Metrics

The [FileEventHistory.Event](https://developer.android.com/reference/android/service/autofill/FillEventHistory.Event.html) class provides a new [TYPE\_CONTEXT\_COMMITTED](https://developer.android.com/reference/android/service/autofill/FillEventHistory.Event.html#TYPE_CONTEXT_COMMITTED) event that autofill services can use to get the following information:

* The datasets that the user selected.
* The datasets that the user ignored.
* The fields in the selected datasets that the user changed.
* The fields that match the user data that the service sets for [field classification](https://developer.android.com/reference/android/service/autofill/AutofillService.html#FieldClassification) purposes.

For more information, see the [TYPE\_CONTEXT\_COMMITTED](https://developer.android.com/reference/android/service/autofill/FillEventHistory.Event.html#TYPE_CONTEXT_COMMITTED) constant.

Autofill services can also identify the datasets that the user selected by using the [getDatasetIds()](https://developer.android.com/reference/android/service/autofill/SaveRequest.html#getDatasetIds) method of the [SaveRequest](https://developer.android.com/reference/android/service/autofill/SaveRequest.html) class.

Input sanitization

The autofill framework provides sanitizer APIs that services can use to clean a field value before a save request is triggered. The sanitization is useful to avoid triggering save requests when an app formats a value that a service just filled out. For example, consider the following flow in an app that has a field for a credit card number:

1. The autofill service has credit card datasets in which the credit card number is stored in the nnnnnnnnnnnnnnnnformat (where n is a digit of the credit card number).
2. The service fills out the field with the data mentioned in the previous step.
3. The client app has formatting logic that adds a space every four digits of the credit card number. As a result, the field contains the data in the format nnnn nnnn nnnn nnnn.
4. The formatting of the data triggers a save request and the save UI is shown for no reason.

Autofill services can avoid the extra save request by adding a sanitizer that removes the spaces added by the client app. For more information, see the [addSanitizer()](https://developer.android.com/reference/android/service/autofill/SaveInfo.Builder.html#addSanitizer) method of the [SaveInfo.Builder](https://developer.android.com/reference/android/service/autofill/SaveInfo.Builder.html) class.

The autofill framework provides sanitizers for text and date values, see the [TextValueSanitizer](https://developer.android.com/reference/android/service/autofill/TextValueSanitizer.html) and [DateValueSanitizer](https://developer.android.com/reference/android/service/autofill/DateValueSanitizer.html) classes.

UI improvements

* Autofill services can provide a custom header and footer in the autofill save UI. For more information, see the [setHeader()](https://developer.android.com/reference/android/service/autofill/FillResponse.Builder.html#setHeader(android.widget.RemoteViews)) and [setFooter()](https://developer.android.com/reference/android/service/autofill/FillResponse.Builder.html#setFooter(android.widget.RemoteViews)) methods.

**Note:** Currently, the header and footer scroll with the datasets. This behavior might change in future releases.

* Autofill services can apply a series of updates to the save UI if a condition is satisfied. The series of updates are useful in cases where the screen has multiple fields (like username, password and address fields) but some are optional. For more information, see the [batchUpdate()](https://developer.android.com/reference/android/service/autofill/CustomDescription.Builder.html#batchUpdate) method of the [CustomDescription.Builder](https://developer.android.com/reference/android/service/autofill/CustomDescription.Builder.html) class.
* The [ImageTransformation](https://developer.android.com/reference/android/service/autofill/ImageTransformation.html) class accepts a content description attribute, which makes it compatible with the accessibility features of the platform. [ImageTransformation.Builder](https://developer.android.com/reference/android/service/autofill/ImageTransformation.Builder.html) class.
* The autofill framework provides a new [DateTransformation](https://developer.android.com/reference/android/service/autofill/DateTransformation.html) class that services can use to display a string representation of a date in a child view of a [CustomDescription](https://developer.android.com/reference/android/service/autofill/CustomDescription.html) object. This is useful to easily display dates in the save UI, for example, the expiration date of a credit card in MM/yyyy format.
* Autofill services can specify the view that commits the autofill context when tapped. In other words, apps can specify the view that serves as a *submit* button. This is useful in scenarios where the client app clears the views before the autofill context is automatically committed. For more information, see the [setTriggerId()](https://developer.android.com/reference/android/service/autofill/SaveInfo.Builder.html#setTriggerId) method of the [SaveInfo.Builder](https://developer.android.com/reference/android/service/autofill/SaveInfo.Builder.html) class.

**Note:** Currently, when this option is used, the content of the view triggering the commit is hidden after it's clicked. This behavior is fixed in the next release.

Dataset filtering

Datasets in Android P can include an optional regular expression that serves as a filter pattern for a field. The fill UI uses the filter pattern to decide which datasets to show. For more information, see [Filtering](https://developer.android.com/reference/android/service/autofill/Dataset.html#Filtering).

An autofill service can set the dataset filter pattern for a field to null, which disables filtering for the dataset. This approach is recommended when setting the values of fields that contain sensitive data, such as passwords. For more information, see the [setValue()](https://developer.android.com/reference/android/service/autofill/Dataset.html#setValue) method of the [Dataset](https://developer.android.com/reference/android/service/autofill/Dataset.html) class.

Compatibility mode

The autofill framework provides a compatibility mode allowing services to work with apps that display their fields in a virtual structure and don't implement the APIs for such structures, such as browsers. For more information, see[custom views with virtual structure](https://developer.android.com/guide/topics/text/autofill.html#virtual).

The compatibility mode allows autofill services to use the accessibility virtual structure for autofill purposes. Compatibility mode uses the accessibility APIs to get the virtual structure, when needed.

Compatibility mode has the following limitations:

* A save request is only triggered when the service uses the [FLAG\_SAVE\_ON\_ALL\_VIEWS\_INVISIBLE](https://developer.android.com/reference/android/service/autofill/SaveInfo.html#FLAG_SAVE_ON_ALL_VIEWS_INVISIBLE) flag.
* The text value of the nodes might not be available in the [onSaveRequest(SaveRequest, SaveCallback)](https://developer.android.com/reference/android/service/autofill/AutofillService.html#onSaveRequest(android.service.autofill.SaveRequest,%20android.service.autofill.SaveCallback)) method.
* When you point Chrome to a different URL, the new page might not be filled out. Restarting the browser might fix the issue.
* The urlBarResourceId metadata property only supports one value per app.
* Compatibility mode doesn't work with the version of Chrome bundled with Android P due to an [accessibility issue](https://www.google.com/url?sa=D&q=https%3A%2F%2Fbugs.chromium.org%2Fp%2Fchromium%2Fissues%2Fdetail%3Fid%3D805014). You must install [Chrome Beta](https://play.google.com/store/apps/details?id=com.chrome.beta) or [Chrome Canary](https://play.google.com/store/apps/details?id=com.chrome.canary).
* You must manually whitelist the apps that you want to support compatibility mode. For example to whitelist Chrome and Chrome Beta, use the following command:

adb shell settings put global autofill\_compat\_allowed\_packages com.android.chrome:com.chrome.beta

For more information, see [compatibility mode](https://developer.android.com/reference/android/service/autofill/AutofillService.html#CompatibilityMode).

Miscellaneous

* Autofill services can get the [android:id](https://developer.android.com/reference/android/view/View.html#attr_android:id) attribute in text views. The [android:text](https://developer.android.com/reference/android/widget/TextView.html#attr_android:text) attribute of some views might not be meaningful to the service, for example, when the view is localized. Autofill services can use the android:id attribute as additional input to determine the meaning of the data. For more information, see the [getTextIdEntry()](https://developer.android.com/reference/android/app/assist/AssistStructure.html#getTextIdEntry) method of the [AssistStructure](https://developer.android.com/reference/android/app/assist/AssistStructure.html) class.
* The [getWebScheme()](https://developer.android.com/reference/android/app/assist/AssistStructure.ViewNode.html#getWebScheme) method returns the scheme part of a URL if the node represents an HTML document. Autofill services can use the web scheme in scenarios where a website is associated with an app. For more information see, [Associate website and mobile app data](https://developer.android.com/guide/topics/text/autofill.html#associate).
* The [AssistStructure.ViewNode](https://developer.android.com/reference/android/app/assist/AssistStructure.ViewNode.html) class exposes the following information about the size of text fields:
  + Maximum width (in ems) of the text associated with the node, see the [getMaxTextEms()](https://developer.android.com/reference/android/app/assist/AssistStructure.ViewNode.html#getMaxTextEms) method.
  + Maximum length of the text associated with this node, see the [getMaxTextLength()](https://developer.android.com/reference/android/app/assist/AssistStructure.ViewNode.html#getMaxTextLength) method.
  + Minimum width in ems of the text associated with the node, see the [getMinTextEms()](https://developer.android.com/reference/android/app/assist/AssistStructure.ViewNode.html#getMinTextEms) method.
* Autofill services can provide an [IntentSender](https://developer.android.com/reference/android/content/IntentSender.html) to the onSuccess() method of a save request. The IntentSenderis useful if the service needs the user to perform extra work before it can process the request. For more information, see the [onSuccess()](https://developer.android.com/reference/android/service/autofill/SaveCallback.html#onSuccess(IntentSender)) method of the [SaveCallback](https://developer.android.com/reference/android/service/autofill/SaveCallback.html) class.
* Autofill services can disable autofill for specific client apps or activities. Disabling autofill is useful in cases where the service has blacklisted an app. Furthermore, disabling autofill can improve device performance because the autofill framework triggers fewer autofill requests. For more information, see the [disableAutofill()](https://developer.android.com/reference/android/service/autofill/FillResponse.Builder.html#disableAutofill(long)) method and the [FLAG\_DISABLE\_ACTIVITY\_ONLY](https://developer.android.com/reference/android/service/autofill/FillResponse.html#FLAG_DISABLE_ACTIVITY_ONLY) flag.