Item 262
git_comments:

git_commits:

1. **summary:** [CXF-3987] Removing confusing Multipart checks when reading plain multipart/form-data payloads
   message: [CXF-3987] Removing confusing Multipart checks when reading plain multipart/form-data payloads git-svn-id: https://svn.apache.org/repos/asf/cxf/trunk@1221511 13f79535-47bb-0310-9956-ffa450edef68
   label: code-design

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
   description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?
2. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
   description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null ||

!MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?

3. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
   description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?
4. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
   description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?
5. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
   description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target =

DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?

6. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
   description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?
7. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
   description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content

disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?

8. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
   description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?
9. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
   description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?
   label: code-design
10. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
    description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for

(Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?
label: documentation

11. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
    description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?
12. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
    description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?
13. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
    description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({

@Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?

14. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?
15. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note

that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?

16. **summary:** incompatible change in JAX-RS from 2.5.0 to 2.5.1
description: One of my multipart tests is now failing with a 400. The 400 is thrown from FormUtils in the runtime. It runs just fine in 2.5.0. My function looks like: {code} @Produces("text/html") @Consumes("multipart/form-data") @Path("/processFormTextToJson") @Descriptions({ @Description(value = "Accepts text from an HTML form, returns analysis results in JSON.", target = DocTarget.METHOD), @Description(value = "Json containing all of the analysis results", target = DocTarget.RETURN) }) public Response processFormTextToJson(@Description(value = "Json specification of the processing options.", target = DocTarget.PARAM) @Multipart(value = "options") String optionsString, @Description(value = "Input text", target = DocTarget.PARAM) @Multipart(value = "data") InputStream data) { {code} and the code leading to the exception in CXF is: {code} public static void populateMapFromMultipart(MultivaluedMap<String, String> params, Annotation[] anns, MultipartBody body, boolean decode) { List<Attachment> atts = body.getAllAttachments(); for (Attachment a : atts) { ContentDisposition cd = a.getContentDisposition(); if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} The annotations present are: {noformat} [@org.apache.cxf.jaxrs.model.wadl.Description(title=, target=param, value=Json specification of the processing options., lang=, docuri=), @org.apache.cxf.jaxrs.ext.multipart.Multipart(value=options, required=true, type=*/*)] {noformat} Note that the required flag is on. cd is null. In other words, even if the names match, if there is no content disposition, we get a 400. Is that really right? Why require a cd? What's wrong with the plain old name field of the part?

jira_issues_comments:

1. Hi Benson How did it work before ? @Multipart in case of form-data just points to an expected Content-Disposition's name parameter. Can you show the sample payload please ? Sergey
2. In 2.5.0 it was: {code:java} if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { throw new WebApplicationException(415); } {code} in 2.5.1: {code:java} if (cd == null || !MULTIPART_FORM_DATA_TYPE.equalsIgnoreCase(cd.getType()) || cd.getParameter("name") == null) { Multipart id = AnnotationUtils.getAnnotation(anns, Multipart.class); if (id == null || id.required()) { throw new WebApplicationException(400); } else { return; } } {code} It is actually less strict in 2.5.1 as we give users a chance to state that a given part is not required to be present
3. No content disposition? start= instead? {noformat} ---------------------------- ID: 1 Address: http://localhost:15000/r4dws/services/doc/processFormTextToJson Encoding: ISO-8859-1 Http-Method: POST Content-Type: multipart/form-data; type="application/json"; boundary="uuid:00b0aabf-6dc8-4a22-a5c4-a5d2eaa2264e"; start="<options>"; start-info="application/json" Headers: {Accept=[text/html], cache-control=[no-cache], connection=[keep-alive], content-type=[multipart/form-data; type="application/json"; boundary="uuid:00b0aabf-6dc8-4a22-a5c4-a5d2eaa2264e"; start="<options>"; start-info="application/json"], host=[localhost:15000], pragma=[no-cache], transfer-encoding=[chunked], user-agent=[Apache CXF 2.5.1]} Payload: --uuid:00b0aabf-6dc8-4a22-a5c4-a5d2eaa2264e Content-Type: application/json Content-Transfer-Encoding: binary Content-ID: <options> {"languageDetection":{"language":"UNKNOWN","strategy":"MULTIPLE"},"text":null} --uuid:00b0aabf-6dc8-4a22-a5c4-a5d2eaa2264e Content-Type: application/octet-stream Content-Transfer-Encoding: binary Content-ID: <data> 1 de junio de 2003, 18h01 Bush y Chirac sonríen para las cámaras en tensa cumbre del G-8
4. WebClient code that produced this ... {code} List<Attachment> atts = new LinkedList<Attachment>(); atts.add(new Attachment("options", "application/json", ptio)); atts.add(new Attachment("data", "application/octet-stream", FileUtils readFileToByteArray(new File(btRoot, "rlp/samples/data/Spanish-French-German.txt")))); {code}
5. In 2.5.0, the wire data looks rather similar, so the mystery is, why is the ContentDisposition reference null? {noformat} ID: 6 Address: http://localhost:51102/r4dws/services/doc/processFormTextToJson Encoding: ISO-8859-1 Http-Method: POST Content-Type: multipart/form-data; type="application/json"; boundary="uuid:22187f34-ab19-411a-b13e-5af4486ef93c"; start="<options>"; start-info="application/json" Headers: {Accept=[text/html], cache-control=[no-cache], connection=[keep-alive], content-type=[multipart/form-data; type="application/json"; boundary="uuid:22187f34-ab19-

411a-b13e-5af4486ef93c"; start="<options>"; start-info="application/json"], host=[localhost:51102], pragma=[no-cache], transfer-encoding=[chunked], user-agent=[Apache CXF 2.5.0]} Payload: ^M --uuid:22187f34-ab19-411a-b13e-5af4486ef93c^M Content-Type: application/json^M Content-Transfer-Encoding: binary^M Content-ID: <options>^M ^M {"languageDetection":{"language":"UNKNOWN","strategy":"MULTIPLE"},"text":null}^M --uuid:22187f34-ab19-411a-b13e-5af4486ef93c^M Content-Type: application/octet-stream^M Content-Transfer-Encoding: binary^M Content-ID: <data> {noformat}

6. Now I think I've lost my mind. I added content-disposition to the attachments, and it does not show up in the log message from the in-interceptor on the service side.
7. OK, I see one thing. You can't add a header to an Attachment once you make it. Another jira, coming up.
8. Here's the function I wrote to work around this: {code} private Attachment makeBrowerLikeAttachment(String id, String mediaType, Object data) throws IOException { JsonFactory factory = new MappingJsonFactory(); ByteArrayOutputStream baos = new ByteArrayOutputStream(); JsonGenerator gen = factory.createJsonGenerator(baos); gen.writeObject(data); gen.flush(); MetadataMap<String, String> headers = new MetadataMap<String, String>(false, true); headers.putSingle("Content-ID", id); headers.putSingle("Content-Type", mediaType); headers.putSingle("Content-Disposition", String.format("form-data; name=\"%s\"", id)); return new Attachment(new ByteArrayInputStream(baos.toByteArray()), headers); } {code}
9. **body:** Benson, I removed some of the code which made it into 2.5.1 to minimize the confusion but either way I think it's Not A Problem issue. When we have multipart/form-data payloads, we do expect Content-Disposition with the "form-data" type; if we relax it then we can capture by mistake the data meant to be processed by MultipartProvider. If you disagree then let me know please what do you think can be improved
   label: code-design
10. **body:** Sergey, I think that the issue here is the client side. Absolutely none of the constructors for Attachment build something that corresponds to what a browser does. The documented examples on the client side don't work. Though, breaking working clients in a double-point release might be a cause for concern. I'll add a constructor or a factory for Attachment. --benson
    label: documentation
11. May be I don't understand something, what is wrong with this code which you used: {code:java} MetadataMap<String, String> headers = new MetadataMap<String, String>(false, true); headers.putSingle("Content-ID", id); headers.putSingle("Content-Type", mediaType); headers.putSingle("Content-Disposition", String.format("form-data; name=\"%s\"", id)); return new Attachment(new ByteArrayInputStream(baos.toByteArray()), headers); {code} This code produces a self-contained part with headers and the data.
12. By the way, I updated that FormUtils code to check Content-Id in case of missing Content-Dispositions - which should let your old code continue working without having to add CDs; can we close this JIRA as Duplicate of 3988 and chat about possible Attachment enhancements in JIRA-3988 ?
13. Yes.
14. Duplicate of https://issues.apache.org/jira/browse/CXF-3988