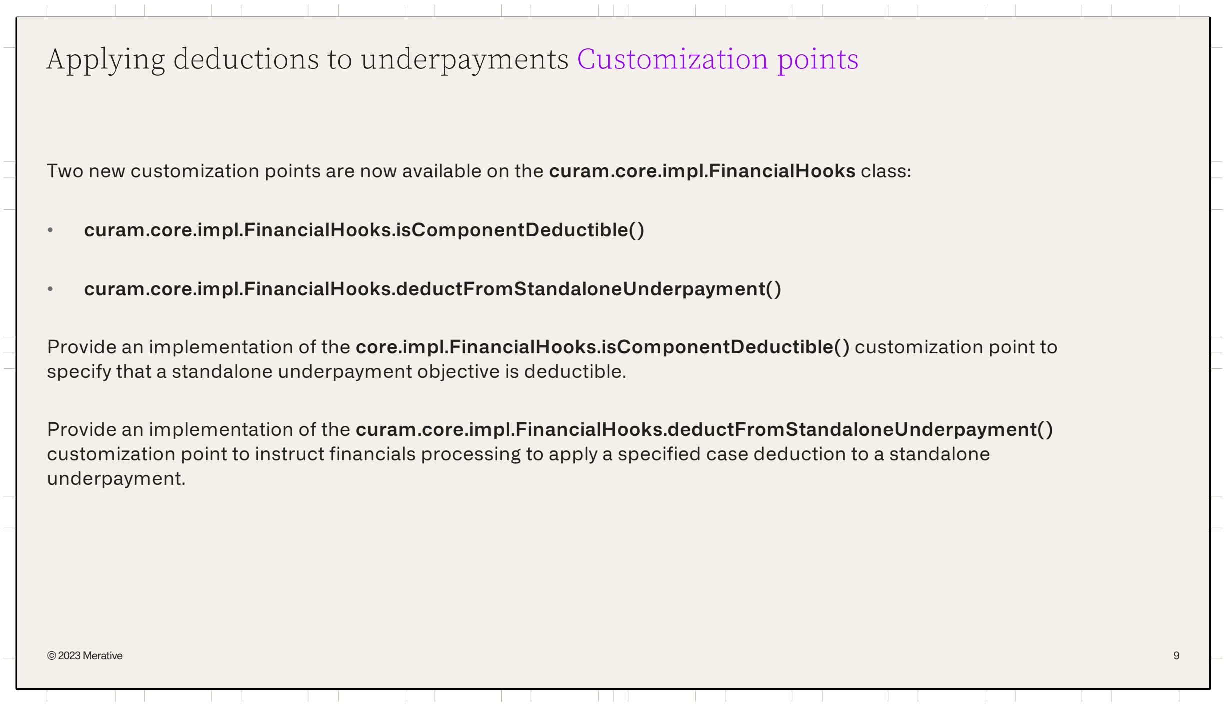
- Reduces the number of payment correction cases that caseworkers must manage
- Supports a more simplified view of a client's payments

Financials processing can be configured to create underpayments in the original benefit case or in a separate payment correction case during case reassessment.

Prior to SPM V8.0.3, deductions could only be applied to underpayments created in a payment correction case. As a result, agencies needed to create separate payment correction cases to manage the underpayments.

Customization points can now be used to offset deductions against standalone underpayments, i.e. underpayments that financials processing creates in the original benefit case.

This reduces the number of payment correction cases that caseworkers must manage and supports a more simplified view of a client's payments.



Applying deductions to underpayments Customization points

Two new customization points are now available on the **curam.core.impl.FinancialHooks** class:

- **curam.core.impl.FinancialHooks.isComponentDeductible()**
- **curam.core.impl.FinancialHooks.deductFromStandaloneUnderpayment()**

Provide an implementation of the **core.impl.FinancialHooks.isComponentDeductible()** customization point to specify that a standalone underpayment objective is deductible.

Provide an implementation of the **curam.core.impl.FinancialHooks.deductFromStandaloneUnderpayment()** customization point to instruct financials processing to apply a specified case deduction to a standalone underpayment.

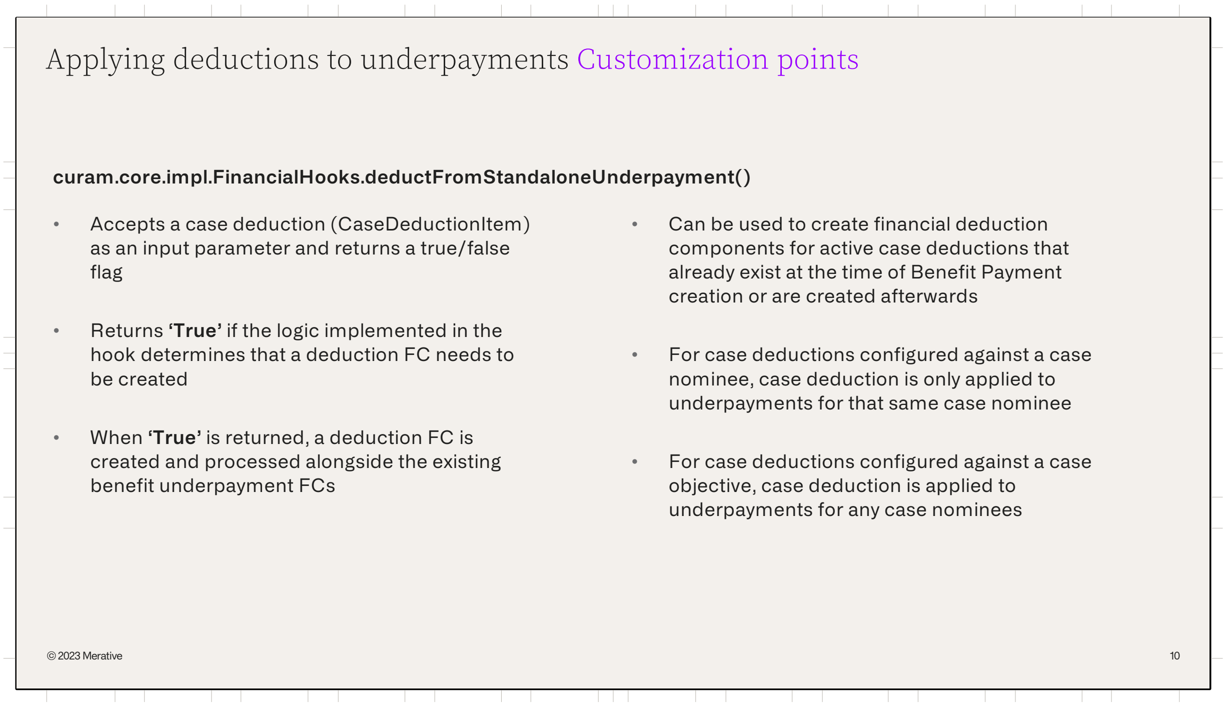
Two new customization points are now available on the curam.core.impl.FinancialHooks abstract class to offset deductions against standalone underpayments.

curam.core.impl.FinancialHooks.isComponentDeductible()
By default, an objective that is associated with an underpayment financial component on the original benefit case is not deductible.
You can specify that a standalone underpayment objective is deductible by providing an implementation of the curam.core.impl.FinancialHooks.isComponentDeductible() customization point.

curam.core.impl.FinancialHooks.deductFromStandaloneUnderpayment()
You can also instruct financials processing to apply a specified case deduction to a standalone underpayment by providing an implementation of the curam.core.impl.FinancialHooks.deductFromStandaloneUnderpayment() customization point.

Provide an implementation of the core.impl.FinancialHooks.isComponentDeductible() customization point to specify that a standalone underpayment objective is deductible.

Provide an implementation of the curam.core.impl.FinancialHooks.deductFromStandaloneUnderpayment() customization point to instruct financials processing to apply a specified case deduction to a standalone underpayment.



Applying deductions to underpayments Customization points

curam.core.impl.FinancialHooks.deductFromStandaloneUnderpayment()

- Accepts a case deduction (CaseDeductionItem) as an input parameter and returns a true/false flag
- Returns **'True'** if the logic implemented in the hook determines that a deduction FC needs to be created
- When **'True'** is returned, a deduction FC is created and processed alongside the existing benefit underpayment FCs
- Can be used to create financial deduction components for active case deductions that already exist at the time of Benefit Payment creation or are created afterwards
- For case deductions configured against a case nominee, case deduction is only applied to underpayments for that same case nominee
- For case deductions configured against a case objective, case deduction is applied to underpayments for any case nominees

© 2023 Merative

10

The customization point **curam.core.impl.FinancialHooks.deductFromStandaloneUnderpayment()** accepts a case deduction (CaseDeductionItem) as an input parameter and returns a true/false flag.

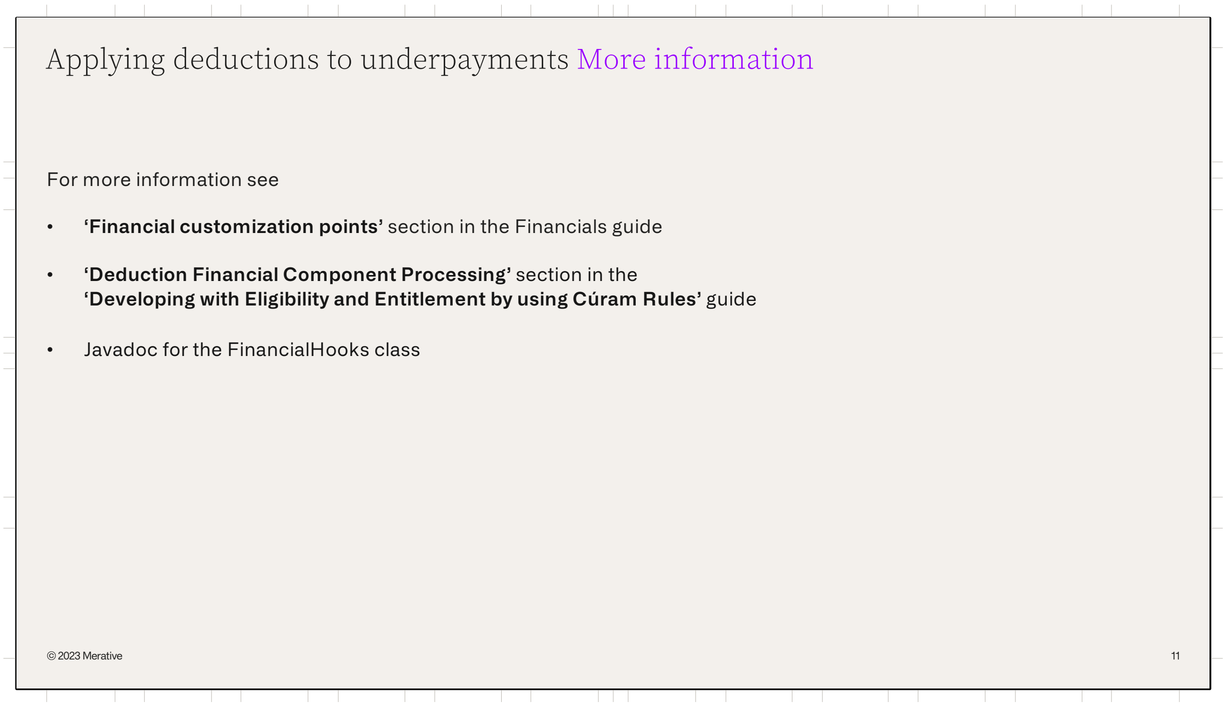
A true flag is returned if the logic implemented in the custom hook determines that a deduction FC needs to be created, i.e., the logic determines that the CaseDeductionItem received as an input parameter should be used to create a deduction FC for an underpayment FC.

When the interface returns a true flag a deduction FC is created and processed alongside existing benefit underpayment FCs and rolled up with them.

The customization point can be used to create financial deduction components for active case deductions that already exist at the time of the creation of the Benefit Underpayment, or for case deductions created after the Benefit Underpayment has been created.

Where a case deduction is configured against a case nominee, the case deduction is only applied to underpayments for that same case nominee.

If the deduction is configured against an objective, the case deduction is applied to underpayments for any of the nominees on the case.



Applying deductions to underpayments More information

For more information see

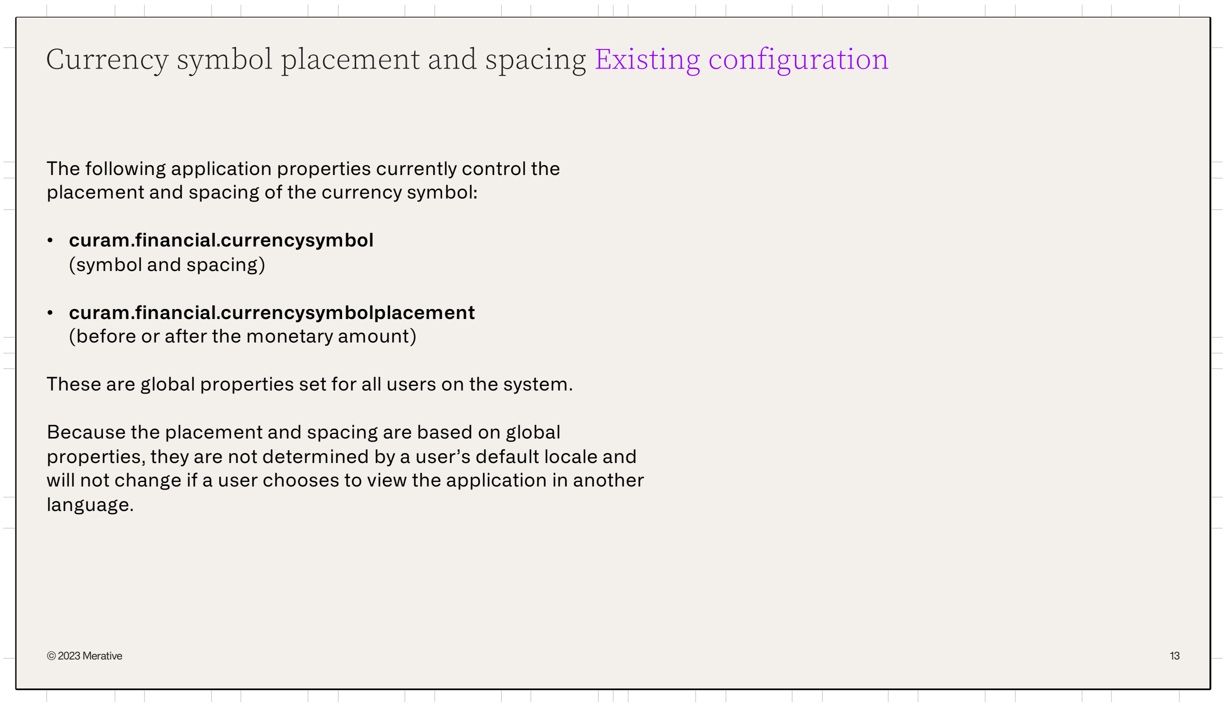
- **‘Financial customization points’** section in the Financials guide
- **‘Deduction Financial Component Processing’** section in the
  ‘Developing with Eligibility and Entitlement by using Cúram Rules’ guide
- Javadoc for the FinancialHooks class

For more information see:

- ‘Financial customization points’ section in the Financials guide
- ‘Deduction Financial Component Processing’ section in the
  ‘Developing with Eligibility and Entitlement by using Cúram Rules’ guide
- Javadoc for the FinancialHooks class

MERATIVE

Currency symbol placement and spacing



Currency symbol placement and spacing Existing configuration

The following application properties currently control the placement and spacing of the currency symbol:

- **curam.financial.currencysymbol**
  (symbol and spacing)
- **curam.financial.currencysymbolplacement**
  (before or after the monetary amount)

These are global properties set for all users on the system.

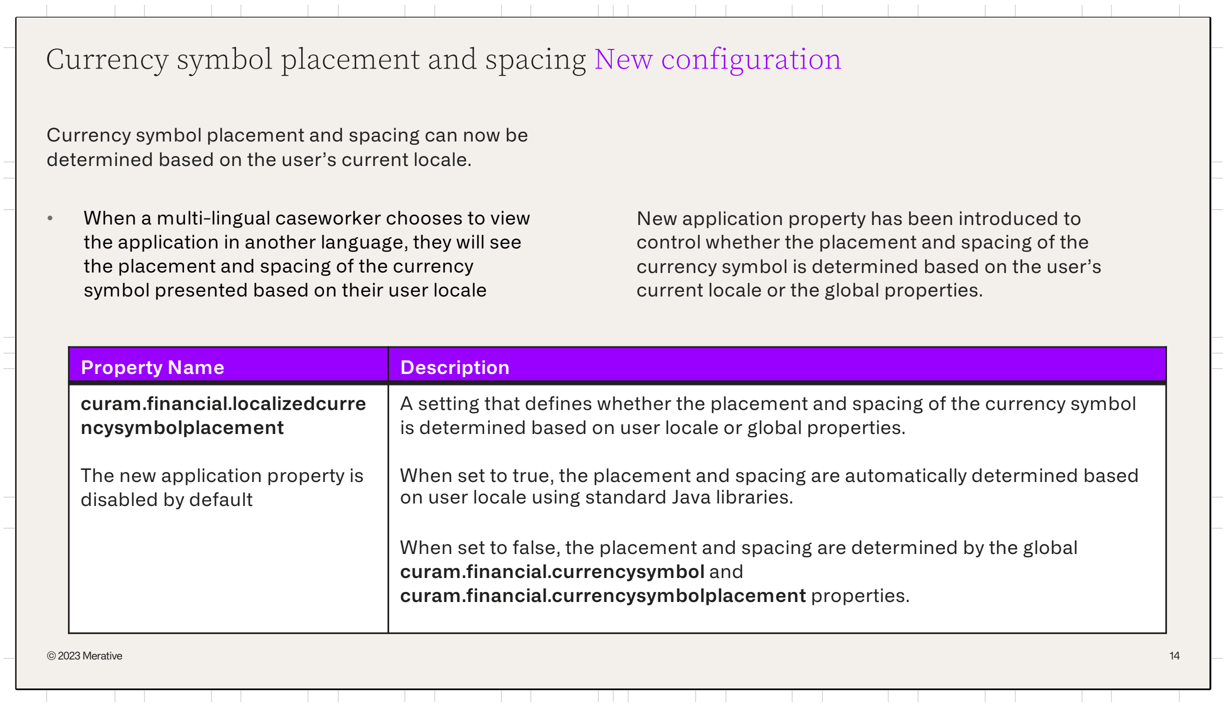
Because the placement and spacing are based on global properties, they are not determined by a user's default locale and will not change if a user chooses to view the application in another language.

The following application properties currently control the placement and spacing of the currency symbol:

- curam.financial.currencysymbol (symbol and spacing)
- curam.financial.currencysymbolplacement (before or after the monetary amount)

These are global properties set for all users on the system.

Because the placement and spacing are based on global properties, they are not determined by a user's default locale and will not change if a user chooses to view the application in another language.



Currency symbol placement and spacing New configuration

Currency symbol placement and spacing can now be determined based on the user's current locale.

- When a multi-lingual caseworker chooses to view the application in another language, they will see the placement and spacing of the currency symbol presented based on their user locale

New application property has been introduced to control whether the placement and spacing of the currency symbol is determined based on the user's current locale or the global properties.

Property Name	Description
curam.financial.localizedcurrencysymbolplacement The new application property is disabled by default	A setting that defines whether the placement and spacing of the currency symbol is determined based on user locale or global properties. When set to true, the placement and spacing are automatically determined based on user locale using standard Java libraries. When set to false, the placement and spacing are determined by the global **curam.financial.currencysymbol** and **curam.financial.currencysymbolplacement** properties.

14

Currency symbol placement and spacing can now be determined based on the user's current locale.

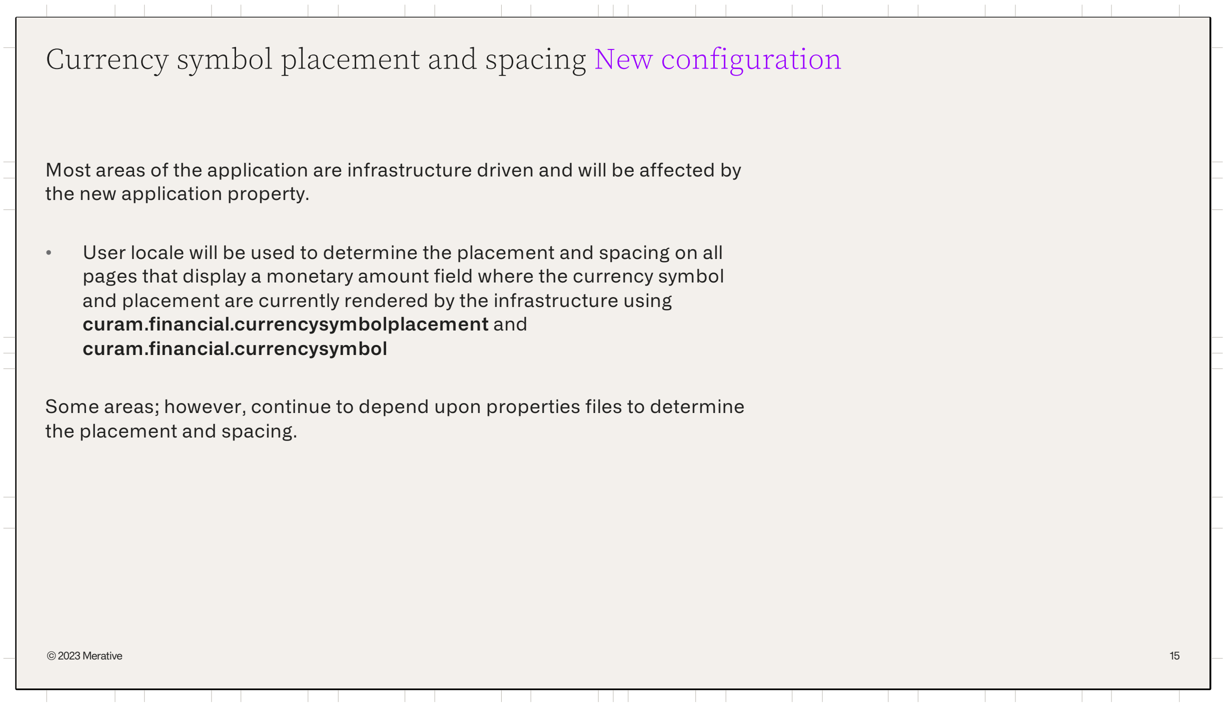
- When a multi-lingual caseworker chooses to view the application in another language, they will see the placement and spacing of the currency symbol presented based on their user locale.

A new application property has been introduced to control whether the placement and spacing of the currency symbol is determined based on the user's current locale or the global properties.

The name of the application property is
“**Curam.financial.localizedcurrencysymbolplacement**”.

- It is a setting that defines whether the placement and spacing of the currency symbol is determined based on user locale or global properties.
- When set to true, the placement and spacing are automatically determined based on user locale using standard Java libraries.
- When set to false, the placement and spacing are determined by the global **curam.financial.currencysymbol** and **curam.financial.currencysymbolplacement** properties.

The new application property is disabled by default.


Currency symbol placement and spacing New configuration

Most areas of the application are infrastructure driven and will be affected by the new application property.

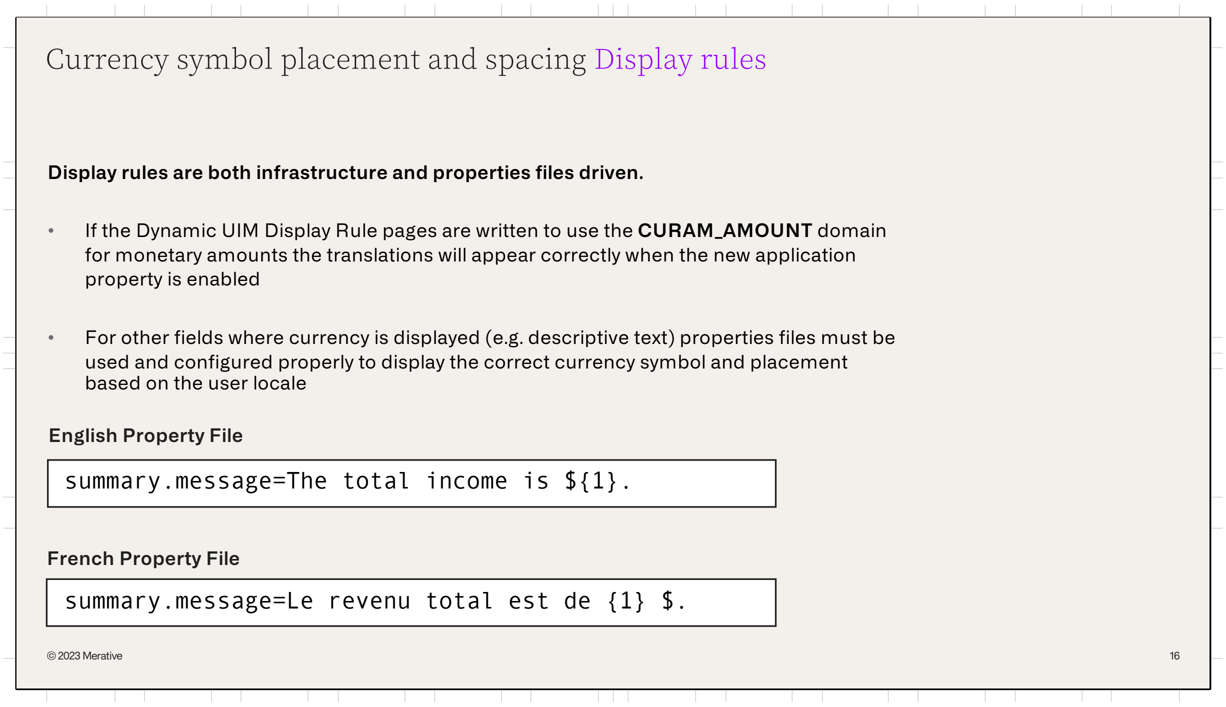
- User locale will be used to determine the placement and spacing on all pages that display a monetary amount field where the currency symbol and placement are currently rendered by the infrastructure using **curam.financial.currencysymbolplacement** and **curam.financial.currencysymbol**

Some areas; however, continue to depend upon properties files to determine the placement and spacing.

Most areas of the application are infrastructure driven and will be affected by the new application property.

- User locale will be used to determine the placement and spacing on all pages that display a monetary amount field where the currency symbol and placement are currently rendered by the infrastructure using the **curam.financial.currencysymbolplacement** and **curam.financial.currencysymbol** application properties.

Some areas; however, depend upon property files to determine the placement and spacing, and this will continue.



Currency symbol placement and spacing Display rules

Display rules are both infrastructure and properties files driven.

- If the Dynamic UIM Display Rule pages are written to use the **CURAM_AMOUNT** domain for monetary amounts the translations will appear correctly when the new application property is enabled
- For other fields where currency is displayed (e.g. descriptive text) properties files must be used and configured properly to display the correct currency symbol and placement based on the user locale

English Property File

```
summary.message=The total income is ${1}.
```

French Property File

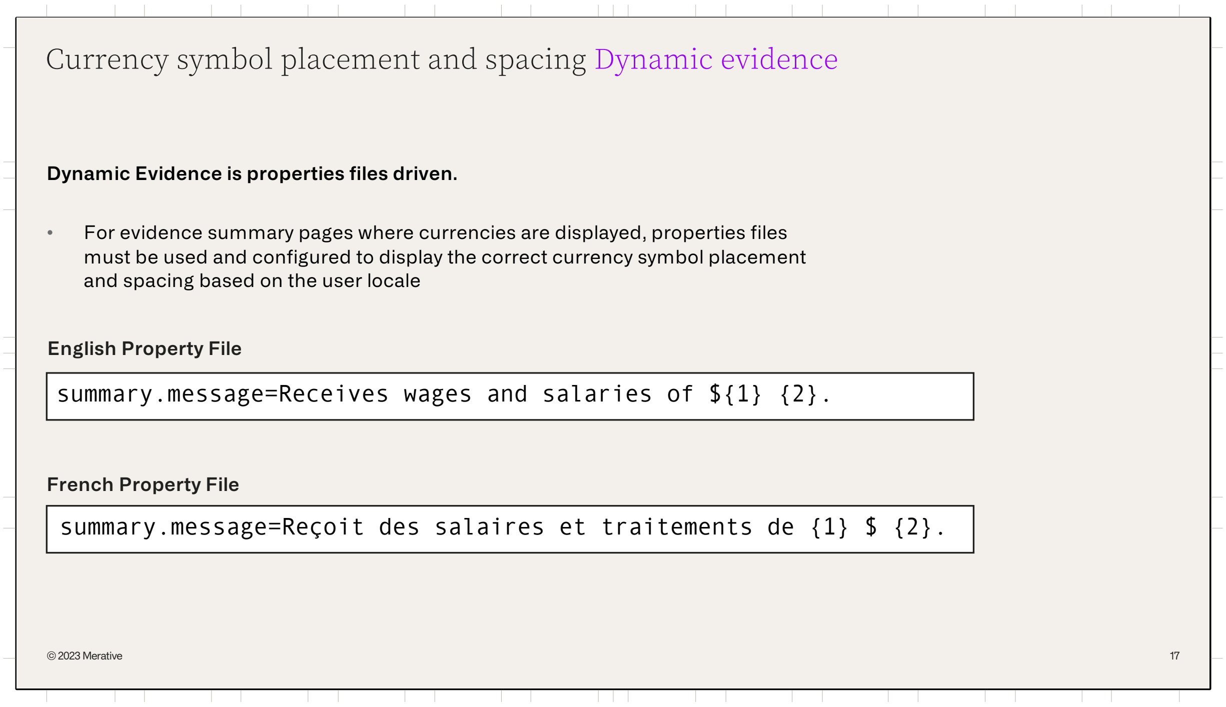
```
summary.message=Le revenu total est de {1} $.
```

Display rules are both infrastructure and properties files driven.

- If the Dynamic UIM Display Rule pages are written to use the **CURAM_AMOUNT** domain for monetary amounts the translations will appear correctly when the new application property is enabled.

- For other fields where currency is displayed (e.g. descriptive text) properties files must be used and configured properly to display the correct currency symbol and placement based on the user locale

The slide shows an example of how an English property file and French property file would be used.


Currency symbol placement and spacing Dynamic evidence

Dynamic Evidence is properties files driven.

- For evidence summary pages where currencies are displayed, properties files must be used and configured to display the correct currency symbol placement and spacing based on the user locale

English Property File

```
summary.message=Receives wages and salaries of ${1} {2}.
```

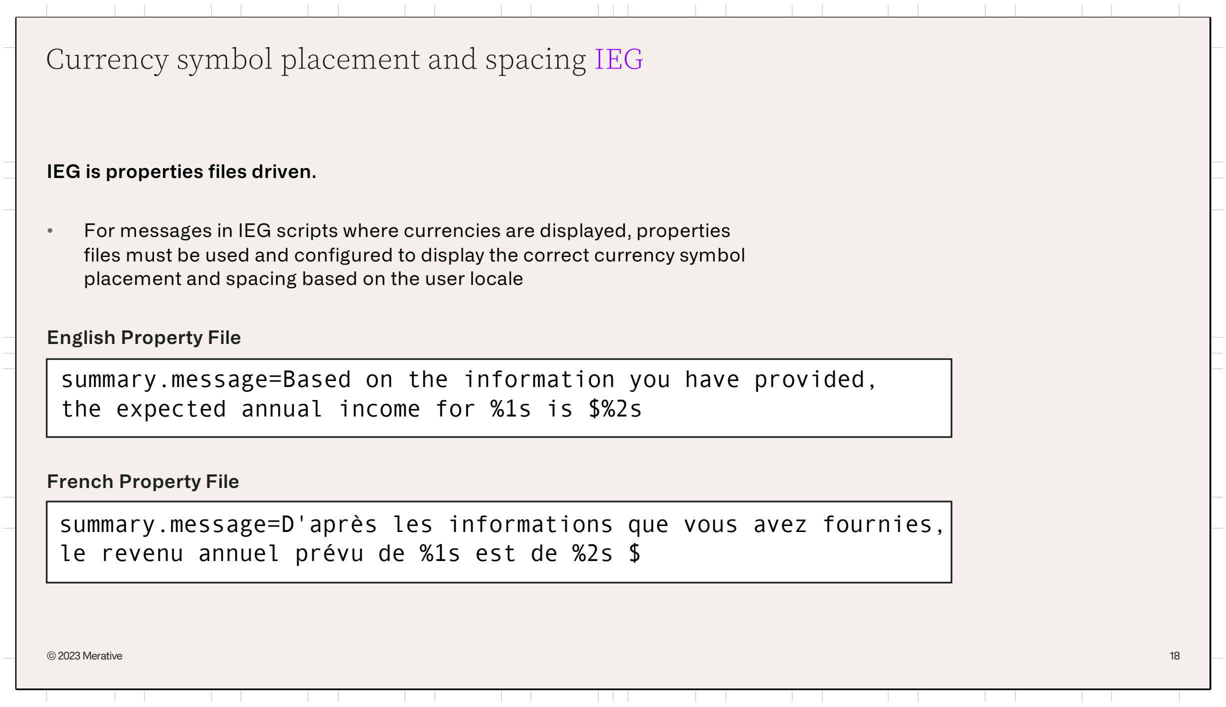
French Property File

```
summary.message=Reçoit des salaires et traitements de {1} $ {2}.
```

Dynamic Evidence is properties files driven.

For evidence summary pages where currencies are displayed, properties files must be used and configured properly to display the correct currency symbol placement and spacing based on the user locale.

The slide shows an example of how an English property file and French property file would be used.



Currency symbol placement and spacing IEG

IEG is properties files driven.

- For messages in IEG scripts where currencies are displayed, properties files must be used and configured to display the correct currency symbol placement and spacing based on the user locale

English Property File

```
summary.message=Based on the information you have provided,
the expected annual income for %1s is $%2s
```

French Property File

```
summary.message=D'après les informations que vous avez fournies,
le revenu annuel prévu de %1s est de %2s $
```

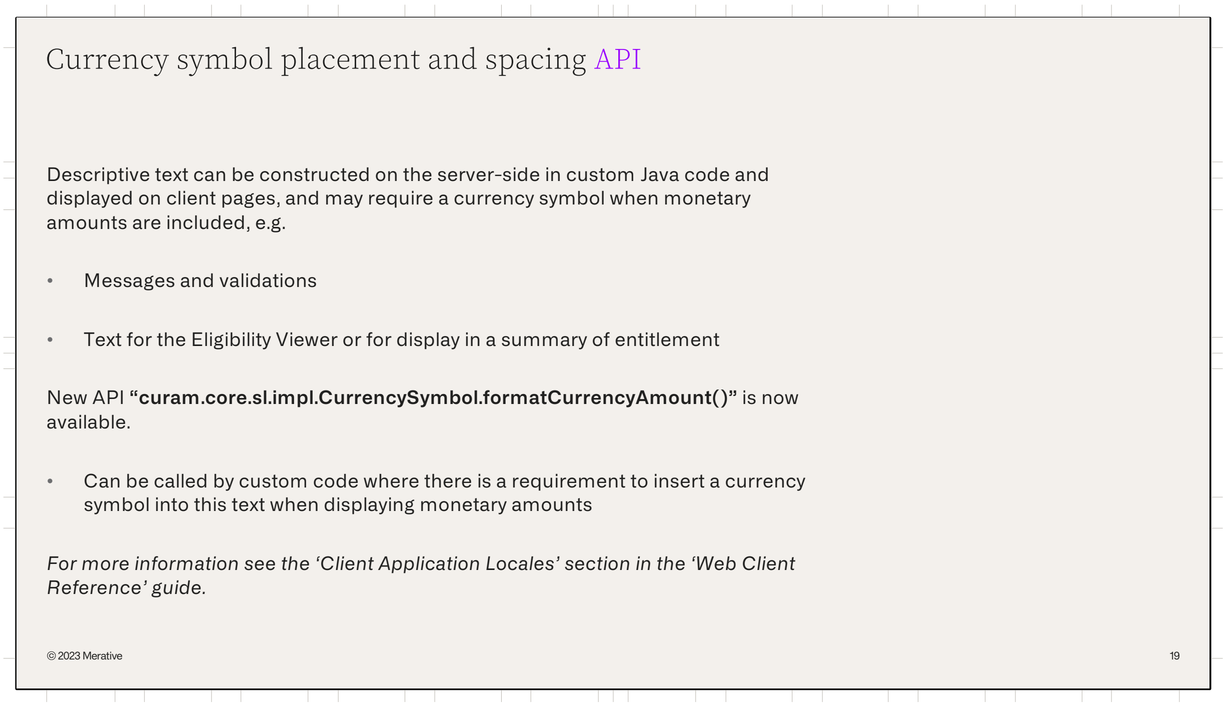
© 2023 Merative

18

IEG is properties files driven.

For messages in IEG scripts where currencies are displayed, properties files must be used and configured to display the correct currency symbol placement and spacing based on the user locale.

The slide shows an example of how an English property file and French property file would be used.



Currency symbol placement and spacing API

Descriptive text can be constructed on the server-side in custom Java code and displayed on client pages, and may require a currency symbol when monetary amounts are included, e.g.

- Messages and validations
- Text for the Eligibility Viewer or for display in a summary of entitlement

New API **"curam.core.sl.impl.CurrencySymbol.formatCurrencyAmount()"** is now available.

- Can be called by custom code where there is a requirement to insert a currency symbol into this text when displaying monetary amounts

For more information see the 'Client Application Locales' section in the 'Web Client Reference' guide.

In some application areas, there may be a requirement to construct descriptive text on the server-side in custom Java code and display it on client pages. You might need to insert a currency symbol into these messages when rendering monetary amounts. For example, this might be a requirement for:

• Messages and validations that are constructed in Java code

• Text that is constructed in Java code for the Eligibility Viewer

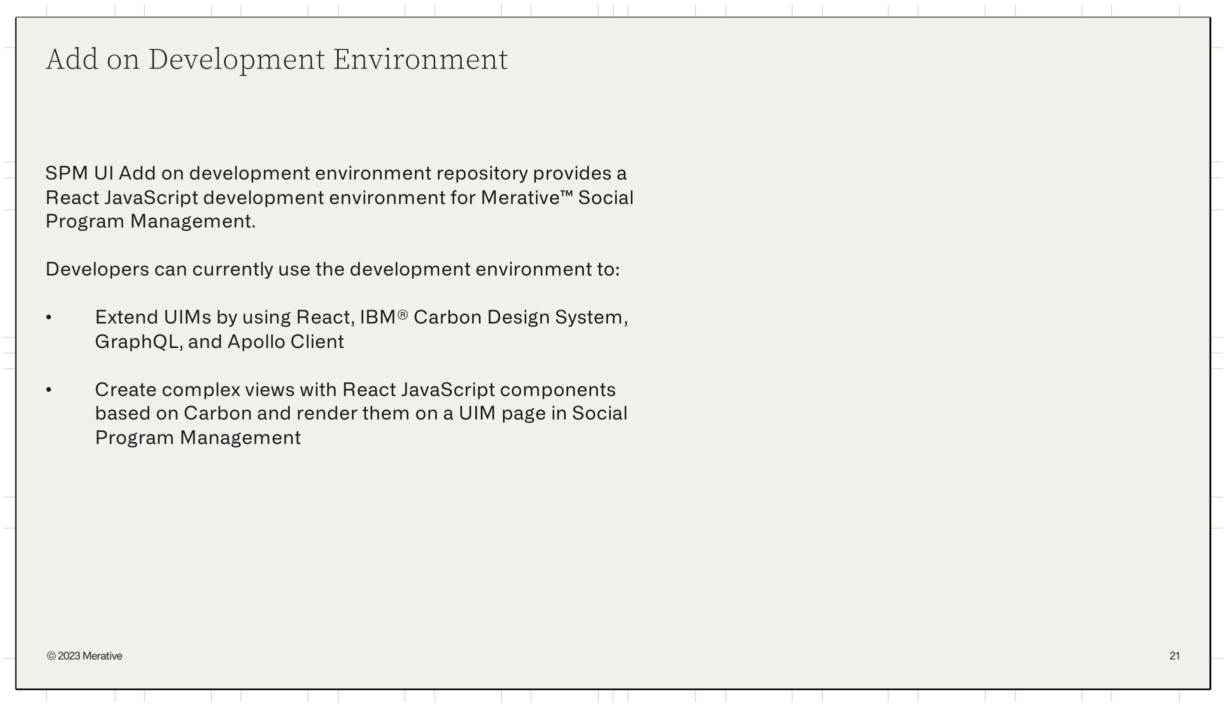
• Text that is constructed in Java code to display a summary of entitlement

Social Program Management now provides an API named curam.core.sl.impl.CurrencySymbol.formatCurrencyAmount() that can be called by custom code where there is a requirement to insert a currency symbol into this text when displaying monetary amounts

When called, this API inserts the appropriate currency symbol for a given monetary amount. The API contains similar logic that is used to format currency symbols for CURAM_AMOUNT domains.

MERATIVE

Add on Development Environment v1.1


Add on Development Environment

SPM UI Add on development environment repository provides a React JavaScript development environment for Merative™ Social Program Management.

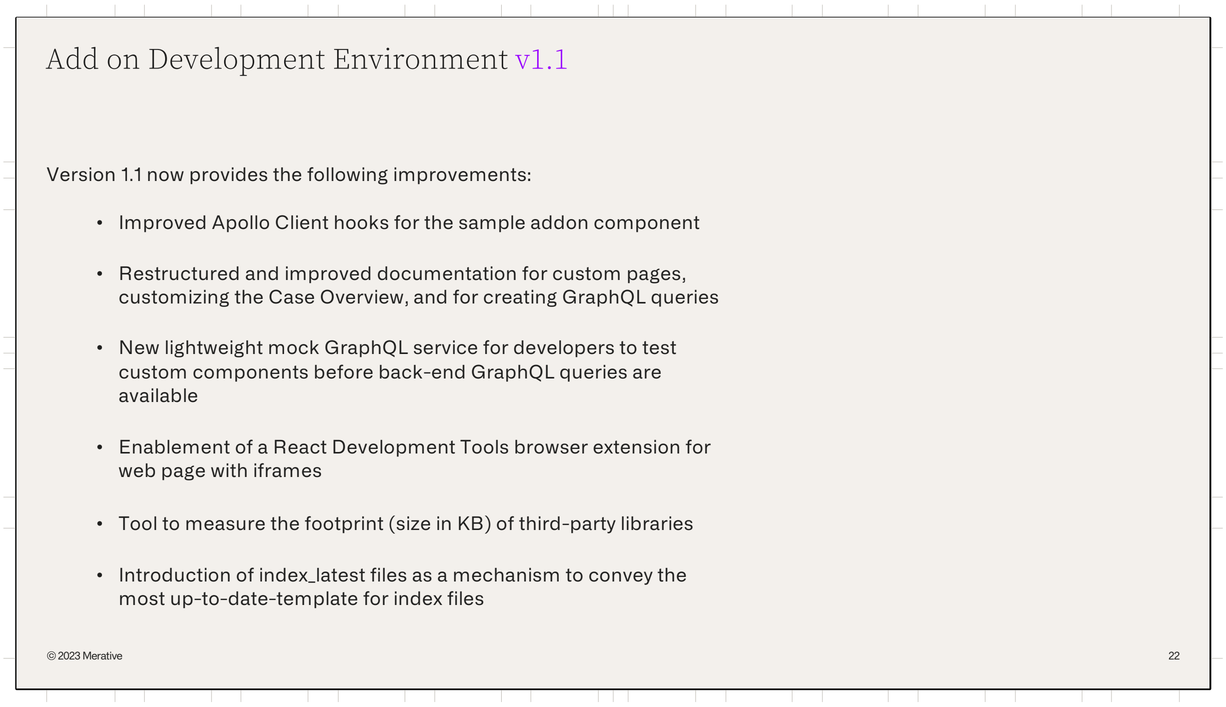
Developers can currently use the development environment to:

- Extend UIMs by using React, IBM® Carbon Design System, GraphQL, and Apollo Client
- Create complex views with React JavaScript components based on Carbon and render them on a UIM page in Social Program Management

The SPM UI Add on development environment (v1 released with 8.0.1) is an open source repository that provides a React JavaScript development environment for Merative™ Social Program Management.

Developers can use the development environment to:

- Extend UIMs by using React, IBM® Carbon Design System, GraphQL, and Apollo Client.
- Create complex views with React JavaScript components based on Carbon and render them on a UIM page in Social Program Management.


Add on Development Environment v1.1

Version 1.1 now provides the following improvements:

- Improved Apollo Client hooks for the sample addon component
- Restructured and improved documentation for custom pages, customizing the Case Overview, and for creating GraphQL queries
- New lightweight mock GraphQL service for developers to test custom components before back-end GraphQL queries are available
- Enablement of a React Development Tools browser extension for web page with iframes
- Tool to measure the footprint (size in KB) of third-party libraries
- Introduction of index_latest files as a mechanism to convey the most up-to-date-template for index files

© 2023 Merative 22

The following improvements to the SPM UI add on development environment are available in Version 1.1:

- Better structure of the Apollo Client hooks for the sample addon component in the carbon-addons-devenv package.
- Restructured documentation, now including an overview diagram.

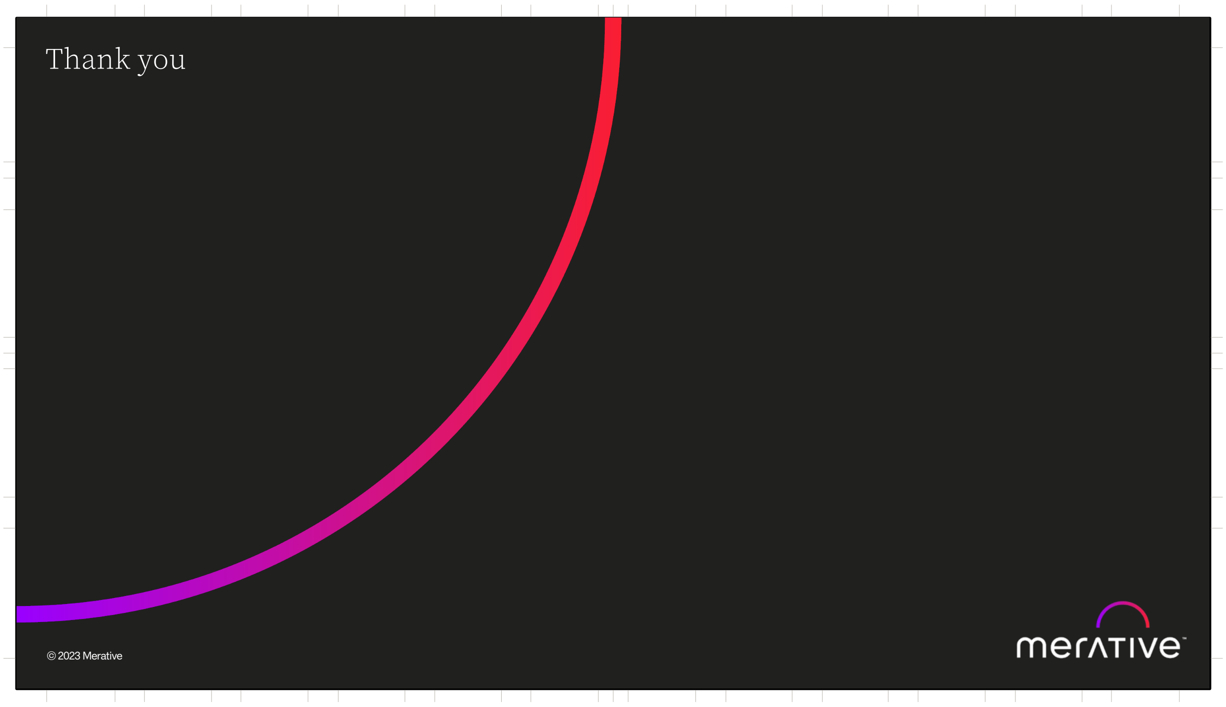
Improved guidance for customization, both for creating custom pages and for customizing the Case Overview. Improved guidance about creating GraphQL queries.

- The provision of a lightweight mock GraphQL service that serves data from the file system and can be used within this sandboxed environment for testing purposes. You can use the mock GraphQL service to test your custom components before production GraphQL queries development is completed.
- Enablement of the React Development Tools browser extension to view the composition of React JavaScript components on web page with iframes.
- The provision of a tool that shows a breakdown of the generated JavaScript files in terms

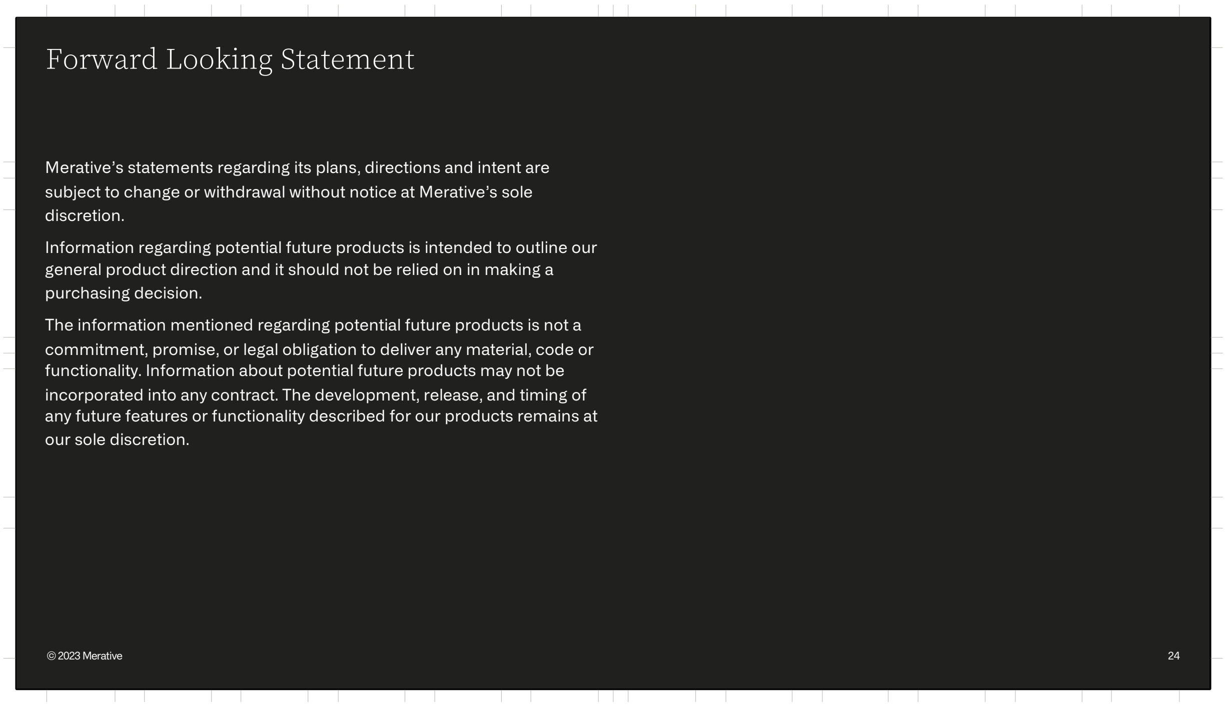
of how much footprint (size in KB) each third-party library contributes to their overall size.

- The introduction of index_latest files as a mechanism to convey the most up-to-date-template for index files. You can diff index_latest files with your version of the index.js or index.scss files to see any differences in the underlying directory structure.

Additional information is available in the changelog https://github.com/merative/spm-ui-addon-devenv/releases and documentation https://merative.github.io/spm-ui-addon-devenv/



Thank you



Forward Looking Statement

Merative's statements regarding its plans, directions and intent are subject to change or withdrawal without notice at Merative's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

merative™